Applied Multivariate Statistics with Python

Theory • Applications • Real Data

Kenneth R. Szulczyk, PhD

Applied Multivariate Statistics with Python

Cover design by Kenneth R. Szulczyk

Edition 1.0, April 2026

Table of Contents

Table of Figures

Chapter 7

Chapter 8

Table of Tables

Preface

This book was developed to support an applied course in statistics that emphasizes understanding, interpretation, and practical skills. Many traditional textbooks rely heavily on software such as SPSS and focus on mechanical procedures. While these tools can be useful, they often limit students' ability to fully understand the methods and apply them in real-world settings.

This text takes a different approach. It integrates statistical concepts with Python programming. It helps students to understand the theory and implement it using a modern, widely used tool. Python is free, flexible, and increasingly important in fields such as business, economics, finance, and data science. By learning Python alongside statistical methods, students develop skills that are directly applicable beyond the classroom.

The structure of this book is intentionally simple. Each topic follows a consistent pattern:

- Introduce the concept
- Explain the intuition
- Apply the method using Python
- Interpret the results

The goal is not only to perform calculations, but to understand what the results mean and how they inform decisions.

This book focuses on practical methods, including multivariate analysis, classification, and clustering techniques. These tools are widely used in research and industry to analyze complex data and identify meaningful patterns.

Ultimately, this text is designed to help students move beyond memorizing formulas and toward thinking critically about data. By combining clear explanations with hands-on coding, students gain both the confidence and the skills needed to apply statistics in real-world situations.

The author provides the datasets on his website and Github:

https://www.szulczyk.net/

https://github.com/KenSzulczyk/Applied-Multivariate-Statistics-with_Python

Chapter 1. Business Statistics Review

Multivariate statistics begins with a strong foundation in business statistics. Before we move into more advanced models, we revisit the essential tools that allow us to analyze and interpret data with confidence. In this chapter, we review key concepts including terminology, hypothesis testing, descriptive statistics, and Pearson correlation. We also examine important probability distributions, such as the normal, t, chi-square, and F distributions. They form the backbone of statistical inference.

This chapter serves as both a refresher and a bridge. Students should view it as an opportunity to consolidate prior knowledge and sharpen their analytical skills. A clear understanding of these concepts is essential before progressing to multivariate models and empirical applications.

This textbook takes an applied approach. In Chapters 1–3, we compute all results by hand to build intuition and a deeper understanding of the underlying mechanics. Beginning in Chapter 4, we transition to Python. We begin to automate these calculations and analyze larger datasets. The remaining chapters build on this foundation, combining statistical theory with practical data analysis.

1.1 Statistics Basics

Statistics refers to both numerical summaries and a scientific discipline [1]. As numerical summaries, statistics include measures, such as means, medians, and percentages that describe data. As a discipline, statistics involves collecting, analyzing, interpreting, and presenting data.

Every dataset contains a story. The role of the analyst is to understand that story and communicate it clearly to others.

1.2 Measurement Scales

Statistics involves numbers, but before we can analyze them, we must measure phenomena. There are four primary scales of measurement: nominal, ordinal, interval, and ratio [2].

Nominal data classify observations into categories without any inherent order. We identify a variable using a name or label. It may be either nonnumeric or numeric. For example, a university may classify students by the school in which they are enrolled using nonnumeric

labels such as Business, Humanities, Education, and so on. Alternatively, we could assign numeric codes to represent the same variable:

- 1 denotes Business
- 2 denotes Humanities
- 3 denotes Education, and so on.

Similarly, political affiliation can be categorized as labor, conservative, liberal democrat, other, or none. Gender is another example of a classification variable.

Ordinal data introduce ranking, but the differences between ranks are not measurable. These data retain the properties of nominal data. However, it adds meaningful order or rank. As with nominal data, we may use either nonnumeric labels or numeric codes. For instance, a university may classify students by class standing using labels such as Freshman, Sophomore, Junior, or Senior. Numeric codes can also be assigned (e.g., 1 denotes Freshman, 2 denotes Sophomore, and so on).

Interval data allow for meaningful differences between values but lack a true zero point. These data retain the properties of ordinal data. The intervals between observations are expressed using a fixed unit of measurement. Interval data are always numeric. A variable is continuous if it is measured on a scale such as length, time, or weight. It can take any value within a range. For example, Melissa has an SAT score of 1985, while Kevin has an SAT score of 1880. Melissa scored 105 points more than Kevin.

Ratio data include a true zero. It allows for meaningful comparisons of magnitude. These data have all the properties of interval data, and ratios between values are meaningful. Ratio data are always numeric and include an absolute zero point. They may also be continuous. For example, a retail store charges $200 for a book, while an online store charges $100 for the same book. The ratio property shows that the retail store charges twice the online price.

We can further classify data as categorical or quantitative. The appropriate statistical analysis depends on whether the variable is categorical or quantitative. In general, quantitative data allow for a wider range of statistical techniques.

Categorical data use labels or names to identify attributes of each element and are often referred to as qualitative data. They may be measured on either the nominal or ordinal scale and can be represented using numeric or nonnumeric values. However, this type of data limits the range of statistical analyses that can be performed. A variable is discrete if it can take only certain values. For example, the number of interactions can take only whole-number values and is therefore discrete. When working with categorical data, we must ensure that categories are mutually exclusive and exhaustive.

- Mutually exclusive: Each case belongs to only one category
- Exhaustive: All possible outcomes must be included

A common binary, or dichotomous, variable has only two values. For example, a person may be classified as either male or female.

Both continuous and discrete variables may be measured on an interval scale. For example, temperature is measured on an interval scale. A rise of 10 degrees has the same meaning whether it occurs near freezing or on a hot summer day. However, temperature has an arbitrary zero. Zero degrees Celsius represents the freezing point of water, while the Fahrenheit scale sets freezing at 32 degrees. Thus, the statement "It is twice as hot in this room as it is outside" does not make sense. If it is 10°C outside and 20°C inside, it is not twice as hot. In Fahrenheit, the corresponding temperatures would be 50°F and 68°F.

In contrast, a ratio scale has a true zero. Therefore, the statement "I am twice as heavy as you are" is meaningful, because weight has the same zero point whether measured in kilograms or pounds.

Understanding these scales is essential because they determine which statistical techniques are appropriate for analysis.

1.3 Functions and Models

Statistical analysis examines relationships between variables. The dependent variable represents the outcome of interest, while independent variables explain variation in that outcome. We denote the dependent variable (DV) as y and the independent variable (IV) as x:

$$y = f(x)$$

In an experiment, we manipulate x to observe its effect on y. We assume that y is at least partly determined by x. Although we typically vary x in an experimental setting, we may also include additional variables, such as z, to control for their influence on y. These variables are called covariates. We do not directly manipulate covariates, but we include them because they may affect the dependent variable. In practice, both x and z are often referred to as independent variables. In the expression below, x is the variable of primary interest, while z is a covariate:

$$y = f(x, z)$$

Some researchers use the term multivariate only when there are at least two dependent variables. In this book, however, we use the term more broadly to describe any situation involving more than two variables. A common linear model is [3]:

$$y = \beta_0 + \beta_1 x + \beta_2 z + \varepsilon$$

In this model, the dependent variable y depends on x and z. With a sample size of n, we have one equation for each observation. The parameter β_0 is the intercept, while β_1 and β_2 are slopes. The marginal effect measures how y changes when x increases by one unit, holding other variables constant. The term ε represents a white noise process. Some researchers refer to this as an innovation, particularly in time series analysis where ε may not be purely random. If we omit an important variable, then ε may capture the influence of that missing factor.

The following equation represents cross-sectional data, where each i refers to a unit in the sample:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 z_i + \varepsilon_i$$

For time series data, each t refers to a point in time:

$$y_t = \beta_0 + \beta_1 x_t + \beta_2 z_t + \varepsilon_t$$

For pooled or panel data, each i represents a cross-sectional unit and each t represents time:

$$y_{it} = \beta_0 + \beta_1 x_{it} + \beta_2 z_{it} + \varepsilon_{it}$$

We often work with linear functions expressed in matrix form:

$$Y = X \cdot \beta + \varepsilon$$

In this representation, Y and ε are vectors containing the observed values of the dependent variable and the error terms, respectively. Each vector has n rows (observations) and one column. The vector β contains the model parameters. The matrix X includes all explanatory variables; typically, its first column consists of ones to represent the intercept, the second column contains values of x, and the third column contains values of z. In practice, data are often stored in data frames, which resemble large matrices or spreadsheets with rows and columns.

To better understand relationships, we often visualize data. A scatter diagram is a graphical representation of the relationship between two quantitative variables. One variable is plotted on the horizontal axis and the other on the vertical axis. While it is possible to

incorporate a third variable (for example, using color or size), visualizing four or more variables becomes difficult. The overall pattern of the plotted points reveals the nature of the relationship.

A trendline provides an approximation of the linear relationship between variables. We formally estimate these relationships using regression analysis, which we introduce in Chapter 3.

Figure 1.1 presents 100 data points showing a positive relationship between x and y. The red trendline captures this relationship, while each blue dot represents an individual observation. The slope, β, is positive: as x increases, y also increases.

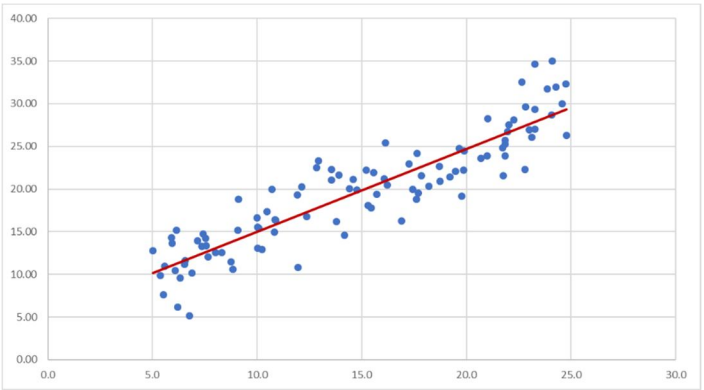


Figure 1.1. Positive Relationship between x and y

Figure 1.2 shows a negative relationship between x and y. The slope, β, is negative: as x increases, y decreases.

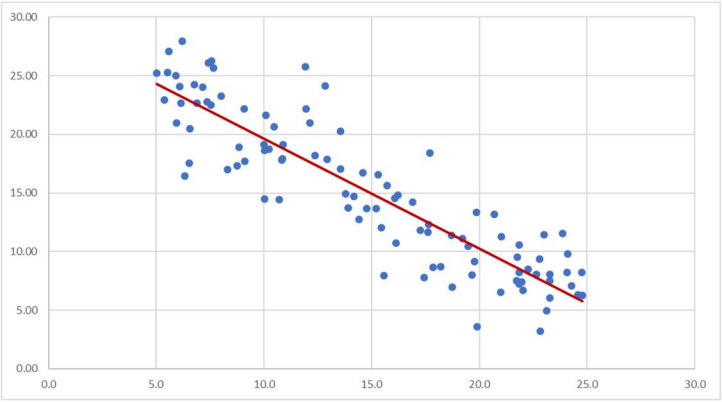


Figure 1.2. Negative Relationship between x and y.

Figure 1.3 illustrates a case where x and y have no relationship. The slope, β, is approximately zero, indicating that it is not statistically significant. In this case, changes in x do not affect y.

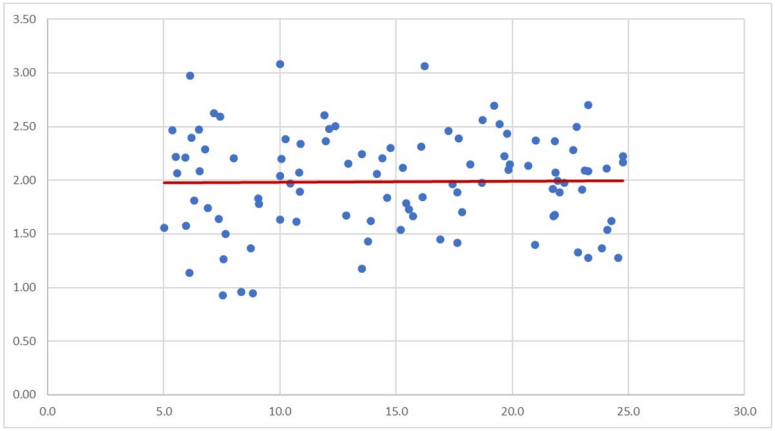


Figure 1.3. No Relationship between x and y.

The examples in this book assume that relationships between variables are linear.

1.4 Hypothesis Testing

In most cases, we do not know the true population parameters. Hypothesis testing provides a formal framework for evaluating claims about these parameters [4]. A population consists of all elements of interest in a study. A sample is a subset of that population. Because populations are often too large to observe. we rely on statistical inference to draw conclusions about population parameters using sample data.

Researchers typically collect data through sample surveys. In principle, every member of the population should have a probability of being selected. After collecting a sample, we perform statistical analysis and test hypotheses about population characteristics. A random sample is one in which every individual in the population has a chance of being selected.

In some cases, population parameters are known. Governments, for example, conduct censuses that collect data on entire populations. The U.S. Census Bureau conducts a comprehensive survey of the U.S. population every ten years. These data are critical for apportioning seats in the House of Representatives and allocating funding for federal programs.

The first step in any study is to clearly define the population. For example, if we want to study cryptocurrency investment among Thai people, the population would be all Thai individuals aged 18 and over. In this case, any of the approximately 54.1 million adult Thais should have a chance of being selected.

The definition of the population always depends on the research question. If we want to survey college students in Thailand, then the population consists of all college students in Thailand. Similarly, if we want to study students at Khon Kaen University regarding the quality of food stalls, then the population includes all students at that university. Every student must have a probability of being selected.

Hypothesis testing allows us to determine whether a statement about a population parameter should be rejected. Because we do not know the true parameter values, we begin with two competing hypotheses. The null hypothesis, denoted by H_0, is a tentative assumption about a population parameter. The alternative hypothesis, denoted by H_a, represents the opposite claim. We use sample data to evaluate these competing statements.

For example, suppose a new teaching method is believed to be more effective than the current method. The alternative hypothesis states that the new method is better, while the null hypothesis states that it is no better than the existing method.

In another example, a new drug is developed to lower blood pressure more effectively than an existing treatment. The alternative hypothesis states that the new drug is more effective. The null hypothesis states that it is not more effective.

We use sample data to challenge the null hypothesis. Consider a soda can labeled as containing at least 0.325 liters. The null hypothesis states that the label is correct, $\mu \geq 0.325$, while the alternative hypothesis states that the label is incorrect, < 0.325. The null hypothesis always includes an equality condition, while the alternative hypothesis reflects the direction of the test on the probability distribution:

H_0: $\mu \geq 0.325$ liters
H_a: $\mu < 0.325$ liters

We collect production data and determine whether the evidence supports H_0 or H_a. If the evidence supports H_0, customers are satisfied. However, firms also avoid overfilling because it reduces profits. If the evidence supports H_a, customers may be dissatisfied. The cans are underfilled.

Another example involves emergency ambulance response times in Khon Kaen, Thailand. Suppose a hospital operates approximately 25 ambulances and aims to achieve an average response time of 10 minutes or less. Faster response times reduce the risk of mortality.

The director of medical services can test whether this goal is being met. Let denote the population mean response time:

H_0: $\mu \leq 10$ minutes

H_a: $\mu > 10$ minutes

If the data support H_o, the service meets its target and no action is required. If the data support Ha, the service fails to meet the target. Management may need to adjust policies to improve response times.

Because hypothesis tests rely on sample data, errors are possible. A Type I error occurs when we reject H_o even though it is true [5]. The probability of committing a Type I error is called the level of significance, denoted by α. Common values for α are 1%, 5%, and 10%. For example, if $\alpha = 0.05$, there is a 5% chance of rejecting a true null hypothesis.

Why not reduce α to minimize Type I errors? Lowering α increases the probability of a Type II error. It occurs when we fail to reject H_o even though it is false. The probability of a Type II error is denoted by β. Unlike α, β is often difficult to determine and is frequently estimated using simulation methods such as Monte Carlo analysis.

The power of a test is defined as $1 - \beta$, which represents the probability of correctly rejecting a false null hypothesis. Researchers often use simulations to compare statistical tests and identify those with higher power.

Type I and Type II errors are inversely related. For example, if we set α=0.01, we obtain a corresponding β. If we reduce α further to 0.0001, the probability of a Type II error increases. As the probability of a Type I error decreases, the probability of a Type II error increases, and vice versa. We can increase the power of a test in several ways:

- Increase the sample size
- Use a one-tailed test instead of a two-tailed test when appropriate
- Increase the effect size by expanding the range of the independent variables

Type I and Type II errors correspond to false positives and false negatives. Consider a medical test for cancer. A Type I error means the patient has cancer, but the test indicates they do not. A Type II error means the patient does not have cancer, but the test indicates that they do.

A Type I error is generally more serious in this context because the patient may leave the hospital believing they are healthy. A Type II error may cause stress, but follow-up tests can often correct the diagnosis.

Table 1.1 summarizes the possible outcomes of hypothesis testing. The diagonal elements represent correct decisions. The off-diagonal elements represent errors.

In Chapters 7 and 8, we revisit this framework in the context of prediction models. It is referred to as a confusion matrix.

Table 1.1. Hypothesis Testing Summary

	Population Parameters	
Statistical Test	H_0 is true	H_0 is false
Accept H_0	Correct	Type II Error
Reject H_0	Type I Error	Correct

1.5 Descriptive Statistics and Correlation

We use descriptive statistics to describe, organize, and summarize the main features of a dataset. In research, descriptive statistics help introduce readers to the data before conducting formal analysis. These statistics can be presented in tables or displayed using graphs.

Measures of central tendency identify the center of the data. The most common measures include:

- Mean: the mathematical average
- Median: the middle value when the data are sorted from lowest to highest
- Mode: the most frequently occurring value

We also report measures of variability, which describe how spread out the data are. Common measures include:

- Minimum: the smallest value
- Maximum: the largest value
- Range: the difference between the maximum and minimum
- Standard deviation: the overall variability of the dataset
- First and third quartiles: when the data are ordered from smallest to largest, the first quartile represents the 25th percentile, and the third quartile represents the 75th percentile

After presenting descriptive statistics, researchers often report correlations between variables. This allows readers to observe how variables are linearly related. The Pearson correlation ranges from −1 to +1 [6]. A value close to +1 indicates that two variables move together in a positive linear relationship. A value close to −1 indicates that as one variable increases, the other decreases in a linear relationship. A value close to zero means there is little to no linear relationship between the variables. However, correlation does not imply causality. Nonlinear relationships may still exist even when the correlation is close to zero.

We now manually calculate descriptive statistics using a dataset on temperature and electricity demand. A city energy planner wants to understand how weather affects electricity usage. The planner collects data on average daily temperature and electricity demand for five days. Table 1.2 presents the data.

Table 1.2. Temperature vs. Electricity Demand

Day	Temperature (°C)	Electricity Demand (MWh)
1	22	210
2	25	250
3	28	290
4	31	330
5	27	270

We compute the minimum, mean, maximum, and standard deviation for each variable. The minimum temperature is 22°C, and the minimum electricity demand is 210 MWh. The maximum temperature is 31°C, and the maximum electricity demand is 330 MWh.

We calculate the mean by summing all values and dividing by the number of observations, which is five. The bar notation indicates the mean.

$$\overline{temp} = \frac{22 + 25 + 28 + 31 + 27}{5} = 26.6$$

$$\overline{elec} = \frac{210 + 250 + 290 + 330 + 270}{5} = 270.0$$

The sample variance measures the average squared deviation from the mean. We calculate it by summing the squared deviations and dividing by the degrees of freedom, n – 1.

$$s_{temp}^2 = \frac{(22 - 26.6)^2 + (25 - 26.6)^2 + (28 - 26.6)^2 + (31 - 26.6)^2 + (27 - 26.6)^2}{5}$$

$$s_{temp}^2 = 11.3$$

$$s_{elec}^2 = \frac{(210 - 270)^2 + (250 - 270)^2 + (290 - 270)^2 + (330 - 270)^2 + (270 - 270)^2}{5}$$

$$s_{elec}^2 = 2{,}000$$

Variance is expressed in squared units, which can make interpretation difficult. For example, temperature is measured in degrees Celsius, but variance is measured in squared degrees. To return to the original units, we take the square root of the variance to obtain the standard deviation.

$$s_{temp} = \sqrt{s_{temp}^2} = \sqrt{11.3} = 3.4$$

$$s_{elec} = \sqrt{s_{elec}^2} = \sqrt{2000} = 44.7$$

Next, we calculate the Pearson correlation. We begin with the sample covariance, which measures how two variables vary together. The term "co" means together, and "variance" refers to variability. The covariance formula is below:

$$cov(x, y) = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{n - 1}$$

Covariance is closely related to variance. If we set x = y, then covariance becomes variance as shown below:

$$cov(x, x) = \frac{\sum(x_i - \bar{x})(x_i - \bar{x})}{n - 1} = \frac{\sum(x_i - \bar{x})^2}{n - 1} = var(x)$$

Covariance can be either positive or negative. To make it easier to interpret, we scale it by the standard deviations of both variables to obtain the correlation coefficient. The correlation (corr) is shown below:

$$corr(x, y) = \frac{cov(x, y)}{s_x \cdot s_y}$$

A convenient way to calculate variance and covariance manually is to organize the calculations in a table. Refer to Table 1.3.

Table 1.3. Variance and Covariance Calculations

Day	Temperature (°C)	Electricity Demand (MWh)	Dev. Temp	Dev. Electr	Cov
1	22	210	-4.6	-60	276
2	25	250	-1.6	-20	32
3	28	290	1.4	20	28
4	31	330	4.4	60	264
5	27	270	0.4	0	0
Average	26.6	270			
Variance			11.3	2000	
Cov					150

We multiply the deviations for temperature and electricity demand, sum the covariance terms, and divide by n − 1 = 4. This yields a covariance of 150. The Pearson correlation is calculated below:

$$corr(temp, elec) = \frac{150}{3.4 \cdot 44.7} = 0.998$$

Temperature and electricity demand are strongly positively correlated. As temperature increases, electricity demand also increases. This result is consistent with economic intuition: on hotter days, people use more air conditioning, which raises electricity consumption.

1.6 Normal Distribution

The normal distribution is the most important distribution for describing a continuous random variable [7]. Researchers use it widely in statistics, especially for making inferences about data. It appears in many real-world situations, including:

- Heights of people
- Test scores
- Amounts of rainfall
- Scientific measurements

In 1733, the French mathematician Abraham de Moivre published *The Doctrine of Chances*. In this work, he derived the normal distribution.

Another key idea in statistics is the Central Limit Theorem [8]. As the sample size increases, the distribution of the sample mean becomes approximately normal—no matter the original distribution. The probability density function is:

$$f(x_i) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{1}{2}\left[\frac{x_i - \mu}{\sigma}\right]^2\right)$$

The terms are defined as:

- μ is the population mean
- σ is the population standard deviation
- exp is Euler's number (≈ 2.71828)
- π ≈ 3.14159

Figure 1.4 shows the shape of the normal distribution, $f(x_i)$.

Inside the formula, the term in parentheses is called the z-score. It measures how far a value is from the mean in standard deviation units. It is defined on the next page:

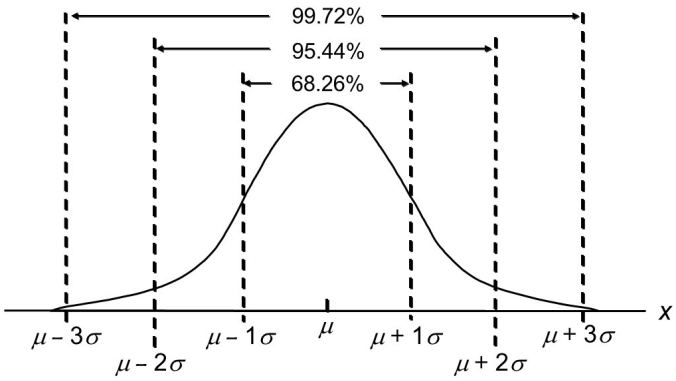


Figure 1.4. Normal Distribution Shape

$$z = \frac{x_i - \mu}{\sigma}$$

By converting values into z-scores, we transform any normal distribution into the standard normal distribution.

The key probabilities for a normal distribution:

- 68.26% of observations lie within μ ± σ
- 95.44% lie within μ ± 2σ
- 99.72% lie within μ ± 3σ

The normal distribution has several important characteristics:

- It is symmetric, i.e., the left side mirrors the right side
- Its skewness is zero
- The mean = median = mode
- Probabilities are represented as areas under the curve
- Total area under the curve equals 1

We often want to estimate a population mean using a sample. The sample mean is:

$$\bar{x} = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

If the data are normally distributed, then the sample mean is also normally distributed. The variability of the sample mean is measured by the standard error:

$$std.error = \frac{\sigma}{\sqrt{n}}$$

As the sample size n increases, the standard error decreases.

This means:

- Larger samples give more precise estimates
- The distribution becomes narrower

In Figure 1.5, notice that we graph $\bar{x}$, not x_i. As the sample size grows, the sample mean gets closer to the population mean.

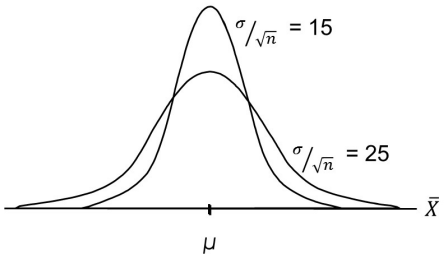


Figure 1.5. Standard Error by Sample Size

We often test whether a population mean equals a specific value. The null hypothesis always contains equality. There are three common forms in Figure 1.6.

H_o: $\mu \geq \mu_o$	H_o: $\mu = \mu_o$	H_o: $\mu \leq \mu_o$
H_a: $\mu < \mu_o$	H_a: $\mu \neq \mu_o$	H_o: $\mu > \mu_o$
Lower Tail	Two Tail	Upper Tail

Figure 1.6. Hypothesis Testing Cases

We apply these concepts to an example. A health report states that adults consume an average of 2.7 cups of coffee per day. A sample of 40 working professionals shows an average consumption of 3.1 cups per day. We assume the population standard deviation is 1.2 cups.

A researcher wants to test whether working professionals consume a different amount. We first state the hypotheses. This is a two-tail test because we do not specify whether coffee consumption for professionals is greater or smaller.

H_o: $\mu = 2.7$
H_a: $\mu \neq 2.7$

Then we calculate the test statistic. We calculate the z-score on the next page:

$$z = \frac{\bar{x} - \mu}{\sigma/\sqrt{n}} = \frac{3.1 - 2.7}{1.2/\sqrt{40}} = 2.1$$

We have two methods to determine whether this value is statistically significant. The first is to calculate the p-value. We use Excel's formula to calculate the p-value below:

=2*(1-NORM.DIST(2.1,0,1,1)) = 0.036

We have to remember that the formula calculated the probability from $-\infty$ to the z value. We get our probability by subtracting the result of NORM.DIST from 1. Then we multiply by two because we have two tails. The p-value equals 0.036. We choose a significant level of a = 0.05. Since a > p-value, we reject the null hypothesis. We conclude that working professionals drink on average a different amount of coffee.

We can use the critical z test to test the hypothesis. A significant level of 5% with two tails yields critical z scores of –1.96 and 1.96. If the calculated z is smaller than –1.96 or exceeds 1.96, we reject the null hypothesis. We get the same decision. We calculate the upper critical value using Excel below:

=NORM.INV(0.975,0,1) = 1.96

Remember, we have two tails and put half of α in each tail. The lower critical value is 0.025, and the upper is 0.975.

We can use the critical z-test to test the hypothesis. At a 5% significance level with two-tailed tests, the critical z scores are –1.96 and 1.96. If the calculated z is less than –1.96 or greater than 1.96, we reject the null hypothesis. We get the same decision.

We could switch this problem to a one-tailed test. We would do this to test whether professionals drink more coffee than the average adult.

We often want to compare two samples and determine whether they come from the same population or different populations. We modify our hypotheses in Figure 1.7.

H_0: $\mu_1 \geq \mu_2$	H_0: $\mu_1 = \mu_2$	H_0: $\mu_1 \leq \mu_2$
H_a: $\mu_1 < \mu_2$	H_a: $\mu_1 \neq \mu_2$	H_0: $\mu_1 > \mu_2$
Lower Tail	Two Tail	Upper Tail

Figure 1.7. Hypothesis Testing of Two Populations

We use another example. A retail analyst wants to test whether customers who shop online spend a different average per-purchase amount than customers who shop in-store. We

develop our hypotheses. We can rewrite our hypotheses with μ's on the left side and zero on the right.

$H_0: \mu_1 = \mu_2 \qquad \mu_1 - \mu_2 = 0$
$H_a: \mu_1 \neq \mu_2 \qquad \mu_1 - \mu_2 \neq 0$

We obtained the data in Table 1.4.

Table 1.4. Retail Sample Data

Sample 1 (Online)	Sample 2 (In-Store)
$n_1 = 45$	$n_2 = 55$
$\bar{x}_1 = 78.4$	$\bar{x}_2 = 75.1$
$\sigma_1 = 12.0$	$\sigma_2 = 10$

We also have two population standard deviations. Thus, we pool the standard deviations into one, which we show below:

$$\sigma_p = \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}} = \sqrt{\frac{12^2}{45} + \frac{10^2}{55}} = 2.240$$

Then we calculate the z statistic below:

$$z = \frac{\bar{x}_1 - \bar{x}_2 - (\mu_1 - \mu_2)}{\sigma_p} = \frac{78.4 - 75.1 - 0}{2.240} = 1.339$$

We calculate the p-value using the Excel function on the next page:

=2*(1-NORM.DIST(2.1,0,1,1)) = 0.181

The p-value equals 0.181. Since the p-value exceeds α, we fail to reject the null hypothesis. Thus, we conclude that online and in-store customers spend the same amount.

In Chapter 2, we extend these ideas to three or more groups using analysis of variance (ANOVA).

1.7 t Distribution

The analysis in Section 1.6 assumed that the population standard deviation is known. In practice, if we do not know the population mean, it is unlikely that we know the population standard deviation either. When the population standard deviation, σ, is unknown, we use the sample standard deviation, s, as an estimate. This situation is referred to as the "σ

unknown" case. Consequently, we replace the normal distribution with the t-distribution. We still assume that the population is normally distributed.

The t-distribution was developed by William Sealy Gosset, who published under the pseudonym "Student." He studied mathematics at Oxford University and later worked for the Guinness Brewery in Dublin. Gosset developed the t-distribution while conducting experiments on small samples related to materials and temperature [9]. For this reason, it is often called the Student's t-distribution.

The t-distribution is actually a family of distributions that depend on a parameter known as the degrees of freedom, denoted by df, where df = n − 1. The degrees of freedom represent the number of independent pieces of information used to calculate the sample standard deviation s.

As the degrees of freedom increase, the dispersion of the t-distribution decreases. In other words, a t-distribution with more degrees of freedom is more concentrated around the mean. As the degrees of freedom approach infinity, the t-distribution converges to the standard normal distribution.

The reason for this difference is that, in the t-case, we must estimate both the sample mean and the sample standard deviation. This introduces additional uncertainty compared to the normal distribution, where the standard deviation is assumed to be known. Because we estimate s, we lose one degree of freedom, which is why the denominator becomes n − 1.

Figure 1.8 illustrates two t-distributions along with the normal distribution. As the degrees of freedom increase, the spread of the t-distribution decreases and approaches that of the normal distribution.

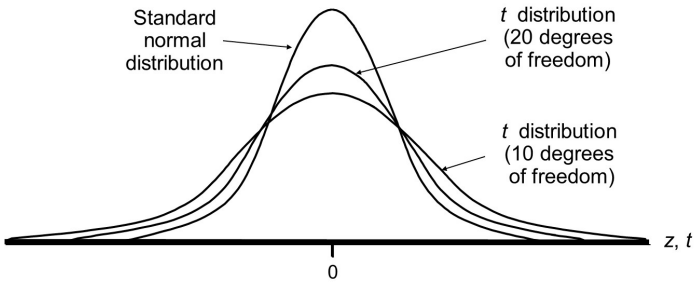


Figure 1.8. Normal vs. t Distribution

We now apply the t-distribution in an example. A company wants to compare the average lifetime of two brands of light bulbs:

Brand A is a new energy-efficient design
Brand B is the current standard product

The company wants to determine whether the new design lasts longer than the standard product. Table 1.5 summarizes the data.

Table 1.5. Bulb Lifetime Comparison

Brand A (Energy Efficient)	Brand B (Standard)
$n_1 = 30$	$n_2 = 35$
$\bar{x}_1 = 1{,}240$	$\bar{x}_2 = 1{,}180$
$s_1 = 120$	$s_2 = 150$

We first state the hypotheses. We test whether Brand A has a longer lifetime than Brand B:

Ho: $\mu_1 \leq \mu_2$
Ha: $\mu_1 > \mu_2$

Next, we calculate the standard error:

$$s_p = \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}} = \sqrt{\frac{120^2}{30} + \frac{150^2}{35}} = 33.509$$

At this point, we must consider whether the sample standard deviations are equal. Since we cannot assume equality, we use the Welch method. It allows for unequal variances. The degrees of freedom are calculated as:

$$df = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{1}{n_1 - 1}\left(\frac{s_1^2}{n_1}\right)^2 + \frac{1}{n_2 - 1}\left(\frac{s_2^2}{n_2}\right)^2} = \frac{\left(\frac{120^2}{30} + \frac{150^2}{35}\right)^2}{\frac{1}{29}\left(\frac{120^2}{30}\right)^2 + \frac{1}{34}\left(\frac{150^2}{35}\right)^2} = \frac{1{,}260.{,}808.16}{7{,}944.83 + 12{,}154.86} = 62.7$$

Many students are overwhelmed by the sheer size of this calculation. These calculations can appear overwhelming. To be conservative, we round down and use df = 62.

We then compute the t-statistic, which is similar to the z-statistic:

$$t = \frac{\bar{x}_1 - \bar{x}_2 - (\mu_1 - \mu_2)}{s_p} = \frac{1240 - 1180 - 0}{33.509} = 1.791$$

Next, we calculate the p-value using Excel. Because this is a one-tailed test, we use:

=T.DIST.RT(1.791,62) = 0.039

Using α = 0.05, we reject the null hypothesis because the p-value is less than α. Therefore, we have sufficient evidence to conclude that Brand A has a longer average lifetime than Brand B.

In practice, the population standard deviation is rarely known. For this reason, we typically default to using the t-distribution.

1.8 Chi-square Distribution

The chi-square distribution arises from sampling from a normally distributed population [10]. Specifically, when a random sample of size n is drawn from a normal population, the sampling distribution of:

$$\chi^2 = \frac{(n-1)s^2}{\sigma^2}$$

follows a chi-square distribution with df = n – 1 degrees of freedom. We use the chi-square distribution to construct interval estimates and conduct hypothesis tests about a population variance. The notation χ^2_α denotes the value of the chi-square distribution that leaves an area of α to the right.

The chi-square distribution is based on squaring a normally distributed variable. While the mean is analyzed using the normal or t-distribution, the chi-square distribution is used for variances because it involves squared values. As a result, the chi-square distribution is not symmetric. Consider the sample variance:

$$s^2 = \frac{\sum(x_i - \bar{x})^2}{n-1}$$

Squared values are always nonnegative, the chi-square distribution cannot take negative values. Figure 1.9 illustrates several chi-square distributions. Notice that the distribution is skewed to the right and becomes more symmetric as the degrees of freedom increase.

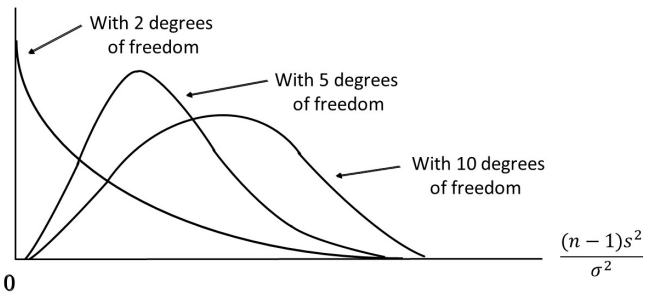


Figure 1.9. Chi-square Distributions

We now consider an example. A retail store is attempting to improve customer experience by making service times more consistent. Historically, the standard deviation of service time is σ = 8 minutes. After implementing a new training program, the manager collects a sample of 14 service times in minutes. The data are shown in Table 1.6.

Table 1.6. Service Time Data

42	38	35	40
37	36	39	41
34	38	36	37
35	39		

We begin by calculating the sample statistics. The sample mean is 37.64 minutes. The sample variance is 5.63, and the sample standard deviation is 2.37 minutes. Using α = 0.05, we construct a hypothesis test to determine whether the variability in service times has decreased. Because the goal is to reduce variability, this is a left-tailed test. We state the hypotheses in terms of variance:

H_0: $\sigma^2 \geq 64$
H_a: $\sigma^2 < 64$

Next, we compute the chi-square test statistic:

$$\chi^2 = \frac{(n-1)s^2}{\sigma^2} = \frac{(14-1)\cdot 5.63}{64} = 1.144$$

The degrees of freedom are: df = n − 1 = 13. We determine the critical value using Excel:

=CHISQ.INV(0.05,13) = 5.892

Because the calculated chi-square value is less than the critical value, we reject the null hypothesis. There is sufficient evidence to conclude that the variability in service times has decreased. The training program appears to have made service delivery more consistent.

1.9 F Distribution

The final distribution used in this book is the F-distribution. It is used to make inferences about two population variances [11]. For example, we may wish to compare variability in:

- product quality resulting from two different production processes
- temperatures from two heating devices
- assembly times for two different assembly methods

To conduct this analysis, we collect data from two independent random samples: one from population 1 and another from population 2. The sample variances form the basis for making inferences about the population variances.

In most cases, we conduct a one-tailed test using the F-distribution. If the test is originally formulated as a lower-tailed test, we can convert it into an upper-tailed test by switching the order of the variances. For example, suppose we want to compare the variances of two processes:

$$H_0: \sigma_1^2 \geq \sigma_2^2$$
$$H_a: \sigma_1^2 < \sigma_2^2$$

We can rewrite this as an equivalent upper-tailed test by reversing the variances:

$$H_0: \sigma_2^2 \leq \sigma_1^2$$
$$H_a: \sigma_2^2 > \sigma_1^2$$

This convention originates from earlier statistical tables, which typically provided only right-tail critical values.

We now illustrate the F-distribution with an example. A logistics company operates two delivery services: Standard Delivery and Express Delivery. Management is concerned about the consistency of delivery times, since high variability can lead to customer dissatisfaction.

The company collects delivery times (in minutes) from recent shipments:

Standard Delivery (Service 1): 30, 42, 35, 50, 43

Express Delivery (Service 2): 28, 32, 30, 35, 31

First, we calculate the sample variance for each service. The variance for Standard Delivery is 59.6, while the variance for Express Delivery is 6.7. We use a 10% significance level. Let s denote Standard Delivery and e denote Express Delivery. The hypotheses are:

$$H_0: \sigma_s^2 \leq \sigma_e^2$$
$$H_a: \sigma_s^2 > \sigma_e^2$$

Next, we compute the F-statistic. Because the variance for Standard Delivery is larger, we place it in the numerator to ensure an upper-tailed test that aligns with the hypotheses:

$$F = \frac{59.6}{6.7} = 8.90$$

The numerator degrees of freedom are: df = 5 – 1 = 4. The denominator degrees of freedom are: df = 5 – 1 = 4. We compute the p-value using Excel:

=F.DIST.RT(8.90,4,4) = 0.029

Because α > p-value, we reject the null hypothesis at the 10% significance level. There is sufficient evidence to conclude that the variability in delivery times is larger for standard delivery than express delivery.

1.10 Summary

In this chapter, we reviewed the core tools of business statistics. These ideas—descriptive statistics, probability distributions, hypothesis testing, and correlation—are the foundation for all statistical analysis. While many of these concepts may be familiar, mastering them is essential before moving forward.

The goal is not just to perform calculations, but to understand what the numbers mean. Each method helps us answer a simple question: What does the data tell us? By working through problems by hand, we build intuition and develop a deeper understanding of how these tools work.

In the next chapter, we take the next step by extending these ideas to compare three or more groups using analysis of variance (ANOVA). From there, we continue building toward multivariate methods, where we analyze multiple variables together and uncover more complex relationships in data.

A strong foundation makes everything that follows easier. Take the time to understand this material well—we will use it throughout the rest of the book.

1.11 Exercises

Problem 1.

An investor is deciding whether to diversify their portfolio by investing in two sectors: technology and energy. The annual returns (%) for two representative stocks over five years are given in Table 1.7:

Table 1.7. Energy vs. Tech Stocks

Year	Tech Stock (%)	Energy Stock (%)
1	10	4
2	18	12
3	-6	-3
4	20	8
5	9	7

a. Please compute the following for each stock: minimum, mean, maximum, and standard deviation.

b. Compute the correlation coefficient between the two stocks.

c. Based on your results:
- Which stock offers higher average returns?
- Which stock is riskier?
- Does diversification appear beneficial?

Problem 2.

A national survey reports that adults spend an average of 6.5 hours per day on screens. A researcher suspects that remote workers spend more time on screens than this average. A sample of 36 remote workers shows an average screen time of 7.1 hours per day. Assume the population standard deviation is 1.8 hours.

a. State the null and alternative hypotheses.

b. Compute the test statistic.

c. Compute the p-value.

d. At $\alpha = 0.05$, state your conclusion.

e. Repeat the test using the critical value approach.

Problem 3.

A technology company wants to compare the average battery life of two smartphone models.

Model A uses a new battery design

Model B uses the current standard battery

The company wants to know whether the mean battery life for the new design lasts longer than the standard battery. Table 1.8 contains the data.

Table 1.8. Battery Life Data

Model A (New Battery)	Model B (Standard Battery)
$n_1 = 36$	$n_2 = 49$
$\bar{x}_1 = 11.8$ hours	$\bar{x}_2 = 10.9$ hours
$\sigma_1 = 2.4$ hours	$\sigma_2 = 2.8$ hours

a. State the hypotheses

b. Compute the test statistic

c. Compute the p-value

d. What is our conclusion using $\alpha = 0.05$?

Problem 4.

A university wants to compare the average exam scores of two teaching methods.

Method A uses traditional lectures
Method B uses interactive learning

The university wants to determine whether the mean scores differ between the two methods. Summary statistics are in Table 1.9.

Table 1.9. Teaching Method Data

Method A (Traditional)	Method B (Interactive)
$n_1 = 25$	$n_2 = 30$
$\bar{x}_1 = 78.6$	$\bar{x}_2 = 74.2$
$s_1 = 10.5$	$s_2 = 9.8$

a. State the hypotheses

b. Compute the test statistic

c. Compute the p-value

d. What is our conclusion using $\alpha = 0.05$?

Problem 5.

A retail company is working to improve the consistency of its customer experience. Management believes that reducing variability in customer satisfaction is just as important as increasing the average score.

Customer satisfaction is measured on a 100-point scale. Based on past data, the company expects a population standard deviation of $\sigma = 10$.

After implementing a new employee training program, the company collects a sample of 15 customer satisfaction scores in Table 1.10.

Table 1.10. Consumer Satisfaction Scores

88	84	91	79	85
87	82	90	86	83
89	81	92	78	84

a. What is the sample mean customer satisfaction score?

b. What is the sample variance?

c. What is the sample standard deviation?

d. Conduct a hypothesis test to determine whether the variability has changed after the training program. Use a significance level of 5%.

- State the null and alternative hypotheses
- Compute the test statistic
- State the decision rule
- Provide your conclusion

Problem 6.

A manufacturing company operates two production lines that produce the same product. Management wants to ensure that both lines maintain consistent quality. In particular, they are concerned about variability in product thickness, since high variability can lead to defects.

To evaluate consistency, the company collects samples from each production line:

Production Line 1: $n_1 = 21$, $s_1^2 = 8.2$
Production Line 2: $n_2 = 26$, $s_2^2 = 3.0$

Management wants to test whether the variability of the two production lines is the same. The hypotheses are on the next page:

H_0: $\sigma_1^2 = \sigma_2^2$
H_a: $\sigma_1^2 \neq \sigma_2^2$

a. Use the p-value approach to test the hypotheses at the 5% significance level.

Hint: Place the larger sample variance in the numerator and conduct a right-tailed test. Remember to use half the α in the upper tail.

b. Repeat the test using the critical value approach.

Chapter 2. Analysis of Variance (ANOVA)

How can we tell whether differences across groups are real or simply due to chance? In many real-world settings, we must compare more than two groups at once—such as performance across firms, treatments in medicine, or outcomes across regions. In this chapter, we introduce the analysis of variance (ANOVA), a powerful method for determining whether three or more samples come from the same population distribution or whether at least one differs [11].

Once ANOVA detects a difference, we go a step further. Using Fisher's Least Significant Difference (LSD) method, we compare pairs of samples to identify exactly which groups have different population means. We then expand our analysis to more structured experiments by introducing block designs, where the same experimental units are used across treatments to control for additional sources of variation. We also examine randomized block designs with replication.

To build a deep understanding, we carry out all calculations by hand. This foundation will be essential when we encounter more complex ANOVA models in Chapter 7, allowing us to understand not just how to apply these methods, but why they work.

2.1 One-Way Analysis of Variance

In Chapter 1, we examined how to determine whether two samples have the same population mean. While a t-test is designed to compare two means, the analysis of variance (ANOVA) extends this idea to compare multiple means simultaneously [12]. This capability makes ANOVA especially useful in business, economics, and scientific research, where comparisons often involve more than two groups.

In this chapter, we introduce ANOVA, a statistical method used to compare the population means of three or more groups. ANOVA can be applied in both experimental and observational studies. In experimental studies, researchers actively control variables to investigate cause-and-effect relationships. In contrast, observational studies analyze data without direct control over the variables. In both settings, ANOVA helps determine whether observed differences across groups are statistically meaningful or simply due to random variation.

To understand ANOVA, it is important to define several key terms. A factor is a variable of interest that we want to study. A treatment is a specific level or category of that factor. For example, suppose we test three types of car wax. The factor is the type of wax, and the treatments are Type 1, Type 2, and Type 3.

The objects being studied are called experimental units. These could be individuals, firms, machines, or any entities on which measurements are taken. In a completely randomized design, treatments are randomly assigned to experimental units. Randomization helps ensure that differences across groups are due to the treatments rather than other factors. For example, assigning different types of car wax randomly to cars constitutes an experiment.

The main objective of ANOVA is to test whether group means are equal. Specifically, the null hypothesis states that all k population means are the same. The alternative hypothesis states that at least one mean differs [4]:

H_o: $\mu_1 = \mu_2 = \ldots = \mu_k$
H_a: At least one population mean differs from the others

If we reject the null hypothesis, we conclude that at least one group mean differs from the others. However, this result does not indicate which specific groups are different.

Figure 2.1 illustrates three sample means drawn from the same population. Random variation causes the sample means to differ slightly from one another. In this case, the null hypothesis is true, and the differences among the sample means are not statistically significant.

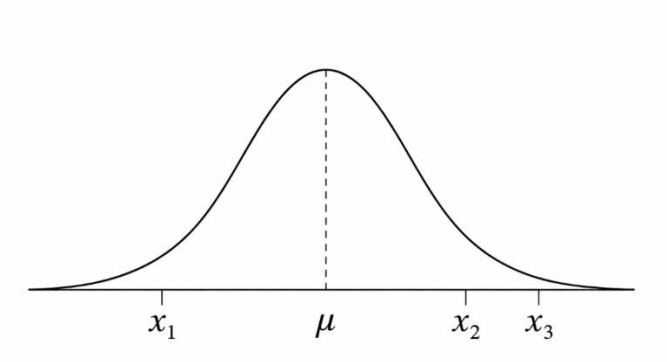


Figure 2.1. Sample Means from One Population

Figure 2.2 shows sample means drawn from different populations. In this case, we would reject the null hypothesis. Each sample mean is close to its own population mean but far from the others.

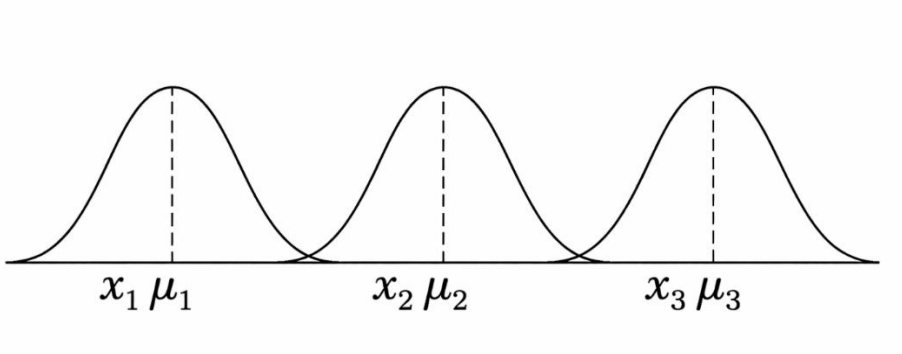


Figure 2.2. Sample Means from Different Populations

Key Insight

ANOVA tells us whether differences exist among group means, but it does not identify where those differences occur. To determine which specific means differ, we must conduct additional tests, such as post hoc comparisons.

The analysis begins by estimating the total variation in the data. Let $x_{i,j}$ denote the i-th observation in group j, and let $\bar{\bar{x}}$ denote the overall mean of all observations. The total variation, called the sum of squares total (SST), is:

$$SST = (n-1) \cdot var(x_{i,j}) = (n-1) \cdot \frac{\sum (x_{i,j} - \bar{\bar{x}})^2}{n-1} = \sum (x_{i,j} - \bar{\bar{x}})^2$$

Next, we measure variation between groups. Let $\bar{x_j}$ denote the mean of group j, and let n_j denote the number of observations in that group. The sum of squares due to treatments (SSTR) is:

$$SSTR = \sum n_j \cdot (\bar{x_j} - \bar{\bar{x}})^2$$

We then measure the variation within groups, which represents random noise. This is called the sum of squares due to error (SSE). Let s_j^2 denote the sample variance of group j. Then SSE equals:

$$SSE = \sum (n_j - 1) \cdot s_j^2$$

Although we do not show it here, ANOVA can be expressed as a linear model. As a result, the total variation can be decomposed into explained and unexplained components:

$$SST = SSTR + SSE$$

This decomposition shows that total variation in the response variable can be divided into variation explained by the factor and unexplained random variation [13].

We summarize these quantities in an ANOVA table. Table 2.1 shows an ANOVA table. ANOVA compares two types of variation:

1. Variation between groups (SSTR), due to the factor
2. Variation within groups (SSE), due to random noise

Table 2.1. Analysis of Variance Summary

ANOVA

Source	df	SS	MS	F	p-value
SSTR	$k - 1$	SSTR	$MSTR = \frac{SSTR}{k-1}$	$F = \frac{MSTR}{MSE}$	
SSE	$n - k$	SSE	$MSE = \frac{SSE}{n-k}$		
SST	$n - 1$	SST			

The mean squares (MS) are calculated by dividing each sum of squares by its corresponding degrees of freedom. These quantities are proportional to chi-square distributions. In practice, we typically do not compute the mean square for SST, as it simply yields the overall variance.

If group means differ substantially, the between-group variation (SSTR) will be large. If differences are primarily due to randomness, the within-group variation (SSE) will dominate. The F-statistic is computed as the ratio of the treatment mean square to the error mean square:

$$F = \frac{MSTR}{MSE}$$

This statistic compares variation between groups to variation within groups [12]. We typically use software, such as Excel, to compute the p-value associated with the F-statistic.

If the null hypothesis is true, the F-statistic should be close to 1, indicating that between-group variation is similar to within-group variation. A large F-statistic suggests that group differences are too large to be explained by random variation alone.

We reject the null hypothesis when the F-statistic is sufficiently large or, equivalently, when the p-value is small.

Figure 2.3 illustrates the rejection region of the F-distribution. Given a significance level α, we reject the null hypothesis if the p-value is less than α.

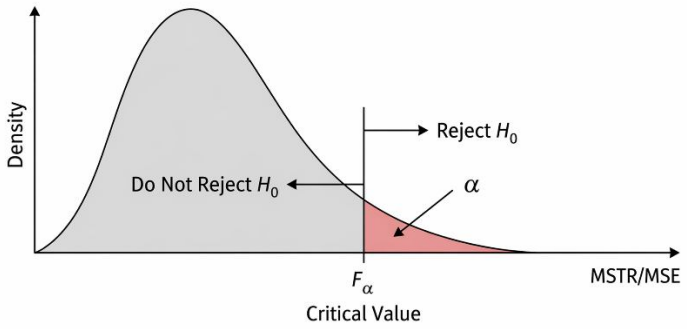


Figure 2.3. F-distribution Rejection Region

ANOVA relies on three key assumptions:

1. The data in each group are approximately normally distributed
2. The population variances are equal across groups
3. Observations are independent

These assumptions ensure that the F-test produces reliable results [3]. In practice, ANOVA is fairly robust, especially with larger sample sizes.

In statistical analysis, robustness refers to whether our conclusions remain valid when assumptions are slightly violated or when conditions in the experiment change. In other words, a result is robust if it is not driven by random chance, unusual observations, or uncontrolled differences among experimental units.

Key Insight

The F-test asks a simple question: Is the variation between groups large relative to the variation within groups? If so, the evidence suggests that the sample means come from different populations.

2.2 Case Study: Magic Wax

Magic Wax tested three types of car wax to compare their durability: Type 1, Type 2, and Type 3. Each wax was applied to five cars. Durability was measured by the number of washes each wax could withstand before showing signs of deterioration.

We test whether the waxes are equally durable or whether at least one type differs from the others. The hypotheses are:

H_0: $\mu_1 = \mu_2 = \mu_3$
H_a: At least one population mean differs from the others

We first calculate the sample mean and sample variance for each wax type, as shown in Table.2.2. Next, we compute the average across all observations, known as the grand mean. In this case, the grand mean equals 30.

Table 2.2. Magic Wax Experiment Data

Car	Wax Type 1	Wax Type 2	Wax Type 3
1	28	32	30
2	29	28	27
3	30	32	31
4	28	29	31
5	32	31	32
Sample Mean	29.4	30.4	30.2
Sample Variance	2.8	3.3	3.7

We also calculate the sample variance for all observations combined, which equals 3. To compute the sum of squares total (SST), we multiply this variance by the total degrees of freedom:

$$SST = 3 \times 14 = 42$$

Next, we measure the variation within groups, often referred to as noise. This is called the sum of squares error (SSE). We calculate SSE by multiplying each group's sample variance by its degrees of freedom. Then we sum across groups:

$$SSE = 2.8 \times 4 + 3.3 \times 4 + 3.7 \times 4 = 39.20$$

We then calculate the variation between groups, known as the sum of squares for treatments (SSTR). This measures how far each group mean is from the grand mean. For each group, we square the difference between the sample mean and the grand mean, multiply by the number of observations (5 in this case). Then we sum the results:

$$SSTR = 5\,(29.4 - 30)^2 + 5\,(30.4 - 30)^2 + 5\,(30.2 - 30)^2 = 2.80$$

Thus, we decompose the total variation (SST) into two components: variation due to treatments (SSTR) and random variation or noise (SSE). These results are summarized in Table 2.3.

Table 2.3. Analysis of Variance for Magic Wax

ANOVA

Source	df	SS	MS	F	p-value
SSTR	2	2.800	1.400	0.43	0.661
SSE	12	39.200	3.267		
SST	14	42.00			

There are three treatments, so the degrees of freedom for SSTR equal k – 1 = 2. The total degrees of freedom equal n – 1 = 14. Thus, the remaining degrees of freedom are assigned to SSE:

$$df_{SSE} = 14 - 2 = 12$$

The mean square (MS) is calculated by dividing each sum of squares by its corresponding degrees of freedom. The MS is distributed as a chi-square. Note that we typically do not compute the mean square for SST, because dividing SST by its degrees of freedom simply gives the overall sample variance.

The F-statistic is calculated as the ratio of the mean square for treatments to the mean square for error:

$$F = \frac{MS_{SSTR}}{MS_{SSE}} = \frac{1.400}{3.267} = 0.43$$

We compute the p-value using Excel with the function =F.DIST.RT(0.43, 2, 12). Python or other statistical software can also be used to obtain p-values and critical values. In ANOVA, the F-statistic is always written as a ratio, with the numerator degrees of freedom listed first.

Finally, we choose a significance level of α=0.05. Since the p-value (0.661) is greater than α, we fail to reject the null hypothesis.

Thus, there is no statistical evidence that the durability differs across the three wax types. From a practical perspective, this suggests that the company may choose the least expensive wax formulation for production, since all perform similarly.

2.3 Case Study: Lazer Manufacturing

Lazer Manufacturing wants to determine whether managers at its three plants work different numbers of hours per week. The plants are located in Chicago, St. Louis, and Detroit.

Because managers cannot be randomly assigned to different cities, this is an observational study. We take the data as given and analyze whether systematic differences exist across locations.

To investigate this question, the company selects a simple random sample of five managers from each plant. For each manager, we record the number of hours worked during the previous week. The results are summarized in Table 2.4. We calculated the average and sample variance for each plant.

Table 2.4. Lazer Manufacturing Study Data

Observation	Plant 1 Chicago	Plant 2 St. Louis	Plant 3 Detroit
1	49	74	50
2	53	62	64
3	58	67	60
4	62	74	56
5	53	63	55
Sample Mean	55	68	57
Sample Variance	25.5	33.5	28

We summarize the key elements of this observational study:

- **Factor:** Manufacturing plant
- **Treatments:** Chicago, St. Louis, Detroit
- **Experimental units:** Managers
- **Response variable:** Number of hours worked per week

We now formulate the hypotheses:

H_0: $\mu_1 = \mu_2 = \mu_3$
H_a: At least one population mean differs from the others

where:

$_1$: Mean weekly hours worked by managers in Chicago
$_2$: Mean weekly hours worked by managers in St. Louis
$_3$: Mean weekly hours worked by managers in Detroit

We conduct the test α at the 5% significance level.

Step 1: Compute Summary Measures

We calculate the overall (grand) mean across all observations. It equals 60. Next, we compute the three components of variation. Then we calculate the degrees of freedom for each variation source.

Step 2: Total Variation (SST). This measures the total variability in hours worked across all managers.

$$SST = 59.8571 \times 14 = 838.0$$

Step 3: Within-Group Variation (SSE). This represents the variation within each plant, which we interpret as random noise.

$$SSE = 25.5 \times 4 + 33.5 \times 4 + 28 \times 4 = 348.0$$

Step 4: Between-Group Variation (SSTR). This measures how much the plant means differ from the overall mean.

$$SSTR = 5\,(55-60)^2 + 5\,(68-60)^2 + 5\,(57-60)^2 = 2.80$$

Step 5: ANOVA Table

We now assemble the ANOVA table. The degrees of freedom are:

Between groups: $k-1 = 3-1 = 2$
Within groups: $n-k = 15-3 = 12$
Total: $n-1 = 15-1 = 14$

We insert the degrees of freedom (df) and the sum of squares (SS) into the ANOVA Table 2.5. Then we calculate the mean square (MS) by dividing the SS by its respective df. The F statistic is the ratio of MSTR and MSE. We calculate the p-value using Excel, =F.DIST.RT(8.45, 2, 12)

Table 2.5. ANOVA for Lazer Manufacturing

ANOVA

Source	df	SS	MS	F	p-value
SSTR	2	490.00	245.00	8.45	0.005
SSE	12	348.00	29.00		
SST	14	838.00			

The F-statistic is large, and the p-value is small. Because:

$$\text{p-value}=0.005 \quad < \quad \alpha=0.05$$

Thus, we reject the null hypothesis. This result provides strong evidence that at least one plant has a different average number of hours worked by managers.

Important Interpretation

Although ANOVA tells us that differences exist, it does not identify which specific plants differ from one another. To determine this, we must perform additional procedures, such as post hoc tests (e.g., Least Significance Differences).

Key Insight

Even in an observational study—where we cannot control assignments—ANOVA allows us to detect meaningful differences across groups. However, we must be cautious when interpreting causality, since other factors, such as local work culture or management practices may explain the differences.

2.4 Multiple Comparisons

The F-test tells us whether at least one sample mean differs from the others. However, it does not indicate which specific groups are different. After rejecting the null hypothesis, we proceed with multiple comparison procedures to identify where those differences occur.

One common method is Fisher's Least Significant Difference (LSD) test, which performs pairwise comparisons between group means using a t-statistic [14].

We compare all pairs of means, for all $i \neq j$. We write the hypotheses in both ways.

$$H_o: \mu_i = \mu_j \quad \text{or} \quad \mu_i - \mu_j = 0$$
$$H_a: \mu_i \neq \mu_j \quad \text{or} \quad \mu_i - \mu_j \neq 0$$

From Chapter 1, we developed the t-test for comparing two means. The LSD test applies the same idea, but uses the mean square error (MSE) from the ANOVA. It is the estimate of the common variance. It is also the noise in the system.

The test statistic is:

$$t = \frac{\bar{x}_i - \bar{x}_j - 0}{\sqrt{MSE\left(\frac{1}{n_i} + \frac{1}{n_j}\right)}}$$

When all groups have the same sample size ($n_i = n_j = n$), this simplifies to:

$$t = \frac{\bar{x}_i - \bar{x}_j - 0}{\sqrt{\frac{2 \cdot MSE}{n_i}}}$$

Using the data from the previous section, we compute the absolute differences between all pairs of sample means for Lazer Manufacturing:

$$| \bar{x}_1 - \bar{x}_2 | = | 55 - 68 | = 13.0$$

$$| \bar{x}_1 - \bar{x}_3 | = | 55 - 57 | = 2.0$$

$$| \bar{x}_2 - \bar{x}_2 | = | 68 - 57 | = 11.0$$

Because this is a two-tailed test, we compare the absolute values of the differences. We obtain the critical t-value using the degrees of freedom from the MSE (which reflects within-group variation). With $\alpha = 0.05$ and df = 12, we use Excel: =T.INV(0.975, 12). The critical value is:

$$t_{\alpha/2} = 2.179$$

We then compute the Least Significant Difference (LSD):

$$LSD = t_{\alpha/2}\sqrt{\frac{2 \cdot MSE}{n_i}} = 2.179\sqrt{\frac{2 \cdot 29.00}{5}} = 7.421$$

Then we put everything into Table 2.6

We compare the absolute difference of each pair to the LSD:

1. If the difference exceeds the LSD → reject H_0
2. If the difference is less than the LSD → fail to reject H_0

Table 2.6. LSD Results for Lazer Manufacturing

Least Significant Difference (LSD)

Pairs	Difference	t critical	LSD	Decision
Chicago-St. Louis	13.0	2.179	7.421	Reject
Chicago-Detroit	2.0	2.179	7.421	Fail to reject
St. Louis-Detroit	11.0	2.179	7.421	Reject

From the results:

Chicago vs. St. Louis:	significantly different
St. Louis vs. Detroit:	significantly different
Chicago vs. Detroit:	not significantly different

Thus, managers in St. Louis appear to work significantly different hours compared to the other two plants. Chicago and Detroit are statistically similar.

When we perform many pairwise comparisons, the probability of making a Type I error increases. A Type I error occurs when we incorrectly reject a true null hypothesis. The number of pairwise comparisons for k groups is:

$$m = \frac{k!}{2!\,(k-2)!}$$

For this example with k = 3:

$$m = \frac{k!}{2!\,(k-2)!} = \frac{3!}{2!\,(3-2)!} = \frac{3 \cdot 2 \cdot 1}{2 \cdot 1 \cdot 1} = 3$$

The probability of making at least one Type I error across all comparisons is called the experimental (or familywise) error rate [13]:

$$\alpha_{EW} = 1 - (1-\alpha)^m$$

For this example:

$$\alpha_{EW} = 1 - (1-\alpha)^m = 1 - (1-0.05)^3 = 0.14$$

Thus, the overall probability of making at least one Type I error is 14%, which is relatively high.

The problem becomes more severe as the number of treatments increases. For example, with k = 10 treatments:

$$m = \frac{k!}{2!\,(k-2)!} = \frac{10!}{2!\,(10-2)!} = \frac{10 \cdot 9 \cdot 8!}{2 \cdot 1 \cdot 8!} = 45$$

Using α = 0.05, the experimental error rate becomes approximately 90%, which is extremely high. This means there is a very high chance of falsely detecting at least one significant difference.

$$\alpha_{EW} = 1 - (1-\alpha)^m = 1 - (1-0.05)^{45} = 0.90$$

In more advanced courses, we introduce alternative methods, such as Bonferroni or Tukey procedures that control this error rate.

Key Insight

We use the F-test to determine whether at least one population mean differs from the others. Then, we use Fisher's Least Significant Difference (LSD) test to identify which specific pairs of means are statistically different. However, as the number of comparisons increases, so does the risk of Type I error—so results must be interpreted with caution.

2.5 Randomized Block Design

In many experiments, the experimental units are not identical. They may differ in ways that influence the response variable. A randomized block design addresses this issue by grouping similar units into blocks [12]. Within each block, all treatments are applied.

Blocking works by comparing treatments within similar groups rather than across all experimental units at once. In this example, automobiles differ in fuel efficiency due to factors such as engine condition, weight, or maintenance history. If we ignored these differences, they would inflate the variability in the data and make it harder to detect differences among gasoline blends.

By grouping similar observations into blocks and applying all treatments within each block, we effectively "hold constant" the characteristics of each automobile. As a result, the comparisons among treatments are made under more controlled conditions. This removes unwanted variation from the error term and allows the analysis to focus more clearly on the treatment effects.

In simple terms, blocking makes the experiment fairer: instead of comparing different gasoline blends across different cars, we compare them within the same car. This leads to more precise estimates and more powerful statistical tests.

By accounting for differences across blocks, we remove a source of variability that is not of primary interest. As a result, we can focus more clearly on the treatment effects. This design increases the precision of our estimates and reduces noise in the analysis.

For example, New Oil Company has developed three new gasoline blends and must decide which blend (or blends) to produce and distribute. To evaluate performance, we measure kilometers per liter (kpl) for each blend.

Our goal is to determine whether the average fuel efficiency is the same across the three blends. We state the hypotheses as:

H_0: $\mu_1 = \mu_2 = \mu_3$
H_a: At least one population mean differs from the others

We choose a significance level of $\alpha = 0.05$. Five automobiles are used in the experiment. Each automobile is tested with all three gasoline blends, making automobiles the natural blocks. The results are shown in Table 2.7.

Table 2.7. New Oil Company Data

Automobile	Gasoline Type (Treatment)			
Block	Blend X	Blend Y	Blend Z	Block Mean
1	13.2	12.8	12.8	12.90
2	12.8	12.3	12.3	12.47
3	12.3	12.3	11.9	12.19
4	14.0	13.2	12.3	13.18
5	11.1	10.6	11.1	10.91
Treatment Mean	12.7	12.2	12.1	12.33

We define the key elements of the experiment:

- **Factor:** Gasoline blend
- **Treatments:** Blend X, Blend Y, Blend Z
- **Blocks:** Automobiles
- **Response variable:** Kilometers per liter

In Table 2.7, we added a block mean column by averaging across each row (i.e., for each automobile). In contrast, the treatment means are computed by averaging down each column (i.e., for each gasoline blend).

In a randomized block design, the total variation is divided into three components:

Treatment variation (SSTR): Differences among gasoline blends

Block variation (SSB):	Differences among automobiles
Error variation (SSE):	Remaining unexplained variation

The degrees of freedom (df) are:

SSTR:	$k - 1$
SSB:	$b - 1$
SSE:	$(k - 1)(b - 1)$
SST:	$n - 1$

For this experiment:

SSTR:	$k - 1 = 3 - 1 = 2$
SSB:	$b - 1 = 5 - 1 = 4$
SSE:	$(k - 1)(b - 1) = 2\ 4 = 8$
SST:	$n - 1 = 15 - 1 = 14$

Step 1: The overall (grand) mean across all observations is 12.33.

Step 2: SST measures the total variability in kilometers per liter across all observations:

$$SST = 0.8 \times 14 = 11.21$$

Step 3: SSTR measures how much the treatment means differ from the grand mean:

$$SSTR = 5\ (12.7-12.33)^2+5\ (12.2-12.33)^2+5\ (12.1-12.33)^2 = 0.94$$

Step 4: SSB captures variability due to differences among automobiles:

$$SSB = 3(12.90-12.33)^2+3(12.47-12.33)^2+3(12.19-12.33)^2+3(13.18-12.33)^2+3\ (10.91-12.33)$$

$$SSB = 9.28$$

Step 5: SSE is the remaining unexplained variation:

$$SSE = SST - SSTR - SSB = 11.21 - 0.94 - 9.28 = 0.98$$

Step 5: We set up the ANOVA in Table 2.8. We calculate the mean square (MS) for each source by dividing by the respective degrees of freedom.

Table 2.8. ANOVA for New Oil Company

ANOVA

Source	df	SS	MS	F	p-value
SSTR	2	0.965	0.483	3.76	0.071
SSB	4	9.180	2.295	17.86	0.0
SSE	8	1.028	0.129		
SST	14	11.17			

The F-statistic for treatments is:

$$F = \frac{MS_{SSTR}}{MS_{SSE}} = \frac{0.470}{0.124} = 3.76$$

Using Excel:

=F.DIST.RT(3.76, 2, 8)

We obtain a p-value of 0.069. Since the p-value exceeds α = 0.05, we fail to reject the null hypothesis. There is insufficient evidence to conclude that gasoline blend affects fuel efficiency.

However, if we had chosen α = 0.10, we would reject the null hypothesis. This highlights how conclusions can depend on the chosen significance level.

Key Insight

The sum of squares due to error (SSE) represents everything that remains unexplained after accounting for treatments and blocks. In other words: SSE is the leftover variation after removing all systematic sources of variation.

A key advantage of the randomized block design is that it removes block-related variability from the error term. This typically reduces SSE and increases the sensitivity of the test, making it easier to detect meaningful treatment differences.

2.6 Factorial Design

In many experiments, we are interested in studying more than one factor at the same time. Factorial experiments and their corresponding ANOVA methods are especially useful when we want to draw conclusions about two or more variables simultaneously.

The term factorial refers to the fact that the experiment includes all possible combinations of the factor levels. For example, if Factor A has a levels and Factor B has b levels, then the experiment contains a × b treatment combinations.

In this design:

- Factor A is typically represented across the columns
- Factor B is represented across the rows
- We also examine the interaction between Factors A and B

A key advantage of factorial experiments is that they allow us to study not only the individual effects of each factor, but also how the factors work together [12].

For example, PowerClean Company studies how cleaning performance depends on the following factors:

Factor A (Detergent):
A_1: Standard
A_2: Premium
Factor B (Water Temperature):
B_1: Cold
B_2: Warm
B_3: Hot

The response variable is a cleaning score, where higher values indicate better performance. Each treatment combination is tested twice, providing replication for estimating experimental error. The data is shown on Table 2.9.

Table 2.9. PowerClean Data

	Temperature			
Detergent	Cold	Warm	Hot	Average A
Standard	65.00	72.00	78.00	72.67
Standard	67.00	74.00	80.00	-
Premium	66.00	75.00	88.00	77.50
Premium	69.00	77.00	90.00	-
Average B	66.75	74.50	84.00	75.08

Step 1: We calculate the grand mean for all observations. It equals 75.08.

Step 2: We calculate the total sum of squares (SST) from the sample variance of all observations.

$$SST = (n-1) \cdot var\left(x_{i,j}\right) = 65.72(12-1) = 722.9$$

Step 3: Each detergent has six observations. The sum of squares for Factor A is:

$$SSA=6\ (72.67-75.08)^2+6\ (77.50-75.08)^2= 70.083$$

Step 4: Each temperature has four observations. The sum of squares for Factor B is:

$$SSB=4\ (66.75-75.08)^2+4\ (74.50-75.08)^2+4\ (84.00-75.08)^2= 597.17$$

Step 5: The interaction measures whether the effect of one factor depends on the level of the other factor. For each cell, we compute:

$$\text{interaction} = (\text{cell mean} - \text{row mean} - \text{column mean} + \text{grand mean})^2$$

For example, we calculate the interaction for the first cell (Standard, Cold):

$$\text{interaction} = (\ ½\ (65.00 + 67.00) - 72.67 - 66.75 + 75.08)^2 = 5.5556$$

We repeat this for all cells. We show all calculations in Table 2.10.

Table 2.10. Interaction Term Calculation

	Cold	Warm	Hot
Standard	5.5556	1.681	13.347
Premium	5.5556	1.681	13.347

The SS due to the interaction equals the sum of all numbers in Table 2.10. The SSAB equals 41.167.

Step 6: We calculate the sum of squares dues to error as below:

$$SSE = SST-SSA-SSB-SSAB = 722.9-70.1-597.2-41.2 = 14.5$$

Step 7: We calculate the degrees of freedom for all sources:

Factor A: $2-1=1$
Factor B: $3-1=2$
Interaction: $(2-1)(3-1)=2$
Error: $2\times3\times(2-1)=6$
Total: $12-1=11$

We insert this information into Table 2.11.

Table 2.11. ANOVA for PowerClean

ANOVA					
Source	df	SS	MS	F	p-value
SSA (Detergent)	1	70.1	70.1	29.0	0.0
SSB (Temperature)	2	597.2	298.6	123.6	0.0
SSAB (Interaction)	2	41.2	20.6	8.5	0.0
SSE	6	14.5	2.4		
SST	11	722.9			

At $\alpha = 0.05$, all F-statistics are large and the p-values are effectively zero. Therefore, we reject all null hypotheses. We conclude that detergent type affects cleaning performance. Water temperature also affects cleaning performance. There is a significant interaction between detergent and temperature.

The interaction effect reveals that the impact of detergent depends on water temperature. At cold temperatures, both detergents perform similarly. At hot temperatures, the premium detergent performs substantially better. In other words, the benefit of the premium detergent increases as temperature rises. This is the defining feature of an interaction: the effect of one factor changes depending on the level of another factor.

Key Insight

Factorial designs are powerful because they allow us to:

- Study multiple factors simultaneously
- Improve efficiency by combining experiments
- Detect interactions that would be missed in single-factor studies

Most importantly, factorial experiments reveal how variables work together, not just in isolation.

2.7 Summary

In this chapter, we introduced the analysis of variance (ANOVA)> it is a powerful tool for comparing three or more group means. Instead of performing multiple t-tests, ANOVA provides a single, coherent framework that evaluates whether observed differences across groups are meaningful or simply the result of random variation. By decomposing total variation into variation between groups and variation within groups, the F-test allows us to assess whether the differences we observe are statistically significant.

When ANOVA indicates that differences exist, the next step is to determine where those differences occur. We used Fisher's Least Significant Difference (LSD) method to conduct pairwise comparisons between group means. This process revealed an important insight: as the number of comparisons increases, so does the probability of making a Type I error. As a result, multiple comparison procedures must be applied carefully and interpreted with caution.

We also expanded our analysis to more structured experimental designs. Block designs allow us to control for additional sources of variation by using the same experimental units across treatments. Randomized block designs with replication further improve the precision of our estimates and strengthen our ability to isolate treatment effects.

Throughout the chapter, we emphasized calculating everything by hand. This approach is intentional. By working through each step, we develop a deeper understanding of how ANOVA measures and partitions variation, and how the F-test is constructed. This foundation will be essential as we move to more advanced models.

In Chapter 7, we will extend these ideas to more complex ANOVA frameworks. With the intuition and mechanics developed here, we will be prepared not only to apply these methods, but to understand the logic behind them.

Big Picture Takeaway

ANOVA answers a fundamental question: Are the differences we observe across groups real, or are they just random noise?

- The F-test tells us whether differences exist.
- Multiple comparison methods tell us where those differences occur.
- Experimental design, such as blocking, helps us measure those differences more accurately.

Together, these tools allow us to move from simple observation to meaningful statistical conclusions.

2.8 Exercises

Problem 1

A researcher wants to examine whether different study environments affect student performance. The experiment includes one factor:

Factor (Study Environment):

A1 : Quiet Room
A2 : Coffee Shop
A3 : Library

The response variable is test score. Each environment is tested with five students. Table 2.12 includes the average and sample variance for each study location. The grand mean equals 80.07.

Table 2.12. Study Environment Exam Scores

	Quiet Room	Coffee Shop	Library
	78	81	80
	82	79	81
	80	83	79
	79	80	82
	81	78	78
Sample Mean	80.0	80.2	80.0
Sample Variance	2.5	3.7	2.5

Please calculate the ANOVA and determine whether the study place affects a student's test scores. Please calculations into Table 2.13.

Table 2.13. ANOVA for Study Environment

ANOVA Source	df	SS	MS	F	p-value
SSTR					
SSE					
SST					

Problem 2

Four machines are tested. Five operators serve as blocks. Table 2.14 shows the data.

Table 2.14. Operators and Machines Data

Operator	M1	M2	M3	M4	Block Average
1	72	80	74	69	73.750
2	75	82	76	70	75.750
3	73	81	75	71	75.000
4	74	83	77	72	76.500
5	76	84	78	73	77.750
Treatment Average	74.000	82.000	76.000	71.000	

Treatment Variance	2.500	2.500	2.500	2.500
Treatment Count	5	5	5	5
Overall Average	75.750			
Overall Variance	19.145			
Overall Count	20			

a. Place calculations of an analysis of variance into Table 2.15. What is your conclusion using $\alpha = 0.1$?

Table 2.15. ANOVA for Operators and Machines

Analysis of Variance (ANOVA)					
Source	df	Sum of Squares	Mean SS	F-stat	p-value
Treatment					
Blocks					
Error					
Total					

b. At the **$\alpha = 0.1$** level of significance, use Fisher's LSD procedure to test for the equality of the means. Please fill in the missing numbers in Table 2.16.

Table 2.16. LSD Results for Machines

Pair	Diff	t-critical	LSD	Decision
M1-M2				
M1-M3				
M1-M4				
M2-M3				
M2-M4				
M3-M4				

c. What is the Type I error for the experiment?

Problem 3

A coffee shop chain wants to study how sales performance depends on both drink prices and store ambiance. The company believes that these two factors may jointly influence customer behavior.

The experiment includes the following factors:

Factor A (Price Level):
- A1: Low Price
- A2: High Price

Factor B (Store Ambiance):
- B1: Basic Interior
- B2: Modern Interior

The response variable is daily revenue in hundreds of dollars, where higher values indicate better performance. Each treatment combination is tested twice, providing replication for estimating experimental error. Table 2.17 shows the data. The grand mean equals 49.50.

Table 2.17. Coffee Price and Environment Data

Environment	Low Price	High Price	Average
Basic	42.00	38.00	40.25
Basic	45.00	36.00	
Modern	55.00	60.00	58.75
Modern	58.00	62.00	
Average	50.00	49.00	

(a) Please construct an ANOVA with an interaction term between price and environment.

Place the calculations of the interaction into Table 2.18:

Table 2.18. Interaction Term for Coffee Data

Environment	Low Price	High Price
Basic		
Modern		
		SSAB

Place calculations of the ANOVA into Table 2.19.

Table 2.19. ANOVA Coffee Data

ANOVA Source	df	SS	MS	F	p-value
SSA (Environment)					
SSB (Price)					
SSAB (Interaction)					
SSE					
SST					

Chapter 3. Simple Linear Regression

In this chapter, we introduce simple linear regression, one of the most powerful and widely used tools in statistics [3, 15]. The goal is straightforward: we study the relationship between one explanatory variable (x) and one outcome variable (y). By estimating a straight line with an intercept and slope, we can describe how changes in x are associated with changes in y.

Rather than treating regression as a black box, we build everything step by step. We manually calculate each component so that we can see exactly where the results come from and how they are connected. Using basic descriptive statistics, we estimate the slope and intercept of the regression line. This approach helps us understand not just what the model does, but why it works.

Next, we connect regression to ideas we already know from ANOVA. We construct an ANOVA table for the regression and use the F-statistic to determine whether the model provides a good overall fit to the data. In other words, we test whether x helps explain variation in y. We then use t-statistics to evaluate the individual parameter estimates. It allows us to determine whether the slope and intercept are statistically meaningful.

This framework parallels what we learned in experimental design. In ANOVA, the F-test tells us whether at least one population mean differs, while follow-up procedures identify where those differences occur. In regression, the F-test evaluates the overall model, and the t-tests examine the contribution of each parameter.

Finally, we study standardized residuals to assess the quality of the model and detect potential outliers. These diagnostics help us determine whether unusual observations are influencing the results and whether the model assumptions are reasonable.

By the end of this chapter, we will not only know how to estimate a regression model—we will understand how all its pieces fit together and how to interpret the results with confidence.

3.1 Overview

Regression analysis is one of the most important and widely used tools in statistics and econometrics. Managers, economists, and researchers often need to understand how one variable is related to another and how changes in one variable influence outcomes. Regression provides a formal and systematic way to model, quantify, and interpret these relationships .

In this chapter, we focus on simple linear regression, which examines the relationship between one independent variable (x) and one dependent variable (y). The goal is to describe this relationship using a straight line that captures the overall pattern in the data.

Although the model is simple, it is extremely powerful. Simple linear regression forms the foundation for more advanced methods involving multiple variables, nonlinear relationships, and more complex data structures. By mastering this framework, we will build the intuition and skills needed to understand and apply more advanced statistical models later on.

3.2 Model Specification

In regression analysis, the dependent variable (y) is the outcome we want to predict or explain. The independent variable (x) is used to explain changes in y. In simple linear regression, we model the relationship between these two variables using a straight line. The population regression model is written as:

$$y = \beta_0 + \beta_1 x + \varepsilon$$

where:

- β_0 is the intercept
- β_1 is the slope
- ε is the error term, which captures all other factors affecting y that are not included in the model [3]

Key Insight

The slope β_1 measures how much y changes, on average, when x increases by one unit.

In practice, we do not know the true parameters β_0 and β_1. Instead, we estimate them using sample data. The most common method is the least squares method, which selects the line that minimizes the total squared error [17].

The estimated regression equation is:

$$\hat{y} = b_0 + b_1 x$$

where:

- b_0 is the estimated intercept
- b_1 is the estimated slope
- $\hat{y}$ is the predicted value of y

The slope b_1 measures the change in y for a one-unit increase in x, while b_0 represents the predicted value of y when x = 0.

A professor wants to examine the relationship between how many hours a student studies and their exam score. We expect that more study time leads to higher exam scores.

y: Exam score
x: Hours studied

We show the data in Table 3.1.

Table 3.1. Study Hours and Exam Scores

Student	Hours Studied (x)	Exam Score (y)
1	2	65
2	4	67
3	6	82
4	8	85
5	10	84

A useful first step in regression analysis is to plot the data. A scatterplot allows us to visually assess the relationship between x and y. From Figure 3.1, we observe a positive relationship between hours studied and exam score: as study time increases, exam scores tend to increase as well. We then draw a straight line that best represents this pattern in the data.

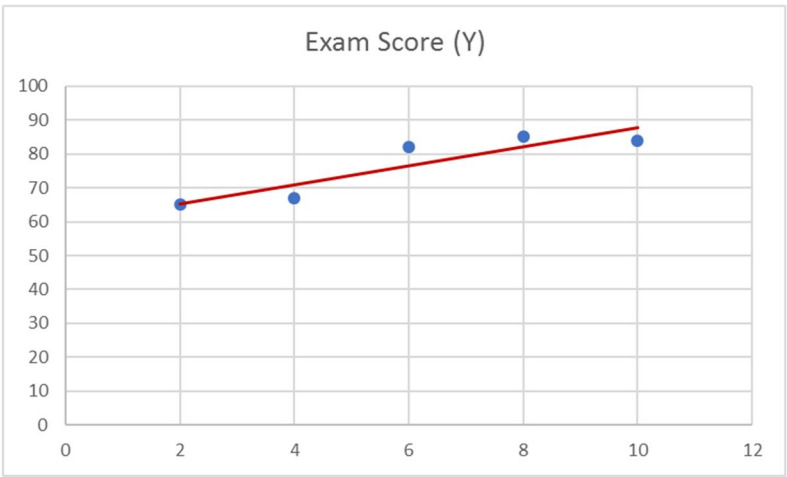


Figure 3.1. Study Hours vs. Exam Score

The goal of regression is to find the line that best fits the data. But what does "best" mean? For each observation, there is a difference between the actual value y_i and the predicted value $\hat{y}$. This difference is called the error or residual:

$$\varepsilon_i = y_i - \hat{y}_i$$

These errors can be positive or negative. If we simply added them together, they could cancel out. To avoid this problem, we square each error and then sum them across all observations. The objective function becomes:

$$f = \sum_i (y_i - \hat{y}_i)^2$$

The least squares method works by adjusting the values of b_0 (intercept) and b_1 (slope) to make this sum of squared errors as small as possible. In other words, we choose the line that makes the predicted values as close as possible to the observed data points. This process provides the "best-fitting" line in the sense that it minimizes the total squared distance between the data points and the regression line.

3.3 Parameter Estimation

This section is more advanced. Students who are primarily interested in applications may choose to skip this section. However, working through the derivation provides deeper insight into how the regression model is constructed.

We begin with the least squares objective function, f: The goal is to choose β_0 and β_1 to minimize this function.

$$f = \sum (y_i - \beta_0 - \beta_1 x_i)^2$$

We take the partial derivative with respect to β_0.

$$\frac{\partial f}{\partial \beta_0} = \sum 2(-1)(y_i - \beta_0 - \beta_1 x_i)$$

We set the derivative equal to zero: Then we replace the Greek parameters with their estimates: We are estimating values for population parameters that we do not know.

$$2(-1)\left(\sum y_i - \sum b_0 - b_1 \sum x_i\right) = 0$$

We apply the summation and break the terms apart. We solve for b_0.

$$b_0 \sum 1 = \sum y_i - b_1 \sum x_i$$

We sum over all observations. Since we have n observations, that would equal n by summing one n times.

$$b_0 \cdot n = \sum y_i - b_1 \sum x_i$$

We divide both sides by n.

$$b_0 = \frac{\sum y_i}{n} - b_1 \frac{\sum x_i}{n}$$

We replace with the average of the y and x variables.

$$b_0 = \bar{y} - b_1 \bar{x}$$

The only problem is we do not know the slope yet. We return to the same objective function again.

$$f = \sum (y_i - \beta_0 - \beta_1 x_i)^2$$

We repeat the process and take the partial derivative with respect to β_1.

$$\frac{\partial f}{\partial \beta_1} = \sum 2(-x_i)(y_i - \beta_0 - \beta_1 x_i) = 0$$

We factor out the -2. Then we can apply the summation over each term. We substitute the Latin letters for Greek.

$$-2\left(\sum y_i\, x_i - b_0 \sum x_i - b_1 \sum x_i^2\right) = 0$$

We substitute the b_0 into the equation.

$$-2\left(\sum y_i\, x_i - (\bar{y} - b_1 \bar{x}) \sum x_i - b_1 \sum x_i^2\right) = 0$$

We use the distributive rule to get rid of all parenthesis.

$$\sum y_i\, x_i - \bar{y} \sum x_i + b_1 \bar{x} \sum x_i - b_1 \sum x_i^2 = 0$$

We place the b_1 terms on the left and the other terms on the right.

$$b_1 \sum x_i^2 - b_1 \bar{x} \sum x_i = \sum y_i \, x_i - \bar{y} \sum x_i$$

We factor out the b_1 and solve for it.

$$b_1 = \frac{\sum y_i \, x_i - \bar{y} \sum x_i}{\sum x_i^2 - \bar{x} \sum x_i}$$

We do a trick. We multiply the numerator and denominator by n and $1/n$. This equals one, but we use this trick to simply the expression.

$$b_1 = \frac{\sum y_i \, x_i - n\bar{y} \frac{1}{n} \sum x_i}{\sum x_i^2 - n\bar{x} \frac{1}{n} \sum x_i}$$

We substitute the average x and y into the equation.

$$b_1 = \frac{\sum y_i \, x_i - n\bar{y}\bar{x}}{\sum x_i^2 - n\bar{x}\bar{x}}$$

We factor a n from both the numerator and denominator.

$$b_1 = \frac{n\left[\frac{1}{n}\sum(y_i x_i) - \bar{y}\bar{x}\right]}{n\left[\frac{1}{n}\sum(x_i^2) - \bar{x}^2\right]}$$

The n's cancel. Then we substitute the covariance in the numerator and variance of x in the denominator. Although these are the population covariance and variance, they would equal the sample equivalent. The population covariance and variance are divided by n, while the sample ones are divided by n – 1. These would cancel and yield the same answer.

$$b_1 = \frac{cov(x, y)}{var(x)}$$

Key Insight

The slope is the ratio of how x and y move together (covariance) to how x varies (variance). This result provides an important interpretation: If x and y move together strongly → large slope. If x varies a lot but has little relationship with y → small slope.

3.4 Case Study: Study Time vs. Exam Score

We now complete the example introduced in Section 3.2. Using the sample data, we compute the summary statistics shown in Table 3.2.

Table 3.2. Experiment Summary Statistics

Statistic	x_i	y_i
Average	6	76.6
Sample variance	10	95.3
Sample covariance	28	

We first calculate the slope using the covariance and variance:

$$b_1 = \frac{cov(x_i, y_i)}{var(x_i)} = \frac{28.0}{10.0} = 2.80$$

Next, we compute the intercept:

$$b_0 = \bar{y} - b_1\bar{x} = 76.6 - 2.80 \cdot 6.0 = 59.8$$

The estimated regression line is below. This equation describes the relationship between study time and exam score.

$$\hat{y} = 59.8 + 2.80\ x$$

We can use the regression equation to predict a student's exam score. For a given value of x, the model provides an estimate of the mean value of y. For example, if a student studies for 5 hours: The predicted grade is calculated below:

$$\hat{y} = 59.8 + 2.80\ (5) = 73.8$$

Thus, the predicted exam score is 73.8 points. We could construct confidence intervals around this prediction to account for more uncertainty.

The slope has an important interpretation. Starting from the population model:

$$y = \beta_0 + \beta_1 x + \varepsilon$$

We take the derivative with respect to x:

$$\frac{\partial y}{\partial x} = \beta_1$$

This shows that β_1 represents the marginal effect of x on y. In practice, we interpret the estimated slope b_1 as the marginal effect. For each additional hour of study, the student's exam score increases by approximately 2.80 points, holding all other factors constant.

Since our data are discrete, we can also interpret the slope as:

$$\frac{\Delta y}{\Delta x} \approx b_1$$

This means that a one-unit increase in study time (one additional hour) leads to an approximate increase of 2.80 points in the exam score.

Key Insight

The regression equation provides both a prediction tool (forecasting exam scores), and an interpretation tool (understanding how study time affects performance).

3.5 Analysis of Variance (ANOVA) and Hypothesis Tests

We now construct the ANOVA for the regression model [12]. We begin by calculating the sample variance of the dependent variable, y, and multiplying it by its degrees of freedom. With five observations, the total degrees of freedom equal n – 1 = 4. Thus,

$$\text{SST} = (n - 1)\,\text{var}(y_i) = 4\,(95.3) = 381.2$$

Next, we use the regression model to generate predicted values $\hat{y}$ for each observation. We compute the residuals (prediction errors), square them, and sum across all observations to obtain the sum of squared errors (SSE). This represents the total unexplained variation in the model. Table 3.3 summarizes these calculations. We switch the Greek letter ε with e because we have obtained actually measurements.

Table 3.3. Predicted Values and Errors

Student	x	y	$\hat{y}$	e	e^2
1	2	65	65.4	-0.4	0.16
2	4	67	71	-4	16
3	6	82	76.6	5.4	29.16
4	8	85	82.2	2.8	7.84
5	10	84	87.8	-3.8	14.44
				SSE	67.6

Thus, SSE=67.6. The degrees of freedom for the regression equal the number of slope coefficients. In this case, we have one slope, so df = 1. We will relax this restriction in Chapter 6. The sum of squares due to regression is:

$$SSR = SST - SSE = 381.2 - 67.6 = 313.6$$

We summarize the ANOVA results in Table 3.4.

Table 3.4. ANOVA for the Experiment

	ANOVA				
	df	SS	MS	F	p-value
SSR	1	313.6	313.6	13.9	0.03
SSE	3	67.6	22.5		
SST	4	381.2			

We compute each mean square by dividing the sum of squares by its corresponding degrees of freedom. The F-statistic is then calculated as the ratio of the mean square due to regression (MSR) to the mean square error (MSE). The p-value is calculated in Excel as:

$$=F.DIST.RT(13.9,1,3) = 0.03$$

We test the following hypotheses:

H_0: All slope coefficients equal zero
H_a: At least one slope coefficient is not zero.

Although the plural "slopes" may seem unusual in a simple regression, this formulation becomes important in multiple regression in Chapter 6. Here, we reject the null hypothesis and conclude that the slope is not equal to zero. Therefore, the regression model provides a meaningful fit to the data.

This procedure parallels the ANOVA framework introduced in Chapter 2. There, the F-test determines whether at least one population mean differs across groups. In regression, the F-test evaluates whether the model explains a significant portion of the variation in y. Afterward, we use t-tests to assess the statistical significance of individual coefficients. It is similar to least significant differences in Chapter 2.

To determine whether a meaningful relationship exists, we test whether the slope equals zero:

H_0: $\beta_1 = 0$
H_a: $\beta_1 \neq 0$

We may also test the statistical significance of the intercept, β_0. To conduct these tests, we first estimate the variance of the error term ε. The mean square error (MSE) provides an estimate of the error variance:

$$s^2 = MSE$$

Taking the square root yields the standard error (s) of the whole regression: It is the standard error of the noise of the system.

$$s = \sqrt{MSE}$$

We then use this quantity to compute the standard errors of the individual coefficients. The standard error of the slope is:

$$s_{b1} = \frac{s}{\sqrt{(v-1) \cdot var(x_i)}}$$

In our example,

$$se_{b1} = \frac{s}{\sqrt{(v-1) \cdot var(x_i)}} = \frac{\sqrt{22.533}}{\sqrt{4 \cdot 10}} = 0.751$$

The corresponding t-statistic is:

$$t = \frac{b_1 - \beta_1}{se_{b1}} = \frac{2.8 - 0}{0.751} = 3.73$$

We do not know the population slope. However, we test whether the evidence suggests it equals zero. In general, we could test whether a population parameter equals any specified value other than zero. Using Excel, the p-value is computed as:

=T.DIST.2T(3.73,3) = 0.034

The degrees of freedom come from the MSE, i.e., the residual degrees of freedom.

We can similarly compute the standard error of the intercept:

$$se_{b0} = s\sqrt{\frac{1}{n} + \frac{\bar{x}^2}{(v-1) \cdot var(x_i)}}$$

Substituting values,

$$se_{b0} = \sqrt{22.533}\sqrt{\frac{1}{5} + \frac{6^2}{4 \cdot 10}} = 4.979$$

The corresponding t-statistic is:

$$t = \frac{b_0 - 0}{se_{b0}} = \frac{59.8 - 0}{4.979} = 12.07$$

Statistical software typically reports coefficient estimates, standard errors, t-statistics, and p-values in a single table, as shown below in Table 3.5.

Table 3.5. Regression Estimates

Coefficient	Parameter Estimate	Std Error	t-stat	p-value
intercept	59.8	4.979	12.01	0.00
slope	2.8	0.751	3.73	0.03

Using a significance level of α = 5%, we reject the null hypotheses for both the intercept and slope because their p-values are less than α. Thus, both coefficients are statistically different from zero.

Key Insight

In simple regression, the F-statistic is equal to the square of the t-statistic for the slope. Consequently, both tests produce the same p-value. In this example,

$$F = 13.9 \quad \text{and} \quad t^2 = (3.73)^2 = 13.9$$

This equivalence arises because there is only one slope coefficient. In multiple regression, however, the F-test evaluates all slopes jointly, while t-tests assess each coefficient individually.

3.6 R^2 and Correlation

To evaluate how well the regression model fits the data, we use the coefficient of determination, denoted by R^2. This statistic measures the proportion of variation in the dependent variable that is explained by the regression model [16].

Using the ANOVA results, we compute R^2 as:

$$R^2 = \frac{SSR}{SST} = \frac{313.6}{381.2} = 0.82$$

The value of R^2 ranges from 0 to 1. A value close to 1 indicates that the model explains most of the variation in the data. A value close to 0 indicates weak explanatory power. In this example, the regression explains 82% of the variation in exam scores. It suggests that the nodel fits the data well.

The Pearson correlation coefficient measures the strength and direction of the linear relationship between two variables. Its value ranges from −1 to +1:

- A value close to +1 indicates a strong positive relationship
- A value close to −1 indicates a strong negative relationship
- A value near 0 indicates little or no linear relationship

In simple linear regression, there is a direct relationship between correlation and R^2. Specifically, the square of the Pearson correlation coefficient equals R^2. Therefore, we can recover the correlation by taking the square root of R^2 and assigning the appropriate sign based on the slope:

$$corr = sign(b_1)\sqrt{R^2}$$

In our example,

$$corr = sign(b_1)\sqrt{R^2} = sign(2.80)\sqrt{0.82} = +0.91$$

Thus, there is a strong positive linear relationship between study time and exam scores.

Although we could compute the correlation directly using the formula introduced in Chapter 1, simple linear regression provides a convenient alternative.

Key Insight

The coefficient of determination, R^2, summarizes how well the model fits the data. However, a high R^2 does not imply that the relationship is causal or correctly specified.

In simple linear regression, R^2 is directly related to the Pearson correlation: it is simply the square of the correlation coefficient with the slope's sign.

3.7 Residual Analysis

Regression analysis relies on several key assumptions [15]:

- The error term has a mean of zero.
- The variance of the error term is constant, i.e., homoscedasticity.
- Errors are independent.
- Errors are normally distributed.
- Errors are uncorrelated with the independent variable.

Violations of these assumptions can lead to unreliable statistical inference. In Chapter 6, we relax some of these assumptions and examine how they affect parameter estimation.

Residual analysis helps us evaluate whether these assumptions are satisfied. Residuals, e, are the differences between observed and predicted values:

$$e_i = y_i - \hat{y}_i.$$

A well-specified model produces residuals that appear randomly scattered around zero, with no clear pattern. Systematic patterns in the residuals may indicate problems such as nonlinearity, heteroscedasticity, or omitted variables.

Figure 3.2 shows the residual plot for this chapter's example.

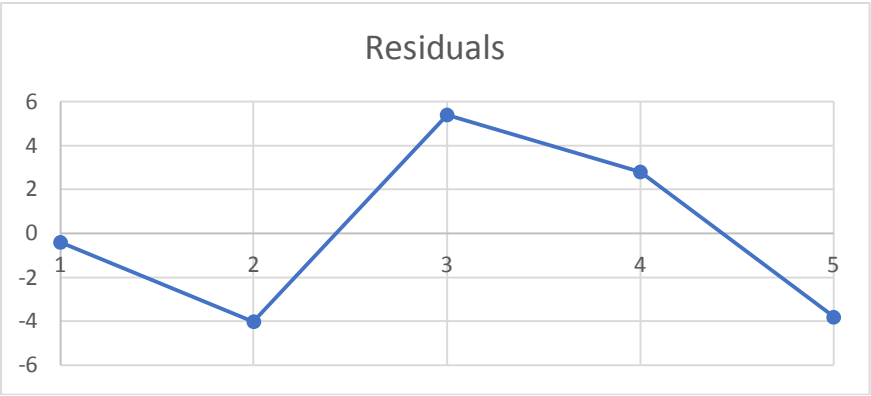


Figure 3.2. Residuals for Exam Scores

Key Insight

Residuals should behave like random noise. Any visible pattern suggests that the model may be misspecified or that one or more assumptions have been violated.

We can also standardize the residuals by subtracting their mean and dividing by their standard deviation. Standardized residuals are useful for detecting outliers because they place all residuals on a common scale. We show the calculation below:

$$standardized\ residuals = \frac{e_i - \bar{e}_i}{standard\ deviation\ e}$$

The average of the residuals equal zero because, on average, the noise has a zero mean. Most standardized residuals should fall between −2 and +2. This range is closely related to the normal distribution, where approximately 95% of observations lie between −1.96 and +1.96. Thus, observations outside this range are relatively rare and may indicate potential outliers that warrant further investigation.

Figure 3.3 presents the standardized residual plot for this example.

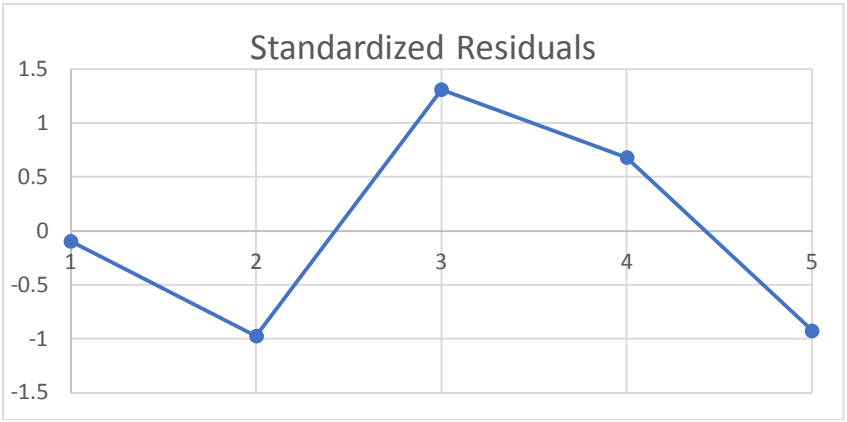


Figure 3.3. Standardized Residuals

In this case, all standardized residuals lie within ±2. Therefore, there is no evidence of outliers in the data.

In practical terms, an outlier in this context might be a student who studies very little but performs exceptionally well on the exam, or a student who studies extensively but performs poorly. Such observations would stand out clearly in the residual analysis and merit closer examination.

3.8 Summary

This chapter introduced the foundations of simple linear regression, one of the most important tools in statistical analysis and empirical research. We began by developing the regression model and learning how to estimate the intercept and slope. These parameters

allow us to quantify the relationship between an independent variable and a dependent variable, providing both interpretation and predictive power.

We then examined how to evaluate the model using the analysis of variance (ANOVA). By decomposing the total variation into explained and unexplained components, we gained a deeper understanding of how well the model fits the data. The F-test allowed us to assess whether the regression as a whole is statistically significant, while t-tests enabled us to evaluate the importance of individual coefficients.

Next, we introduced the coefficient of determination, R^2, as a measure of goodness of fit. This statistic summarizes how much of the variation in the dependent variable is explained by the model. We also connected regression to the Pearson correlation coefficient. It shows how strength and direction of linear relationships are reflected in simple regression.

Finally, we explored residual analysis as a diagnostic tool. By examining residuals, we assessed whether the key assumptions of regression are satisfied. This step is essential, as violations of these assumptions can undermine the validity of our conclusions. Residual plots and standardized residuals provide practical ways to detect issues such as nonlinearity, heteroscedasticity, and outliers.

Taken together, these tools form a complete framework for building, evaluating, and interpreting a regression model. Simple linear regression is more than a method for fitting a line—it is a structured approach to understanding relationships in data, testing hypotheses, and making informed decisions.

In the Chapter 6, we extend these ideas to multiple regression, where we incorporate more than one independent variable. This allows us to model more complex relationships and better reflect the richness of real-world data.

3.9 Exercises

Problem 1.

We want to estimate the relationship between coffee demand and its price. We have the following six observations in Table 3.6. We also include the descriptive statistics.

Table 3.6. Coffee Demand Estimation

Observation	Price (x)	Demand (y)	Predicted y	Residuals	Resid²
1	2	92			
2	3	79			
3	4	77			
4	5	69			
5	6	63			

6	7	52	
Average	4.5	72	SSE
Variance	3.5	192.8	
Covariance	-25.6		

a. Plot the data. Please comment on its shape.

b. Please estimate the slope and intercept. Also, estimate the standard errors, t-statistics, and p-values. Put them in Table 3.7. Perform hypothesis testing for the slope and intercept.

Table 3.7. Regression Estimates

Coefficient	Parameter Estimate	Std Error	t-stat	p-value
intercept				
slope				

c. Calculate the ANOVA for coffee demand. Please evaluate the F statistic. Please place the residuals in the coffee data table 3.6. Then the ANOVA calculations into Table 3.8.

Table 3.8. ANOVA for Coffee Demand

	ANOVA				
	df	SS	MS	F	p-value
SSR					
SSE					
SST					

d. Please calculate the R^2 and Pearson correlation. Please comment on the magnitude.

e. Please plot the standardized residuals. Does it appear that any observation is an outlie?

Problem 2.

A business collected data over 12 consecutive quarters. Revenue (y) and advertising expenditures (x) are both measured in thousands of dollars. Table 3.9 shows the summary statistics.

Table 3.9. Revenue and Advertising Summary Statistics

$Sum(x_i)$	337.80	$Cov(x_i,y_i)$	118.43
$Sum(y_i)$	1,092.00	$Var(x_i)$	43.83
N	12	$Var(y_i)$	418.91

SSE	1,088.29
$\Sigma (e_t)$	0.00

(a) Place the ANOVA calculations in Table 3.10 for this example.

Table 3.10. ANOVA for Revenue

	ANOVA			
	df	SS	MS	F	p-value
SSR					
SSE					
SST					

(b) Please calculate the R^2.

(c) Please comment on the R^2 statistic.

(d) Please calculate the Pearson correlation coefficient. Please comment on it.

(e) Please calculate the least squares estimate b_1.

(f) Please calculate the least squares estimate b_0.

(g) Please calculate the standard errors, t statistics, and p-values for the intercept and slope. Place calculations in Table 3.11. Perform hypothesis testing.

Table 3.11. Regression Results

Coefficient	Parameter Estimate	Std Error	t-stat	p-value
intercept				
slope				

(h) Please calculate the marginal effect of b_1 or $\Delta y/\Delta x \left(or\ dy/dx\right)$ and explain what it means.

(i) Please plot the residuals and comment on the plot.

Chapter 4. Python for Data Analysis

In this chapter, we build a strong foundation in Python by learning its essential commands and structures. We begin by assigning values to variables and then explore how programs make decisions using conditions. Next, we develop loops to automate repetitive tasks and create functions to organize code into reusable, logical blocks. We also introduce dictionaries, which allow us to store and retrieve structured data efficiently.

These tools form the core building blocks of programming. By mastering them, we will be able to write clear, flexible, and efficient code.

Most importantly, this chapter connects programming to statistical analysis. We apply these concepts to the statistical techniques introduced earlier in the book. We show how Python can be used to implement calculations, analyze data, and visualize results. This integration transforms Python from a simple coding language into a practical tool for solving real-world statistical problems.

4.1 Why Python

Python is a high-level programming language that is widely used in data science, business analytics, and machine learning [18, 19]. Its name is inspired by the British comedy group Monty Python. Python was designed to be simple, readable, and efficient [18]. Unlike many other programming languages, it removes unnecessary complexity. It allows users to focus on solving problems rather than struggling with complicated syntax.

In recent years, Python has become extremely popular because of its ability to work with data, build statistical models, and create clear visualizations. It is also open source and supported by a large global community, which means there are many free resources, libraries, and tools available.

For students studying statistics and business, Python provides a powerful and practical way to apply statistical concepts to real-world problems. Instead of only learning theory, students can use Python to analyze data, test ideas, and communicate results effectively.

Key Insight

Python is easy to learn, yet powerful enough to perform advanced data analysis and modeling.

4.2 Setup and Environment

Python can run on Windows, Linux, and macOS. There are several ways to install and use Python, but two of the most popular options for students are Anaconda and Jupyter Notebook. Both allow you to write and run code in a web browser, organizing your work into small, manageable blocks.

Python can also be run as a standalone program using tools such as Microsoft's Command Prompt (CMD) or the Ubuntu terminal. However, in this book, we will use Google Colab. It is one of the easiest ways to get started. Google Colab runs entirely in our web browser and already includes most of the important libraries used in data analysis and statistics [20]. This means we do not need to install anything on our computer.

We can access Google Colab here:

https://colab.research.google.com/

Colab provides an interactive environment where we can write code, run it, and immediately see the results. This is especially useful in statistics because we can combine code, output, and explanations in one place.

The simplest Python command is print(), which displays output on the screen. Python uses indentation (spaces at the beginning of a line) instead of brackets to organize code blocks. This makes programs easier to read, but it also means we must be careful to use consistent spacing.

Let's look at a simple example.

The first line prints the classic message "Hello World," which has been used since the 1980s to introduce programming. Next, we assign the value 1 to a variable named x. The if statement then checks whether x is equal to 1. If the condition is true, Python prints the message "x is 1."

Type the following code into Python, and then click the green arrow to run it: The Python code is available at: https://github.com/KenSzulczyk

```
print("Hello World")

x = 1
if x == 1:
    print("x is 1")
```

The output is on the next page:

```
Hello World
x is 1
```

4.3 Variables and Data Types

In Python, we do not need to declare a variable's type in advance. Python automatically determines the type based on the value we assign. This feature makes Python easy to use, but it is still important to understand the type of data we are working with.

The most common data types in Python include:

- Integers: whole numbers, e.g., 1, 5, 100
- Floating-point numbers: numbers with decimals, e.g., 3.14, 5.0
- Strings: text enclosed in quotation marks, e.g., "John"

Let's look at a simple example.

- The first line assigns the value 5 to the variable x. This is an integer.
- The second line assigns 5.0 to y, which is a floating-point number.
- The third line assigns the text "John" to the variable name, which is a string.

To see the type of each variable, we use the type() function.

Click + Code to create a new code cell, and then run the following commands. The Python code is available at: https://github.com/KenSzulczyk

```
x = 5
y = 5.0
name = "John"

print(type(x))
print(type(y))
print(type(name))
```

The output is below:

```
<class 'int'>
<class 'float'>
<class 'str'>
```

Key Insight

Python automatically assigns data types, but we should always be aware of them because they affect how calculations and operations behave.

4.4 Lists

A list is a way to store multiple values in a single variable. In statistics and data analysis, lists are very useful because we often work with collections of numbers.

In Python, a list is written using square brackets [ ], with values separated by commas. For example, [1, 2, 3, 4, 5] is a list of numbers.

Python lists are indexed starting at zero, which means:

- The first element has index 0
- The second element has index 1
- And so on

We can use a loop to go through each value in a list. A loop allows us to repeat an action for every element. In this case, we will use a for loop to read each number in the list and print it.

Click + Code to create a new code cell, and then run the following commands. The Python code is available at: https://github.com/KenSzulczyk

```
mylist = [1,2,3,4,5]

for x in mylist:
    print(x)
```

How the code works:

- mylist stores five numbers in a list
- for x in mylist: tells Python to go through each value in the list
- x takes on one value at a time (1, then 2, then 3, etc.)
- print(x) displays each value

The output is below:

```
1
2
3
4
5
```

4.5 Operators

Python supports standard arithmetic operations such as addition, subtraction, multiplication, division, and exponents. These operators allow us to perform calculations quickly and efficiently.

In the first example, we calculate 2^3 using the ** operator, which represents an exponent. In the second example, we divide 10 by 3 and use the % operator to find the remainder, which is also called the modulus.

Click + Code to create a new code cell, and then run the following commands. The Python code is available at: https://github.com/KenSzulczyk

```
x = 2 ** 3
y = 10 % 3
print(x, y)
```

The output is below:

```
8 1
```

Next, we look at a more detailed example using a Python library. A library is a collection of pre-written functions that extend Python's capabilities. In this example, we use the built-in statistics library to calculate the mean and standard deviation [21].

First, we create a list of data values. Then, we calculate:

- the mean (average) using statistics.mean()
- the standard deviation using statistics.stdev()

The notation statistics.mean tells Python to use the mean function from the statistics library.

Finally, we print the results using an f-string. The {} brackets allow us to insert variable values directly into the text.

Click + Code to create a new code cell, and then run the following commands. The Python code is available at: https://github.com/KenSzulczyk

```
import statistics

data = [10, 13, 19, 18, 17, 24, 20, 15]

# Calculate the mean (sample arithmetic mean)
```

```
mean_value = statistics.mean(data)
# Calculate the standard deviation (sample standard deviation)
std_deviation = statistics.stdev(data)

print(f"Mean of the data: {mean_value}")
print(f"Standard Deviation of the data: {std_deviation}")
```

The output is below:

```
Mean of the data: 17
Standard Deviation of the data: 4.342481186734475
```

4.6 Strings

A string is a sequence of text characters, such as words or sentences. In Python, strings are written inside quotation marks, for example “Hello World”.

Python provides many useful tools for working with strings, including:

- measuring the length of a string
- changing letter case
- selecting specific parts of the text, called slicing

Let’s look at an example.

- The first line assigns the text "Hello World" to the variable text.
- The len() function counts how many characters are in the string, including spaces.
- The upper() method converts all letters to uppercase.
- The final command selects the first five characters of the string.

Remember, Python uses zero-based indexing, so counting starts at 0. This means:

- index 0 is the first character
- index 4 is the fifth character

Click + Code to create a new code cell, and then run the following commands. The Python code is available at: https://github.com/KenSzulczyk

```
text = "Hello World"
print(len(text))
print(text.upper())
print(text[0:5])
```

The output is below:

```
11
```

```
HELLO WORLD
Hello
```

Please note that Python only takes the straight " and '. Using curly quotation marks or apostrophe result in an error.

Key Insight

Strings allow us to work with text data, and Python provides simple tools to measure, modify, and extract parts of that text.

4.7 Conditional Statements

Conditions allow programs to make decisions. Instead of running every line of code in order, a program can choose different actions depending on whether a statement is True or False. This ability makes programs flexible, interactive, and intelligent.

A condition evaluates an expression and returns either True or False. Based on this result, Python controls the flow of the program by deciding which block of code to execute.

In the example below, we create a condition to check whether x equals 2. It is important to distinguish between the assignment operator and the equality operator. A single equal sign (=) assigns a value to a variable, while two equal signs (==) test whether two values are equal.

If the condition evaluates to True, Python executes the code inside the if block. If the condition evaluates to False, Python skips that block and executes the code inside the else block instead.

Click + Code to create a new code cell, and then run the following commands. The Python code is available at: https://github.com/KenSzulczyk

```
x = 2
if x == 2:
    print("x equals 2")
else:
    print("x does not equal 2")
```

The output is below:

```
x equals 2
```

Key Insight

Conditions are the foundation of decision-making in programming. They allow a program to adapt its behavior based on data, user input, or changing circumstances. Without conditions, programs would always behave the same way, regardless of the situation.

4.8 Loops

Loops allow programs to repeat code efficiently. Instead of writing the same instructions multiple times, a loop executes a block of code repeatedly. This is especially useful when working with data, performing calculations, or automating repetitive tasks.

Python provides two main types of loops: the for loop and the while loop.

A for loop is commonly used to iterate over a sequence of values, such as numbers, lists, or strings. In many cases, we use the range() function to specify how many times the loop should run.

A while loop works differently. It continues executing as long as a specified condition remains True. Once the condition becomes False, the loop stops.

In the example below, the for loop prints the numbers from 0 to 4. The range(5) function generates five values: 0, 1, 2, 3, and 4.

The while loop also prints numbers from 100 to 104. It starts with count = 100 and continues looping as long as count < 104. Inside the loop, the variable count increases by 1 each time using count += 1.

Click + Code to create a new code cell, and then run the following commands. The Python code is available at: https://github.com/KenSzulczyk

```
for i in range(5):
    print(i)

count = 100
while count < 104:
    print(count)
    count += 1
```

The output is below:

```
0
1
2
3
```

```
4
100
101
102
103
```

Key Insight

Loops are a core building block of programming. They allow us to process large amounts of data, automate repetitive tasks, and write concise, efficient code. Without loops, even simple tasks would require many repeated lines of code, making programs longer and harder to manage.

4.9 Functions

Functions organize code into reusable blocks. Instead of writing the same code multiple times, we can define a function once and use it whenever needed. This makes programs easier to read, debug, and maintain.

A function takes inputs, performs a task, and can return an output. The inputs are called parameters, and the output is specified using the return statement.

In the example below, we define a function called add that takes two parameters, a and b. The function adds these two values together and returns the result.

We then call the function by passing in the numbers 2 and 3. The function computes the sum and returns 5.

Click + Code to create a new code cell, and then run the following commands. The Python code is available at: https://github.com/KenSzulczyk

```
def add(a,b):
    return a + b

print(add(2,3))
```

The output is below:

```
5
```

Key Insight

Functions are one of the most powerful tools in programming. They promote code reuse, improve organization, and make programs easier to understand. As programs grow larger,

functions become essential for managing complexity and building structured, efficient solutions.

4.10 Dictionaries

Dictionaries store data as key-value pairs. They are useful for organizing structured data, such as names and phone numbers, where each key is associated with a specific value.

In the example below, we use a dictionary to store names and phone numbers in an object called phonebook. Each name (the key) is linked to a phone number (the value).

To access a value, we place the key inside square brackets. In this case, we insert "John" into the dictionary, and Python returns John's phone number.

Click + Code to create a new code cell, and then run the following commands. The Python code is available at: https://github.com/KenSzulczyk

```
phonebook = {"John":123456, "Jane":456554, "Ken": 484545}

print("John phone number is: ", phonebook["John"])
```

The output is below:

```
John phone number is:  123456
```

Be careful with quotation marks when working with text. Python allows both single (' ') and double (" ") quotation marks. They can often be used interchangeably. However, if a string contains an apostrophe (for example, "John's phone"), using single quotes without care can cause errors because the apostrophe may be interpreted as the end of the string.

Big Idea

Dictionaries are a powerful way to store and retrieve data quickly. They allow us to connect related pieces of information and access them using meaningful labels instead of numeric positions. This makes our code more intuitive and easier to work with, especially when handling real-world data.

4.11 Libraries and Modules

Python's strength comes from its vast ecosystem of modules and libraries. A module is a file containing Python code, while a library is a collection of related modules. These tools allow users to perform complex tasks without writing everything from scratch.

Instead of building advanced features our self, we can import existing libraries that have already been optimized and tested. This saves time, reduces errors, and allows us to focus on solving real-world problems.

Some of the most important libraries include:

- NumPy for numerical computations and working with arrays [22]
- Pandas for data analysis and data manipulation [19]
- Matplotlib for creating graphs and visualizations [23]
- Seaborn, which is built on top of Matplotlib, for more visually appealing and statistical graphics

To use a library, we must first import it. In the example below, we import NumPy and Matplotlib. We assign them shorter names, called aliases, using the as keyword. This allows us to write less code and improves readability.

- numpy becomes np
- matplotlib.pyplot becomes plt

The linspace() function from NumPy creates 100 evenly spaced values between 0 and 50. We then use Matplotlib's plot() function to graph x and x^2. Finally, the show() function tells Python to display the graph. In some environments, this command is necessary; otherwise, the graph may not appear.

We also use the # symbol to insert comments. Comments are ignored by Python but are extremely important for explaining what the code does. When we return to our code later, comments help us and others quickly understand our logic.

Click + Code to create a new code cell, and then run the following commands. The Python code is available at: https://github.com/KenSzulczyk

```
import numpy as np
import matplotlib.pyplot as plt

# Create 200 x values between 0 and 100
x = np.linspace(0,100,200)

# Plot the graph of x and x squared
plt.plot(x, x**2)

plt.show()
```

The Python output is Figure 4.1.

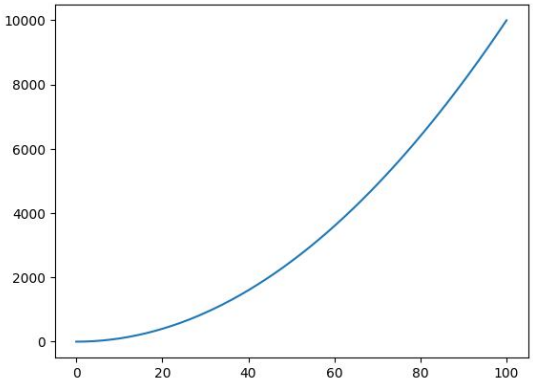


Figure 4.1. X and X Squared Relationship

Key Insight

Libraries allow Python to solve real-world business, scientific, and statistical problems efficiently. They transform Python from a simple programming language into a powerful analytical tool capable of handling large datasets, complex models, and professional-quality visualizations.

4.12 Practice: Chapter 1 Problems

In this section, we revisit key examples from Chapter 1 and solve them using Python. This allows us to verify our earlier results while introducing a more efficient computational approach.

We begin with the dataset on temperature and electricity demand from Chapter 1, shown in Table 4.1. A city energy planner is interested in understanding how weather conditions affect electricity usage. To explore this relationship, the planner collects data on average daily temperature and electricity demand over five days.

Table 4.1. Temperature vs. Electricity Demand

Day	Temperature (°C)	Electricity Demand (MWh)
1	22	210
2	25	250
3	28	290
4	31	330
5	27	270

We compute the descriptive statistics and correlation using Python. The Python code is available at: https://github.com/KenSzulczyk

```
# ------------------------------------------------
```

```
# Temperature and Electricity Demand Analysis
# ------------------------------------------------

# Step 1: Import libraries
import pandas as pd

# ------------------------------------------------
# Step 2: Manually create the dataset
# ------------------------------------------------

data = {
    "Day": [1, 2, 3, 4, 5],
    "Temperature": [22, 25, 28, 31, 27],
    "Electricity_Demand": [210, 250, 290, 330, 270]
}

# Convert to DataFrame
df = pd.DataFrame(data)

# Display dataset
print("Dataset:")
print(df)

# ------------------------------------------------
# Step 3: Descriptive Statistics
# ------------------------------------------------

print("\nDescriptive Statistics:")

# Minimum values
print("\nMinimum:")
print(df[["Temperature", "Electricity_Demand"]].min())

# Mean values
print("\nMean:")
print(df[["Temperature", "Electricity_Demand"]].mean())

# Maximum values
print("\nMaximum:")
print(df[["Temperature", "Electricity_Demand"]].max())

# Standard deviation
print("\nStandard Deviation:")
print(df[["Temperature", "Electricity_Demand"]].std())

# ------------------------------------------------
# Step 4: Pearson Correlation
# ------------------------------------------------

correlation = df["Temperature"].corr(df["Electricity_Demand"])

print("\nPearson Correlation:")
print(correlation)
```

The Python output is below:

```
Dataset:
   Day  Temperature  Electricity_Demand
0    1           22                 210
1    2           25                 250
2    3           28                 290
3    4           31                 330
4    5           27                 270

Descriptive Statistics:

Minimum:
Temperature            22
Electricity_Demand    210
dtype: int64

Mean:
Temperature            26.6
Electricity_Demand    270.0
dtype: float64

Maximum:
Temperature            31
Electricity_Demand    330
dtype: int64

Standard Deviation:
Temperature            3.361547
Electricity_Demand    44.721360
dtype: float64

Pearson Correlation:
0.9977851578566089
```

The output shows:

- The mean temperature is 26.6°C, and the mean electricity demand is 270 MWh
- The standard deviations indicate moderate variability in both variables
- The Pearson correlation is approximately 0.998, indicating a very strong positive linear relationship

As temperature increases, electricity demand also increases, likely due to higher usage of cooling systems. Importantly, these results match exactly with the calculations we performed manually in Chapter 1.

Next, we revisit the hypothesis testing problem involving the lifetimes of two types of light bulbs. The summary statistics are shown in Table 4.2.

Table 4.2. Bulb Lifetime Comparison

Brand A (Energy Efficient)	Brand B (Standard)
$n_1 = 30$	$n_2 = 35$
$\bar{x}_1 = 1{,}240$	$\bar{x}_2 = 1{,}180$
$s_1 = 120$	$s_2 = 150$

We first state the hypotheses. We test whether Brand A has a longer lifetime than Brand B:

Ho: $\mu_1 \leq \mu_2$
Ha: $\mu_1 > \mu_2$

We use Python to perform Welch's two-sample t-test based on summary statistics. The Python code is available at: https://github.com/KenSzulczyk

```
# ------------------------------------------------
# Two-Sample t-Test (Welch's Test)
# Comparing Bulb Lifetimes
# ------------------------------------------------

# Step 1: Import libraries
import numpy as np
from scipy import stats

# ------------------------------------------------
# Step 2: Input sample statistics
# ------------------------------------------------

n1 = 30
x1_bar = 1240
s1 = 120

n2 = 35
x2_bar = 1180
s2 = 150

# ------------------------------------------------
# Step 3: Compute test statistic
# ------------------------------------------------

# Standard error
se = np.sqrt((s1**2 / n1) + (s2**2 / n2))

# t-statistic
t_stat = (x1_bar - x2_bar) / se

print("t-statistic:", t_stat)

# ------------------------------------------------
# Step 4: Degrees of freedom (Welch formula)
# ------------------------------------------------
```

```
df = ((s1**2 / n1 + s2**2 / n2)**2) / \
     (((s1**2 / n1)**2 / (n1 - 1)) + ((s2**2 / n2)**2 / (n2 - 1)))

print("Degrees of freedom:", df)

# ------------------------------------------------
# Step 5: Compute p-value (one-tailed test)
# ------------------------------------------------

p_value = 1 - stats.t.cdf(t_stat, df)

print("p-value:", p_value)

# ------------------------------------------------
# Step 6: Decision
# ------------------------------------------------

alpha = 0.05

if p_value < alpha:
    print("Reject H0: Brand A lasts longer than Brand B.")
else:
    print("Fail to reject H0: Not enough evidence that Brand A lasts longer.")
```

The output is below:

```
t-statistic: 1.7905604905249661
Degrees of freedom: 62.727743198306584
p-value: 0.039094699652395426
Reject H0: Brand A lasts longer than Brand B.
```

Interpreting the results:

- The t-statistic is approximately 1.79
- The p-value is approximately 0.039

Since the p-value is less than 0.05, we reject the null hypothesis. There is sufficient evidence to conclude that Brand A has a longer average lifetime than Brand B. Once again, this result matches the conclusion obtained in Chapter 1 using manual calculations.

Python provides an even simpler method when raw data are available. Instead of computing formulas manually, we can pass the data directly into a built-in function.

The Python program is below:

```
from scipy import stats

x1 = [ ... 30 observations ... ]
x2 = [ ... 35 observations ... ]
```

```
# If we have raw data:
result = stats.ttest_ind(x1, x2, equal_var=False)

# Print out the t-statistic and p-value
print("t-statistic:", result.statistic)
print("p-value:", result.pvalue)
```

This approach automatically computes the test statistic and p-value. It saves time and reduces the risk of calculation errors.

Key Insight

Python allows us to replicate manual statistical calculations quickly and accurately [19, 21]. It is a powerful tool for both learning and applied data analysis.

4.13 Summary

In this chapter, we introduced Python as a practical and powerful tool for data analysis. We explored the fundamental building blocks of programming, including variables, conditions, loops, functions, and dictionaries. Together, these concepts form the foundation for writing programs that can process data, automate tasks, and support statistical analysis.

As we continue learning Python, it is important to focus on understanding the logic behind the code rather than memorizing specific syntax. Syntax can always be looked up, but a clear understanding of how and why code works will allow us to solve new problems with confidence.

This chapter serves as a bridge between programming and statistics. In Chapter 5, we will apply Python to the problems introduced in Chapters 2 and 3. Specifically, we will use Python to estimate ANOVA models from Chapter 2 and solve linear regression models from Chapter 3. By doing so, we will see how programming transforms theoretical concepts into practical tools for real data analysis.

Big Picture

Learning Python is not just about writing code—it is about developing a new way of thinking. By combining programming with statistical reasoning, we gain the ability to analyze data, test ideas, and make informed decisions in a wide range of real-world applications.

4.14 Exercises

For this chapter's problems, we solve the problems at the end of Chapter 1. Please try to write the Python code before looking at the answers.

Problem 1.

We visit the End-of-Chapter Problems from Chapter 1. An investor is deciding whether to diversify their portfolio by investing in two sectors: technology and energy. The annual returns (%) for two representative stocks over five years are given in the Python code below. The Python code is available at: https://github.com/KenSzulczyk

```
# Libraries
import pandas as pd

# Create the dataset using pandas
data = {
    "Year": [1, 2, 3, 4, 5],
    "Tech": [10, 18, -6, 20, 9],
    "Energy": [4, 12, -3, 8, 7]
}

df = pd.DataFrame(data)

# Display the dataset
print("Dataset:")
print(df)
print()
```

a. Please compute the following for each stock: minimum, mean, maximum, standard deviation, skewness, and kurtosis.

b. Compute the correlation coefficient between the two stocks.

c. Based on your results:
- Which stock offers higher average returns?
- Which stock is riskier?
- Does diversification appear beneficial?

Problem 2.

A national survey reports that adults spend an average of 6.5 hours per day on screens. A researcher suspects that remote workers spend more time on screens than this average. A sample of 36 remote workers shows an average screen time of 7.1 hours per day. Assume the population standard deviation is 1.8 hours.

a. Compute the test statistic.

b. Compute the p-value.

c. At $\alpha = 0.05$, state your conclusion.

The Python code is below:

```
# Import the library
import math
from scipy.stats import norm

# Given data
x_bar = 7.1      # sample mean
mu = 6.5         # population mean
sigma = 1.8      # population standard deviation
n = 36           # sample size
alpha = 0.05     # significance level
```

Problem 3.

A technology company wants to compare the average battery life of two smartphone models.

Model A uses a new battery design
Model B uses the current standard battery

The company wants to know whether the mean battery life for the new design lasts longer than the standard battery. Table 4.3 contains the data.

Table 4.3. Battery Life Data

Model A (New Battery)	Model B (Standard Battery)
$n_1 = 36$	$n_2 = 49$
$\bar{x}_1 = 11.8$ hours	$\bar{x}_2 = 10.9$ hours
$\sigma_1 = 2.4$ hours	$\sigma_2 = 2.8$ hours

a. State the hypotheses

b. Compute the test statistic

c. Compute the p-value

d. What is our conclusion using $\alpha = 0.05$?

Problem 4.

A university wants to compare the average exam scores of two teaching methods.

Method A uses traditional lectures
Method B uses interactive learning

The university wants to determine whether the mean scores differ between the two methods. Table 4.4 shows the summary statistics.

Table 4.4. Teaching Method Data

Method A (Traditional)	Method B (Interactive)
$n_1 = 25$	$n_2 = 30$
$\bar{x}_1 = 78.6$	$\bar{x}_2 = 74.2$
$s_1 = 10.5$	$s_2 = 9.8$

a. State the hypotheses

b. Compute the test statistic

c. Compute the p-value

d. What is our conclusion using $\alpha = 0.05$?

The first part of the Python program is below:

```
# ------------------------------------------------
# Two-Sample t-Test (Welch's Test)
# Comparing Teaching Methods
# ------------------------------------------------

# Step 1: Import libraries
import numpy as np
from scipy import stats

# ------------------------------------------------
# Step 2: Input sample statistics
# ------------------------------------------------

# Method A (Traditional)
n1 = 25
```

```
x1_bar = 78.6
s1 = 10.5

# Method B (Interactive)
n2 = 30
x2_bar = 74.2
s2 = 9.8
```

Problem 5.

A retail company is working to improve the consistency of its customer experience. Management believes that reducing variability in customer satisfaction is just as important as increasing the average score.

Customer satisfaction is measured on a 100-point scale. Based on past data, the company expects a population standard deviation of $\sigma = 10$.

After implementing a new employee training program, the company collects a sample of 15 customer satisfaction scores. The Python code holds the 15 observations.

```
# Libraries
import pandas as pd
from scipy.stats import chi2

# Create the dataset
data = {
    "Score": [88, 84, 91, 79, 85,
              87, 82, 90, 86, 83,
              89, 81, 92, 78, 84]
}

df = pd.DataFrame(data)
```

a. What is the sample mean customer satisfaction score?

b. What is the sample variance?

c. What is the sample standard deviation?

d. Conduct a hypothesis test to determine whether the variability has changed after the training program. Use a significance level of 0.05.

- State the null and alternative hypotheses
- Compute the test statistic
- State the decision rule
- Provide your conclusion

Problem 6.

A manufacturing company operates two production lines that produce the same product. Management wants to ensure that both lines maintain consistent quality. In particular, they are concerned about variability in product thickness, since high variability can lead to defects.

To evaluate consistency, the company collects samples from each production line:

Production Line 1: $n_1 = 21$, $s_1^2 = 8.2$
Production Line 2: $n_2 = 26$, $s_2^2 = 3.0$

Management wants to test whether the variability of the two production lines is the same. Please use Python to solve this problem. The hypotheses are below:

H_0: $\sigma_1^2 = \sigma_2^2$
H_a: $\sigma_1^2 \neq \sigma_2^2$

a. Use the p-value approach to test the hypotheses at the 5% significance level.

Hint: Place the larger sample variance in the numerator and conduct a right-tailed test.

b. Repeat the test using the critical value approach.

Chapter 5. Python for Statistics

In the previous chapters, we developed the theory and worked through calculations by hand. Now we take the next step: we let Python do the heavy lifting. With a few lines of code, we can analyze datasets that would take hours—or even days—to compute manually.

In this chapter, we use Python to bring data to life. We begin by constructing and importing datasets, then summarize them using descriptive statistics. We explore relationships between variables with the Pearson correlation coefficient and visualize the data using box-and-whisker plots and scatter plots. These tools allow us to move beyond numbers and see the structure of the data directly.

We then revisit the problems from Chapters 2 and 3, using Python to verify our earlier results. This transition is crucial. The goal is not to replace theory, but to extend it. By automating calculations, we can work with larger datasets, reduce errors, and focus on what truly matters—interpretation, intuition, and economic insight.

5.1 Descriptive Statistics and Correlation

We begin with a dataset on coffee consumption spanning the years 1990 to 2023. The dataset contains annual observations and includes several economically meaningful variables. Coffee consumption is measured as cups per person per year. Coffee and tea prices are measured in dollars per cup, while sugar prices are measured in dollars per pound. Annual income is reported in dollars per household per year.

This dataset allows us to explore consumer behavior and test basic economic relationships such as the Law of Demand, income effects, and substitution patterns.

Before estimating any formal econometric model, it is essential to explore and understand the data. Descriptive statistics and correlation analysis provide a first look at the structure of the dataset, helping us identify patterns, potential anomalies, and relationships between variables.

For now, we manually enter the dataset directly into Python. This approach allows students to focus on data structure and analysis without introducing file handling. In Chapter 6, we will extend this approach by importing datasets from CSV and Excel files.

Click + Code to create a new code cell, and run the following. The Python code is available at: https://github.com/KenSzulczyk

```
# Import libraries
```

```
import pandas as pd

# Step 1: Create the dataset
data = {
    "Year": [
        1990,1991,1992,1993,1994,1995,1996,1997,1998,1999,
        2000,2001,2002,2003,2004,2005,2006,2007,2008,2009,
        2010,2011,2012,2013,2014,2015,2016,2017,2018,2019,
        2020,2021,2022,2023
    ],
    "Coffee_consumption": [
        267.22,270.57,309.06,301.31,214.79,227.25,258.69,197.95,256.66,292.83,
        312.61,362.14,369.48,378.24,358.46,335.95,328.86,331.54,313.35,320.28,
        293.27,180.05,269.84,340.41,266.42,352.33,346.80,384.27,417.02,437.73,
        431.12,378.27,315.29,376.35
    ],
    "Coffee_price": [
        0.91,0.86,0.65,0.69,1.45,1.46,1.16,1.79,1.29,1.04,
        0.90,0.56,0.54,0.62,0.77,1.08,1.08,1.18,1.33,1.26,
        1.64,2.53,1.75,1.26,1.78,1.33,1.37,1.33,1.14,1.02,
        1.11,1.69,2.14,1.70
    ],
    "Sugar_price": [
        0.13,0.09,0.09,0.10,0.12,0.12,0.11,0.11,0.09,0.06,
        0.08,0.08,0.06,0.07,0.07,0.10,0.15,0.10,0.12,0.18,
        0.22,0.27,0.22,0.17,0.16,0.13,0.18,0.16,0.12,0.12,
        0.13,0.18,0.19,0.24
    ],
    "Tea_price": [
        0.92,0.84,0.91,0.84,0.83,0.75,0.81,1.08,1.08,1.06,
        1.13,0.90,0.81,0.88,0.90,0.98,1.10,0.96,1.23,1.43,
        1.44,1.57,1.59,1.21,1.08,1.55,1.31,1.65,1.36,1.22,
        1.15,1.21,1.26,1.32
    ],
    "Annual_income": [
        42650,43240,44220,47220,49340,51350,53680,56900,59590,62570,
        65770,66860,66970,68560,70390,73300,77320,78850,79630,78540,
        78180,81010,82840,87670,88770,92670,97360,103200,106000,116700,
        115300,121800,126500,135700
    ]
}

# Create a Pandas dataframe
df = pd.DataFrame(data)

# Display first few rows
print(df.head())
```

The Python output is below:

```
   Year  Coffee_consumption  Coffee_price  Sugar_price  Tea_price  \
0  1990              267.22          0.91         0.13       0.92
1  1991              270.57          0.86         0.09       0.84
```

```
2  1992              309.06          0.65          0.09        0.91
3  1993              301.31          0.69          0.10        0.84
4  1994              214.79          1.45          0.12        0.83

   Annual_income
0          42650
1          43240
2          44220
3          47220
4          49340
```

The data are stored in a Pandas DataFrame, which resembles an Excel spreadsheet. Variable names appear as column headers, and each row represents an observation. The head() command displays the first five observations. It allows us to quickly verify that the data were entered correctly. If we wanted the first 20 observations, we put 20 into the parenthesis. Lastly, if we wanted the end of the dataset, we use .tail(). It functions similarly to .head().

Next, we compute descriptive statistics to summarize the dataset. These statistics provide insight into the distribution, variability, and shape of each variable.

We first create a copy of the dataset and remove the Year variable. Although time is important for later analysis, it is not a variable in the economic sense. It should be excluded in summary statistics.

We then compute the minimum, average, maximum, standard deviation, skewness, and kurtosis. Storing these results in a DataFrame produces a clean, tabular output.

Click + Code to create a new code cell, and then run the following commands. The Python code is available at: https://github.com/KenSzulczyk

```
# Descriptive statistics

# Create a copy of the dataframe
stats = df.copy()

# Remove the year
stats.drop("Year", axis=1, inplace=True)

# Compute statistics
desc_stats = pd.DataFrame({
    "Minimum": stats.min(),
    "Average": stats.mean(),
    "Maximum": stats.max(),
    "Std Dev": stats.std(),
    "Skewness": stats.skew(),
    "Kurtosis": stats.kurt()
})
```

```
# Round for nicer display (optional)
desc_stats = desc_stats.round(2)

# Print results
print("Descriptive Statistics")
print(desc_stats.T)
```

The Python output is below:

```
Descriptive Statistics
          Coffee_consumption  Coffee_price  Sugar_price  Tea_price  \
Minimum               180.05          0.54         0.06       0.75
Average               317.54          1.25         0.13       1.13
Maximum               437.73          2.53         0.27       1.65
Std Dev                63.35          0.45         0.05       0.25
Skewness               -0.20          0.66         0.81       0.45
Kurtosis               -0.27          0.69         0.10      -0.78

          Annual_income
Minimum        42650.00
Average        78548.53
Maximum       135700.00
Std Dev        25220.28
Skewness           0.56
Kurtosis          -0.40
```

The transpose (.T) rearranges the table so that statistics appear as rows and variables as columns. This format is commonly used in academic papers and textbooks.

Descriptive statistics serve several purposes:

- They reveal the scale and variability of each variable
- They help identify outliers or unusual observations
- They provide insight into distributional shape through skewness and kurtosis

For example, large standard deviations indicate substantial variability, while skewness reveals whether the data are symmetric or skewed.

After summarizing each variable individually, we examine how variables move together using a correlation matrix. As before, we remove the Year variable to avoid treating time as an explanatory variable.

Click + Code to create a new code cell, and then run the following commands. The Python code is available at: https://github.com/KenSzulczyk

```
# Correlation matrix

# Remove Year variable
stats = df.drop("Year", axis=1)
```

```
# Compute correlation matrix
corr_matrix = stats.corr()

# Round for nicer display
corr_matrix = corr_matrix.round(2)

# Display results
print("Correlation Matrix")
print(corr_matrix)
```

The Python output is below:

```
Correlation Matrix
                    Coffee_consumption  Coffee_price  Sugar_price  Tea_price  \
Coffee_consumption                1.00         -0.43        -0.14       0.17
Coffee_price                     -0.43          1.00         0.80       0.59
Sugar_price                      -0.14          0.80         1.00       0.72
Tea_price                         0.17          0.59         0.72       1.00
Annual_income                     0.61          0.44         0.54       0.63

                    Annual_income
Coffee_consumption           0.61
Coffee_price                 0.44
Sugar_price                  0.54
Tea_price                    0.63
Annual_income                1.00
```

The correlation matrix measures the strength and direction of linear relationships between variables, with values ranging from −1 to +1.

Correlation analysis provides a preliminary check of whether the data are consistent with economic theory. Coffee price and consumption exhibit a negative relationship, supporting the Law of Demand. Sugar price is negatively correlated with coffee consumption, consistent with sugar being a complement. Higher sugar prices reduce coffee consumption. Tea price is positively correlated with coffee consumption, suggesting tea is a substitute. As tea becomes more expensive, consumers shift toward coffee. Income is positively correlated with coffee consumption, indicating that coffee is a normal good.

Key Insight

While correlations are informative, they do not imply causation. Correlation measures association, not structural relationships. For example, two variables may be correlated due to a third, omitted factor or due to common trends over time.

This highlights the need for regression analysis, which allows us to control for multiple factors simultaneously and test economic hypotheses more rigorously.

5.2 Data Visualization

Graphical analysis is a powerful complement to descriptive statistics. Well-designed plots allow us to quickly identify patterns, detect outliers, and assess relationships between variables. In many cases, a visual inspection of the data provides intuition that guides formal econometric modeling.

In this section, we construct two commonly used plots: box plots and scatter plots. We begin with box-and-whisker plots, which provide a compact visual summary of a variable's distribution. A box plot displays the median, dispersion, and potential outliers in a single figure.

The code structure follows the same logic as before. We create a new DataFrame that includes only the variables of interest. In this case, we exclude both Year and Annual_income. The Year variable is not meaningful for distributional analysis. In addition, Annual_income is measured on a much larger scale and would dominate the visualization. Removing it allows us to present the remaining variables clearly.

We then create a grid of subplots so that each variable is displayed on its own axis.

Click + Code to create a new code cell, and run the following. The Python code is available at: https://github.com/KenSzulczyk

```
# Import library
import matplotlib.pyplot as plt

# Remove Year variable
stats = df.drop( ["Year","Annual_income"], axis=1)

# Create subplots (one for each variable)
fig, axes = plt.subplots(nrows=2, ncols=2)

# Flatten axes for easy looping
axes = axes.flatten()

# Loop through variables and plot
for i, col in enumerate(stats.columns):
    axes[i].boxplot(stats[col])
    axes[i].set_title(col)

# Adjust layout
plt.tight_layout()

# Show plot
plt.show()
```

The Python output is Figure 5.1. This plot summarizes the distributions of coffee consumption and prices. The orange line inside each box represents the median, while the top and bottom of the box correspond to the 75th and 25th percentiles, respectively. The distance between these two percentiles is known as the interquartile range (IQR) and measures the spread of the middle 50 percent of the data.

Figure 5.1. Coffee Data Box Plot

The whiskers extend to the range of the data, excluding extreme observations, while individual points indicate outliers.

Several insights emerge from the figure:

- Coffee consumption appears relatively symmetric, suggesting a balanced distribution over time.
- Coffee, sugar, and tea prices exhibit some degree of positive skewness, as indicated by longer upper whiskers.
- Coffee prices include at least one outlier, highlighting periods of unusually high prices.

These observations are consistent with the skewness and kurtosis statistics computed earlier. It demonstrates how graphical and numerical summaries complement one another.

Next, we examine the relationship between coffee price and coffee consumption using a scatter plot. This visualization allows us to assess whether the data are consistent with the Law of Demand, which predicts an inverse relationship between price and quantity demanded.

We use the Matplotlib library to construct the plot and add a fitted trend line to highlight the overall relationship.

Click + Code to create a new code cell, and run the following. The Python code is available at: https://github.com/KenSzulczyk

```
# Import libraries
import matplotlib.pyplot as plt
import numpy as np

# Define variables (switched axes)
x = df["Coffee_consumption"]
y = df["Coffee_price"]

# Create scatter plot
plt.figure()
plt.scatter(x, y, color="cornflowerblue")

# Add regression line
m, b = np.polyfit(x, y, 1)

# Sort for smooth line
sorted_idx = np.argsort(x)
x_sorted = x.iloc[sorted_idx]
y_pred = m * x_sorted + b

plt.plot(x_sorted, y_pred, color="crimson")

# Labels and title
plt.xlabel("Coffee Consumption")
plt.ylabel("Coffee Price")
plt.title("Coffee Price vs Coffee Consumption")

# Grid
plt.grid()

# Show plot
plt.show()
```

The Python output is Figure 5.2. This plot reveals a negative relationship between coffee price and coffee consumption. Although the relationship is not perfectly linear, the overall pattern suggests that higher prices are associated with lower consumption.

The fitted line (shown in red) represents a simple linear approximation of this relationship. Conceptually, this line corresponds to a simple regression model of coffee price on coffee consumption. The negative slope of the line provides visual evidence consistent with the Law of Demand.

Several important insights can be drawn from this figure:

- The negative slope supports standard economic theory, the Law of Demand.

- The dispersion of points indicates that factors other than price—such as income and substitute goods—also influence consumption.
- The presence of variability around the line suggests that a multivariate regression model is necessary to better explain coffee demand.

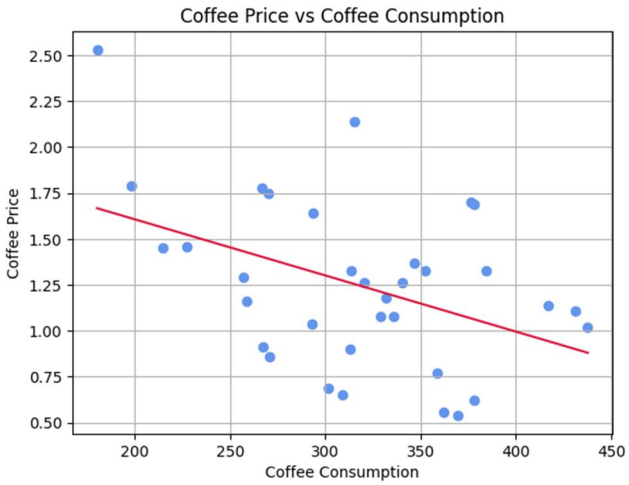


Figure 5.2. Coffee Price vs. Consumption

Key Insight

Graphical analysis serves as a bridge between raw data and formal econometric modeling. The patterns observed in Figures 5.1 and 5.2 motivate the specification of a demand function that includes prices, income, and related goods.

5.3 ANOVA in Python

In Chapter 2, we introduced analysis of variance (ANOVA) through several experimental design problems and computed all results by hand. While these manual calculations are essential for understanding the mechanics of ANOVA, they quickly become tedious in practice.

In this section, we revisit those same problems and use Python to perform the analysis. This approach allows us to focus on interpretation and economic reasoning, rather than arithmetic.

In Example 1, Magic Wax tests three types of car wax—Type 1, Type 2, and Type 3—to compare durability. Each wax is applied to five cars, and durability is measured by the number of washes the wax withstands before deterioration.

We test whether all waxes have the same average durability.

H_0: $\mu_1 = \mu_2 = \mu_3$
H_a: At least one population mean differs from the others

Click + Code to create a new code cell, and run the following. The Python code is available at: https://github.com/KenSzulczyk

```
# Import libraries
import pandas as pd
import statsmodels.api as sm
from statsmodels.formula.api import ols

# Step 1: Create the dataset
data = {
    "Type 1": [28, 29, 30, 28, 32],
    "Type 2": [32, 28, 32, 29, 31],
    "Type 3": [30, 27, 31, 31, 32]
}

# Create a Pandas dataframe
df = pd.DataFrame(data)

# Step 2: Convert to long format
df_long = df.melt(var_name="Wax", value_name="Durability")

# Step 3: Fit the ANOVA model
model = ols('Durability ~ C(Wax)', data=df_long).fit()

# Step 4: Get ANOVA table
anova_table = sm.stats.anova_lm(model, typ=2)

# Step 5: Print results
print("\n=== ANOVA Table ===")
print(anova_table)
```

The Python output is below:

```
=== ANOVA Table ===
            sum_sq    df         F    PR(>F)
C(Wax)         2.8   2.0  0.428571  0.661029
Residual      39.2  12.0       NaN       NaN
```

The p-value exceeds the significance level of $\alpha = 0.05$. Therefore, we fail to reject the null hypothesis. There is no statistical evidence that the average durability differs across the three wax types.

In Example 2, Lazer Manufacturing investigates whether managers at three plants—Chicago, St. Louis, and Detroit—work different numbers of hours per week. A random sample of five managers is selected from each plant.

Click + Code to create a new code cell, and run the following. The Python code is available at: https://github.com/KenSzulczyk

```
# Import libraries
import pandas as pd
import statsmodels.api as sm
from statsmodels.formula.api import ols

# Step 1: Create the dataset
data = {
    "Chicago": [49,53,58,62,53],
    "StLouis": [74,62,67,74,63],
    "Detroit": [50,64,60,56,55]
}

# Create a Pandas dataframe
df = pd.DataFrame(data)

# Step 2: Convert to long format
df_long = df.melt(var_name="Plant", value_name="Output")

# Step 3: Fit the ANOVA model
model = ols('Output ~ C(Plant)', data=df_long).fit()

# Step 4: Get ANOVA table
anova_table = sm.stats.anova_lm(model, typ=2)

# Step 5: Print results
print("\n=== ANOVA Table ===")
print(anova_table)
```

The Python output is below:

```
=== ANOVA Table ===
            sum_sq    df         F    PR(>F)
C(Plant)     490.0   2.0  8.448276  0.005129
Residual     348.0  12.0       NaN       NaN
```

In this case, the p-value is below $\alpha = 0.05$. We reject the null hypothesis and conclude that at least one plant has a different average number of hours worked. To identify which groups differ, we conduct pairwise comparisons.

Click + Code to create a new code cell, and run the following. The Python code is available at: https://github.com/KenSzulczyk

```
# Step 6: Least Significant Difference (LSD) pairwise comparisons
from statsmodels.stats.multicomp import pairwise_tukeyhsd

# Perform pairwise comparisons (LSD-style interpretation)
```

```
lsd = pairwise_tukeyhsd(endog=df_long["Output"],
                        groups=df_long["Plant"],
                        alpha=0.05)

print("\n=== Pairwise Comparisons (LSD) ===")
print(lsd)
```

The Python output is on the next page:

```
=== Pairwise Comparisons (LSD) ===
 Multiple Comparison of Means - Tukey HSD, FWER=0.05
====================================================
 group1  group2 meandiff p-adj   lower   upper  reject
----------------------------------------------------
Chicago Detroit      2.0 0.8294 -7.0864 11.0864  False
Chicago StLouis     13.0 0.0064  3.9136 22.0864   True
Detroit StLouis     11.0 0.0184  1.9136 20.0864   True
----------------------------------------------------
```

The results indicate that Chicago and Detroit are not significantly different, while the other pairs differ. This highlights an important feature of ANOVA: rejecting the null only tells us that differences exist—it does not identify where those differences occur.

In Example 3, New Oil Company tests three gasoline blends (X, Y, and Z) using the same vehicles. Because the same cars are used across treatments, each car acts as a block, controlling for vehicle-specific effects.

Click + Code to create a new code cell, and run the following. The Python code is available at: https://github.com/KenSzulczyk

```
# Import libraries
import pandas as pd
import statsmodels.api as sm
from statsmodels.formula.api import ols

# Step 1: Create the dataset
data = {
    "Block": [1,2,3,4,5]*3,
    "Treatment": ["X"]*5 + ["Y"]*5 + ["Z"]*5,
    "Output": [
        13.2,12.8,12.3,14.0,11.1,   # Blend X
        12.8,12.3,12.3,13.2,10.6,   # Blend Y
        12.8,12.3,11.9,12.3,11.1    # Blend Z
    ]
}

# Create a Pandas dataframe
df = pd.DataFrame(data)

# Step 2: (Already in long format)
```

```
df_long = df

# Step 3: Fit the ANOVA model (with blocks)
model = ols('Output ~ C(Treatment) + C(Block)', data=df_long).fit()

# Step 4: Get ANOVA table
anova_table = sm.stats.anova_lm(model, typ=1)

# Step 5: Print results
print("\n=== ANOVA Table (With Blocks) ===")
print(anova_table)
```

The Python output is below:

```
=== ANOVA Table (With Blocks) ===
               df    sum_sq   mean_sq          F    PR(>F)
C(Treatment)  2.0  0.965333  0.482667   3.756161  0.070738
C(Block)      4.0  9.180000  2.295000  17.859922  0.000473
Residual      8.0  1.028000  0.128500        NaN       NaN
```

The results depend on the chosen significance level: $\alpha = 0.10$. We reject the null hypothesis and conclude that at least one blend differs. At $\alpha = 0.05$ or 0.01, we fail to reject the null hypothesis.

Importantly, the block variable is statistically significant. This indicates that differences across vehicles affect fuel efficiency. Thus, it validates the use of a randomized block design.

In Example 4, PowerClean Company studies how cleaning performance depends on:

- Detergent type (Standard vs. Premium)
- Water temperature (Cold, Warm, Hot)

Each combination is tested twice. It allows us to estimate interaction effects.

Click + Code to create a new code cell, and run the following. The Python code is available at: https://github.com/KenSzulczyk

```
# Import libraries
import pandas as pd
import statsmodels.api as sm
from statsmodels.formula.api import ols

# Step 1: Create the dataset
data = {
    "Detergent": ["Standard"]*6 + ["Premium"]*6,
    "Temperature": ["Cold","Warm","Hot"]*4,
    "Output": [
        65,72,78, 67,74,80,   # Standard (2 reps)
        66,75,88, 69,77,90    # Premium (2 reps)
    ]
```

```
}

# Create a Pandas dataframe
df = pd.DataFrame(data)

# Step 2: (Already in long format)
df_long = df

# Step 3: Fit the ANOVA model (with interaction)
model = ols('Output ~ C(Detergent) + C(Temperature) + C(Detergent):C(Temperature)',
data=df_long).fit()

# Step 4: Get ANOVA table
anova_table = sm.stats.anova_lm(model, typ=1)

# Step 5: Print results
print("\n=== ANOVA Table (Two-Way with Interaction) ===")
print(anova_table)
```

The Python output is below:

```
=== ANOVA Table (Two-Way with Interaction) ===
                            df      sum_sq     mean_sq           F    PR(>F)
C(Detergent)               1.0   70.083333   70.083333   29.000000  0.001687
C(Temperature)             2.0  597.166667  298.583333  123.551724  0.000013
C(Detergent):C(Temperature)  2.0   41.166667   20.583333    8.517241  0.017673
Residual                   6.0   14.500000    2.416667         NaN       NaN
```

The results show three F-statistics:

- One for detergent
- One for temperature
- One for the interaction

All three are statistically significant. Therefore:

- Detergent type affects cleaning performance
- Water temperature affects cleaning performance
- The interaction effect is also significant.

The interaction result is particularly important. It implies that the effectiveness of a detergent depends on the water temperature. In other words, the impact of one factor varies depending on the level of another factor.

Key Insight

This section illustrates several important ideas:

- Python allows us to perform ANOVA quickly and accurately
- ANOVA tests whether group means differ, but not which ones differ
- Post-hoc tests, such as pairwise comparisons, identify specific differences

- Experimental design matters
- Blocking controls for unwanted variation
- Interaction terms reveal deeper relationships between factors

5.4 Regression in Python

In this section, we introduce linear regression using a simple and intuitive example. In the previous section, we used ANOVA to test for differences across group means. Regression analysis extends this framework by allowing us to examine the relationship between continuous variables.

We consider a dataset that relates the number of hours a student studies to their exam score. Our goal is to estimate whether study time has a measurable effect on performance. We begin by creating the dataset and estimating a simple linear regression model using the statsmodels library.

Click + Code to create a new code cell, and run the following. The Python code is available at: https://github.com/KenSzulczyk

```
# Import libraries
import pandas as pd
import statsmodels.api as sm
from statsmodels.formula.api import ols

# Step 1: Create the dataset
data = {
    "Hours": [2, 4, 6, 8, 10],
    "Score": [65, 67, 82, 85, 84]
}

# Create a Pandas dataframe
df = pd.DataFrame(data)

# Step 2: (Already in correct format)
df_long = df

# Step 3: Fit the regression model
model = ols('Score ~ Hours', data=df_long).fit()

# Step 4: Get ANOVA table
anova_table = sm.stats.anova_lm(model, typ=1)

# Step 5: Print results
print("\n=== ANOVA Table (Regression) ===")
print(anova_table)

print("\n=== Regression Summary ===")
print(model.summary())
```

The Python output is below:

```
=== ANOVA Table (Regression) ===
           df  sum_sq     mean_sq         F    PR(>F)
Hours     1.0   313.6  313.600000  13.91716  0.033562
Residual  3.0    67.6   22.533333       NaN       NaN

=== Regression Summary ===
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Score   R-squared:                       0.823
Model:                            OLS   Adj. R-squared:                  0.764
Method:                 Least Squares   F-statistic:                     13.92
Date:                Fri, 10 Apr 2026   Prob (F-statistic):             0.0336
Time:                        23:55:57   Log-Likelihood:                -13.605
No. Observations:                   5   AIC:                             31.21
Df Residuals:                       3   BIC:                             30.43
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     59.8000      4.979     12.011      0.001      43.956      75.644
Hours          2.8000      0.751      3.731      0.034       0.411       5.189
==============================================================================
Omnibus:                        nan   Durbin-Watson:                   2.243
Prob(Omnibus):                  nan   Jarque-Bera (JB):                0.514
Skew:                         0.243   Prob(JB):                        0.773
Kurtosis:                     1.506   Cond. No.                         15.8
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
/usr/local/lib/python3.12/dist-packages/statsmodels/stats/stattools.py:74:
ValueWarning: omni_normtest is not valid with less than 8 observations; 5 samples were
given.
  warn("omni_normtest is not valid with less than 8 observations; %i "
```

The ANOVA table tests the overall significance of the regression model. Specifically, it evaluates the null hypothesis:

$$H_0: \beta_1 = 0$$

This hypothesis states that study time has no effect on exam scores. The results show that the p-value is below the 5% significance level, so we reject the null hypothesis. Study time is a statistically significant predictor of exam performance.

The regression output provides additional information:

- The intercept is approximately 59.8. This represents the predicted exam score for a student who studies zero hours. While this value may not be meaningful in practice, it is necessary for defining the regression line.
- The slope coefficient is approximately 2.80. This indicates that each additional hour of study increases the expected exam score by about 2.8 points, on average.
- Both coefficients are statistically significant at the 5% level, indicating strong evidence of a relationship between study time and performance.

The model also reports an R^2 value of 0.823. This means that approximately 82.3% of the variation in exam scores is explained by study time. While this suggests a strong relationship, it is important to remember that R^2 alone does not imply causality or model adequacy.

Key Insight

An important insight is that regression and ANOVA are closely related. The ANOVA table decomposes total variation into explained and unexplained components, which corresponds directly to the regression framework.

- Explained variation: variation captured by the regression model
- Unexplained variation: residual or error variation

Thus, regression can be viewed as a natural extension of ANOVA when explanatory variables are continuous rather than categorical.

Limitations of the Simple Model

Although the results are encouraging, this model is highly simplified. Exam performance is influenced by many factors beyond study time, such as:

- Student ability
- Quality of instruction
- Test difficulty
- Study methods

Omitting these variables may lead to omitted variable bias, which can distort the estimated relationship. We expand linear regression in Chapter 6 to include two or more explanatory variables.

5.5 Summary

This chapter marks an important transition from theory to application. We have taken the statistical concepts developed in earlier chapters and implemented them using Python. It transforms manual calculations into efficient, scalable analysis. Along the way, we explored

how to summarize data, visualize distributions, examine relationships, and replicate ANOVA and regression results using real datasets.

The key objective is not simply to learn Python syntax, but to understand how computational tools enhance economic reasoning. Code allows us to work with larger datasets, reduce errors, and focus on interpretation rather than arithmetic. At the same time, the underlying theory remains essential—without it, the output from any program is meaningless.

Students should aim to develop two complementary skills: the ability to write and understand code, and the ability to interpret results within an economic framework. When combined, these skills form the foundation of modern data analysis.

In the next chapter, we extend these ideas to multiple regression, where we analyze more complex relationships and build models that better reflect real-world behavior.

5.6 Exercises

Please use Python to solve all problems.

Problem 1

A researcher wants to examine whether different study environments affect student performance. The experiment includes one factor:

Factor (Study Environment):

- A1 : Quiet Room
- A2 : Coffee Shop
- A3 : Library

The response variable is test score. Each environment is tested with five students.

Please calculate the ANOVA and determine whether the study place affects a student's test scores.

The Python code below contains the appropriate library and data.

```
# Import the required library
from scipy import stats

# ------------------------------
# Step 1: Enter the data
# ------------------------------
```

```
# Test scores for each study environment
quiet_room = [78, 82, 80, 79, 81]
coffee_shop = [81, 79, 83, 80, 78]
library = [80, 81, 79, 82, 78]
```

Problem 2

Four machines are tested. Five operators serve as blocks.

a. Construct an analysis of variance table. What is your conclusion using $\alpha = 0.1$?

b. At the **$\alpha = 0.1$** level of significance, use Fisher's LSD procedure to test for the equality of the means. Please fill in the missing numbers.

c. What is the Type I error for the experiment?

The Python program to import libraries and data is below:

```
# Import libraries
import pandas as pd
import statsmodels.api as sm
from statsmodels.formula.api import ols
from itertools import combinations
from scipy import stats
import numpy as np

# ------------------------------
# Step 1: Enter the data
# ------------------------------

# Create dataset in long format (required for statsmodels)
data = pd.DataFrame({
    'Operator': [1,1,1,1, 2,2,2,2, 3,3,3,3, 4,4,4,4, 5,5,5,5],
    'Machine': ['M1','M2','M3','M4'] * 5,
    'Score': [
        72,80,74,69,
        75,82,76,70,
        73,81,75,71,
        74,83,77,72,
        76,84,78,73
    ]
})
```

Problem 3

A coffee shop chain wants to study how sales performance depends on both drink prices and store ambiance. The company believes that these two factors may jointly influence customer behavior.

The experiment includes the following factors:

Factor A (Price Level):
- A1: Low Price
- A2: High Price

Factor B (Store Ambiance):
- B1: Basic Interior
- B2: Modern Interior

The response variable is daily revenue in hundreds of dollars, where higher values indicate better performance. Each treatment combination is tested twice, providing replication for estimating experimental error.

(a) Please construct an ANOVA with an interaction term between price and environment.

The Python code below contains the libraries and data:

```
# Import libraries
import pandas as pd
import statsmodels.api as sm
from statsmodels.formula.api import ols

# ------------------------------
# Step 1: Enter the data
# ------------------------------

# We enter the data in "long format"
# Each row is one observation

data = pd.DataFrame({
    'Price': [
        'Low', 'Low', 'High', 'High',
        'Low', 'Low', 'High', 'High'
    ],
    'Environment': [
        'Basic', 'Basic', 'Basic', 'Basic',
        'Modern', 'Modern', 'Modern', 'Modern'
    ],
    'Revenue': [
        42, 45, 38, 36,   # Basic
```

```
        55, 58, 60, 62    # Modern
    ]
})
```

Problem 4.

We want to estimate the relationship between coffee demand and its price. We have the following six observations.

a. Plot the data.

b. Please estimate the simple linear regression between coffee demand and coffee price.

c. Calculate the ANOVA for coffee demand. Please evaluate the F statistic. Then comment on the coffee's intercept and slope.

d. Please comment on the R^2.

e. Please plot the standardized residuals. Does it appear that any observation is an outlie?

The Python code for the libraries and data is below:

```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.formula.api import ols

# ------------------------------
# Step 1: Enter the data
# ------------------------------

data = pd.DataFrame({
    'Price': [2, 3, 4, 5, 6, 7],
    'Demand': [92, 79, 77, 69, 63, 52]
})
```

Chapter 6. Multiple Regression

This chapter extends simple linear regression to multiple regression, one of the most widely used tools in statistics, economics, and business analytics. In practice, most outcomes are influenced by several factors at the same time. Multiple regression provides a framework for measuring these effects within a single model [3]. By including several explanatory variables simultaneously, we can build richer and more realistic models of real-world behavior.

We also explore several important challenges that arise in multiple regression, including multicollinearity, outliers, autocorrelation, and heteroscedasticity. In addition, we show how the regression framework can be extended to include dummy variables and experimental design. The chapter concludes by introducing the F-test, which helps us determine whether adding one or more explanatory variables significantly improves the model.

6.1 Model Specification

We now extend our study of regression analysis to situations involving two or more independent variables. In many real-world problems, outcomes are influenced by several factors at the same time. Multiple regression analysis allows us to incorporate these factors into a single model, leading to more accurate and realistic estimates than those obtained from simple linear regression.

The general multiple regression model describes how a dependent variable y is related to several independent variables x_1, x_2,...,x_p and a random error term:

$$y = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \cdots + \beta_p \cdot x_p + \varepsilon$$

where:

- β_0 is the intercept (the expected value of y when all x's are zero)
- β_1, β_2,...,β_p are the slope coefficients
- ε is the error term. It captures all other factors affecting y that are not included in the model.

While the first equation describes an individual observation, we are often more interested in the average (expected) value of y given the independent variables. This relationship is expressed as:

$$E(y) = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \cdots + \beta_p \cdot x_p$$

This equation represents the systematic part of the model—the portion of y that can be explained by the independent variables.

In practice, the true parameters β_0, β_1,...,β_p are unknown. We estimate them using sample data. From a random sample, we compute the estimates b_0, b_1, b_2,...,b_p, which serve as point estimates of the true parameters.

The estimated regression equation is:

$$\hat{y} = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \cdots + \beta_p \cdot x_p$$

where:

- $\hat{y}$ (pronounced "y-hat") is the predicted value of the dependent variable
- each b_j represents the estimated effect of x_j on y

To estimate the coefficients, we use the least squares method, which chooses the values of b_0, b_1, b_2,...,b_p that minimize the total squared difference between the observed values and the predicted values [15].

$$f = min \sum (y_i - \hat{y}_i)^2$$

where:

- f is the objective function to minimize
- y_i is the observed value
- $\hat{y}$ is the predicted value
- $y_i - \hat{y}_i$ is called the residual

The formulas used to compute the regression coefficients b_0, b_1, b_2,...,b_p involve matrix algebra. It can become complex when many variables are included. Rather than performing these calculations by hand, we will use Python to estimate the model efficiently.

Our focus will be on:

- understanding the model
- interpreting the results
- applying regression analysis to real-world problems

Key Insight

Multiple regression allows us to isolate the effect of one variable while holding all other variables constant.

This idea is extremely important. For example, when estimating the demand for coffee, we can measure how coffee price affects consumption while holding income and other prices constant. This helps us uncover true relationships that might otherwise be hidden.

6.2 Case Study: Coffee Date

We now return to the coffee dataset introduced in Section 5.1. This dataset contains annual observations from 1990 to 2023 and includes several economically meaningful variables used to explain coffee consumption.

- Coffee consumption (y): measured as cups per person per year
- Coffee price (x_1): dollars per cup
- Tea price (x_2): dollars per cup
- Sugar price (x_3): dollars per pound
- Annual income (x_4): dollars per household per year

Using these variables, we specify the coffee demand function as:

$$y = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \beta_3 \cdot x_3 + \beta_4 \cdot x_4 + \varepsilon$$

This equation states that coffee consumption depends on its own price, the price of related goods (tea and sugar), and consumer income.

In Chapter 5, we manually entered the data. In practice, data are typically stored in files. Here, we store the dataset in a CSV file and load it into Python.

After uploading the file, create a new code cell and run the following Python code. The Python code is available at: https://github.com/KenSzulczyk

```
# ------------------------------------------------
# Demand Estimation for Coffee using OLS Regression
# ------------------------------------------------

# Step 1: Import libraries
import pandas as pd                      # For data handling
import numpy as np                       # For mathematical functions (e.g., log)
import statsmodels.formula.api as smf    # For regression using formulas

# Step 2: Load the dataset from a CSV file
# Make sure "coffee_demand.csv" is in your working directory
df = pd.read_csv("Chapter_06-coffee_demand.csv")

# Step 3: Display the first few rows of the dataset
# This helps verify that the data loaded correctly
```

```
print(df.head())

# Step 4: Estimate the demand function using OLS
# The formula syntax allows us to write the model like an equation
# Coffee_consumption is the dependent variable (Q)
# The independent variables include prices and income
model = smf.ols(
    formula="Coffee_consumption ~ Coffee_price + Tea_price + Sugar_price +
Annual_income",
    data=df).fit()

# Step 5: Print the regression results
# The summary includes coefficients, p-values, R-squared, and more
print(model.summary())
```

We get the following Python output below:

```
   Year  Coffee_consumption  Coffee_price  Sugar_price  Tea_price  \
0  1990              267.22          0.91         0.13       0.92
1  1991              270.57          0.86         0.09       0.84
2  1992              309.06          0.65         0.09       0.91
3  1993              301.31          0.69         0.10       0.84
4  1994              214.79          1.45         0.12       0.83

   Annual_income
0          42650
1          43240
2          44220
3          47220
4          49340
                            OLS Regression Results
==============================================================================
Dep. Variable:     Coffee_consumption   R-squared:                       0.978
Model:                            OLS   Adj. R-squared:                  0.975
Method:                 Least Squares   F-statistic:                     325.5
Date:                Wed, 08 Apr 2026   Prob (F-statistic):           1.22e-23
Time:                        03:26:01   Log-Likelihood:                -123.74
No. Observations:                  34   AIC:                             257.5
Df Residuals:                      29   BIC:                             265.1
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      257.5593      8.411     30.623      0.000     240.357     274.761
Coffee_price  -123.6743      6.452    -19.169      0.000    -136.870    -110.479
Tea_price       28.3967     10.764      2.638      0.013       6.381      50.412
Sugar_price    -25.3330     65.061     -0.389      0.700    -158.398     107.732
Annual_income    0.0024   8.96e-05     26.381      0.000       0.002       0.003
==============================================================================
Omnibus:                        0.518   Durbin-Watson:                   1.317
Prob(Omnibus):                  0.772   Jarque-Bera (JB):                0.239
Skew:                          -0.205   Prob(JB):                        0.887
Kurtosis:                       2.974   Cond. No.                     3.15e+06
==============================================================================
```

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 3.15e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

The regression output provides several important statistics that help us evaluate the model.

We first test whether the model, as a whole, is useful:

H_0: $\beta_1 = \beta_2 = \beta_3 = \beta_4 = 0$
H_a: At least one slope does not equal zero

The F-statistic is 325.5 with a very small p-value [24]. Since the p-value is less than any common significance level, e.g., 5%, we reject the null hypothesis. Thus, we conclude that at least one independent variable significantly affects coffee consumption. Next, we copied the t-statistics and p-values from the output, which is below:

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	257.5593	8.411	30.623	0.000	240.357	274.761
Coffee_price	-123.6743	6.452	-19.169	0.000	-136.870	-110.479
Tea_price	28.3967	10.764	2.638	0.013	6.381	50.412
Sugar_price	-25.3330	65.061	-0.389	0.700	-158.398	107.732
Annual_income	0.0024	8.96e-05	26.381	0.000	0.002	0.003

Next, we examine each coefficient individually using t-tests. For example, for the intercept:

H_0: $\beta_0 = 0$
H_a: $\beta_0 \neq 0$

The t-statistic is:

$$t = \frac{b_0 - \beta_0}{std\ err} = \frac{257.5593 - 0}{8.411} = 30.623$$

The p-value is written as P > | t |. The p-value is effectively zero, so we reject the null hypothesis. We interpret this as the intercept is statistically different from zero.

The output also reports 95% confidence intervals for each coefficient. These intervals provide a range of plausible values for the true parameter. For the intercept:

- Lower bound: 240.357
- Upper bound: 274.761

This means we are 95% confident that the true population intercept lies within this range.

A Key Idea

If a confidence interval does not include zero, the coefficient is statistically significant at the 5% level.

We can determine statistical significance using:
Large absolute t-statistics
Small p-values
Confidence intervals that do not include zero

In practice, the p-value is the easiest to use.

Using a 5% significance level, we find that:

- Coffee price: statistically significant
- Tea price: statistically significant
- Sugar price: not statistically significant
- Income: statistically significant

Thus, coffee consumption is influenced by coffee price, tea price, and income. The estimated regression equation is:

$$\hat{y} = 257.5593 - 123.6743 \cdot x_1 + 28.3967 \cdot x_2 - 25.3330 \cdot x_3 + 0.0024 \cdot x_4$$

Now we ask an important question: Do the results make economic sense?

> **Coffee price** (–123.6743): When the price of coffee increases, consumption decreases, holding all other variables constant (ceteris paribus). This is consistent with the Law of Demand.
>
> **Tea price** (+28.3967): When tea becomes more expensive, consumers switch to coffee. Tea is a substitute good.
>
> **Sugar price** (not significant): This variable does not have a statistically meaningful effect in this model. However, we still must include it in the model.
>
> **Income** (+0.0024): Higher income increases coffee consumption. Coffee is a normal good.

Each coefficient measures a marginal effect. It means the change in y from a one-unit change in an independent variable. In calculus terms:

$$\frac{\partial y}{\partial x_1} = \beta_1$$

In simpler terms, this is just the slope:

$$\frac{\Delta y}{\Delta x_1} \approx -123.6743$$

We interpret this as follows: If the price of coffee increases by $1, consumption decreases by approximately 123.67 cups per year.

The parameter estimate for the tea price is 28.3967. If the price of tea increases by $1, then people increase their demand for coffee by 28.3967 cups per year.

The parameter estimate for sugar price is not statistically significant. Thus, we skip over it. The last parameter estimate is for annual income. If a customer's annual income rises by $1, then their demand for coffee increases by 0.0024.

We can also use the model to make predictions. Suppose a consumer faces:

- Coffee price = $1.50
- Tea price = $1.25
- Sugar price = $0.15
- Income = $65,000

Then:

$$\hat{y} = 257.5593 - 123.6743 \cdot 1.50 + 28.3967 \cdot 1.25 - 25.3330 \cdot 0.15 + 0.0024 \cdot 65{,}000 = 259.7$$

We predict this individual consumes approximately 259.7 cups of coffee per year.

This example demonstrates how multiple regression allows us to:

- quantify relationships between variables
- test economic theory
- make data-driven predictions

6.3 ANOVA and R^2

The regression output in Section 6.2 reports an F-statistic, which comes directly from analysis of variance (ANOVA). In this section, we connect the classical ANOVA framework to multiple regression and show how it helps us understand model performance.

The key idea is simple: The total variation in the dependent variable can be divided into what the model explains and what it cannot explain.

We decompose the total variation in coffee consumption into three components:

$$SST = SSR + SSE$$

where:

- Sum of Squares Total (SST): total variation in y
- Sum of Squares due to Regression (SSR): variation explained by the model
- Sum of Squares due to Error (SSE): unexplained variation or noise

Suppose we have n observations and k independent variables.

Total degrees of freedom: n – 1
Regression degrees of freedom: k
Error degrees of freedom: n – k – 1

These degrees of freedom are used to scale the variation appropriately.

The total variation is calculated using the sample variance of y. This measures how much coffee consumption varies around its mean.

$$SST = (n-1) \cdot var(y)$$

The error term measures how far the predicted values are from the actual values:

$$residuals = SSE = \sum (y_i - \hat{y}_i)^2$$

Recall that the least squares method chooses coefficients that minimize SSE. This is why regression provides the “best fit” line.

The explained variation is simply:

$$SSR = SST - SSE$$

This represents how much of the variation in coffee consumption is explained by the independent variables.

To compare explained and unexplained variation, we compute mean squares:

$$MSR = \frac{SSR}{k} \quad \text{and} \quad MSE = \frac{SSE}{n-k-1}$$

The F-statistic is then:

$$F = \frac{MSR}{MSE}$$

A large F-statistic means the model explains much more variation relative to noise. This provides evidence that the model is statistically significant.

Although Python does not present this exact "classical" table by default, we can compute it manually with the Python below. The Python code is available at: https://github.com/KenSzulczyk

```
# ------------------------------------------------
# Step 6: Classical ANOVA Table (Manual Construction)
# ------------------------------------------------

# Degrees of freedom
n = len(df)                 # Number of observations
k = model.df_model          # Number of independent variables

df_reg = int(k)             # Regression degrees of freedom
df_err = int(n - k - 1)     # Error degrees of freedom
df_tot = int(n - 1)         # Total degrees of freedom

# Total Sum of Squares (SST)
# Measures total variation in the dependent variable
SST = df_tot * ( df["Coffee_consumption"].var() )

# Sum of Squared Errors (SSE)
# Measures unexplained variation (residuals)
SSE = sum(model.resid**2)

# Regression Sum of Squares (SSR)
# Measures explained variation
SSR = SST - SSE

# Mean Squares
MSR = SSR / df_reg          # Mean square regression
MSE = SSE / df_err          # Mean square error

# F-statistic
F = MSR / MSE

# Create a clean ANOVA table
anova_classic = pd.DataFrame({
    "Source": ["Regression", "Error", "Total"],
    "SS": [SSR, SSE, SST],
```

```
    "df": [df_reg, df_err, df_tot],
    "MS": [MSR, MSE, ""],
    "F": [F, "", ""]
})

print(anova_classic)
```

The Python output is below:

```
       Source             SS  df            MS           F
0  Regression  129555.846987   4  32388.961747  325.507317
1       Error    2885.587639  29     99.503022
2       Total  132441.434626  33
```

From the output, the F-statistic is 325.51, which matches the regression output in Section 6.2.
This confirms that the ANOVA framework and regression output are consistent.

The R^2 statistic measures the proportion of total variation explained by the model:

$$R^2 = \frac{SSR}{SST}$$

Using our data:

$$R^2 = \frac{SSR}{SST} = \frac{129{,}555.85}{132{,}441.43} = 0.978$$

The model explains 97.8% of the variation in coffee consumption.

While R^2 is useful, it has an important limitation: Adding more independent variables will always increase (or at least not decrease) R^2, even if those variables are not meaningful.

This happens because:

- Adding variables reduces SSE (even slightly)
- A smaller SSE increases SSR
- Therefore, R^2 = SSR / SST increases

To address this issue, we use the adjusted R^2, which penalizes unnecessary variables [25]:

$$adj.R^2 = 1 - (1 - R^2)\frac{n-1}{n-k-1}$$

Using our data:

$$adj.R^2 = 1 - (1 - 0.978)\frac{34 - 1}{34 - 4 - 1} = 0.975$$

The adjusted R^2 is slightly lower than R^2, reflecting the penalty for including multiple variables. We see the adjusted R^2 lies below the R^2 in the Python output in the last section.

Interpretation

- If adjusted R^2 increases, the new variable improves the model
- If adjusted R^2 decreases, the variable adds little explanatory power

Key Idea

Adjusted R^2 penalizes unnecessary variables and provides a more reliable measure of model quality when multiple predictors are included.

6.4 Dummy Variables

In many applications, some important factors are qualitative rather than numerical. For example, certain years may experience unusual conditions such as economic crises, policy changes, or structural shifts in the economy. These types of effects cannot be measured directly with numbers. However, we can incorporate them into a regression model using dummy variables.

A dummy variable takes only two possible values:

- 1 → condition is present
- 0 → condition is not present

A typical example is gender:

$$D = \begin{cases} 1 & if\ female \\ 0 & if\ male \end{cases}$$

Another common example is education level:

$$D = \begin{cases} 1 & if\ college\ graduate \\ 0 & otherwise \end{cases}$$

In both cases, the dummy variable captures a difference between two groups.

We now extend our coffee demand model by including a dummy variable to capture the effect of the 2008–2009 financial crisis on coffee consumption.

$$D = \begin{cases} 1 & if\ year\ equals\ 2008\ or\ 2009 \\ 0 & otherwise \end{cases}$$

This variable allows us to test whether coffee consumption behaved differently during the crisis years. We include the dummy (D) variable as an additional explanatory variable:

$$y = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \beta_3 \cdot x_3 + \beta_4 \cdot x_4 + \beta_5 \cdot D + \varepsilon$$

Here, β_5 measures the difference in coffee consumption during crisis years relative to normal years, holding all other variables constant.

We add the dummy variable and estimate the model below. The Python code is available at: https://github.com/KenSzulczyk

```
# Create dummy variable for financial crisis years
df["Crisis_dummy"] = ((df["Year"] >= 2008) & (df["Year"] <= 2009)).astype(int)

# Check the data
print(df[["Year", "Crisis_dummy"]].head(20))

# Estimate regression with dummy variable
model_dummy = smf.ols(
    formula="Coffee_consumption ~ Coffee_price + Tea_price + Sugar_price +
Annual_income + Crisis_dummy",
    data=df
).fit()

print(model_dummy.summary())
```

We omitted the list of the first 20 observations for the crisis variable for brevity. We get the Python output below.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     Coffee_consumption   R-squared:                       0.978
Model:                            OLS   Adj. R-squared:                  0.974
Method:                 Least Squares   F-statistic:                     251.8
Date:                Wed, 08 Apr 2026   Prob (F-statistic):           2.31e-22
Time:                        08:37:38   Log-Likelihood:                -123.72
No. Observations:                  34   AIC:                             259.4
Df Residuals:                      28   BIC:                             268.6
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      257.3422      8.616     29.868      0.000     239.693     274.991
Coffee_price  -123.7726      6.578    -18.817      0.000    -137.247    -110.299
Tea_price       28.9802     11.297      2.565      0.016       5.839      52.122
```

```
Sugar_price     -25.3702     66.162     -0.383      0.704    -160.896     110.156
Annual_income     0.0024    9.2e-05     25.637      0.000       0.002       0.003
Crisis_dummy     -1.6038      7.681     -0.209      0.836     -17.338      14.130
==============================================================================
Omnibus:                        0.609   Durbin-Watson:                   1.291
Prob(Omnibus):                  0.738   Jarque-Bera (JB):                0.307
Skew:                          -0.232   Prob(JB):                        0.858
Kurtosis:                       2.976   Cond. No.                     3.15e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 3.15e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

The coefficient on the dummy variable is: $\beta_5 = -1.6038$. During the crisis years (2008–2009), coffee consumption decreased by approximately 1.6 cups per person per year, holding all other variables constant. However, the p-value is 0.836, which is very large.

Because the p-value is much greater than 0.05, we fail to reject the null hypothesis and conclude that the parameter does not differ from zero. There is no statistical evidence that the financial crisis had a significant effect on coffee consumption.

We can also compare the adjusted R^2:

- Without dummy variable: 0.975
- With dummy variable: 0.974

The adjusted R^2 decreases slightly. It indicates that the dummy variable does not improve the model.

Key Insight

If a categorical variable has k categories, we must include only k – 1 dummy variables in the regression.

Why?

Including all k dummy variables creates perfect multicollinearity. It means one variable can be written as a linear combination of the others. This makes estimation impossible.

For example, if we had three categories:

- Low income
- Middle income
- High income

We would include only two dummy variables, and the omitted category becomes the reference group.

Dummy variables allow us to incorporate qualitative effects into regression models, but they must be used carefully to avoid multicollinearity.

6.5 Experimental Design

Dummy variables allow us to use multiple regression as an alternative approach to analysis of variance (ANOVA) and experimental design. Instead of comparing group means directly, we estimate a regression model and interpret the coefficients to test whether the population means are equal.

In this section, we revisit a problem from Chapter 2 and show how regression and ANOVA lead to the same conclusion.

Lazer Manufacturing wants to determine whether managers at its three plants work different numbers of hours per week. The plants are located in Chicago, St. Louis, and Detroit. The data are summarized in Table 6.1.

Table 6.1. Lazer Manufacturing Study Data

Observation	Plant 1 Chicago	Plant 2 St. Louis	Plant 3 Detroit
1	49	74	50
2	53	62	64
3	58	67	60
4	62	74	56
5	53	63	55
Sample Mean	55	68	57
Sample Variance	25.5	33.5	28

At first glance, the sample means suggest that managers in St. Louis work more hours than those in Chicago and Detroit. However, we need a formal statistical test to determine whether these differences are statistically significant.

We can convert this experimental design problem into a regression model by introducing dummy variables.

We define two dummy variables:

- A = 1 if the observation is from St. Louis, 0 otherwise
- B = 1 if the observation is from Detroit, 0 otherwise
- Chicago is the baseline group, where A = 0 and B = 0.

Thus, we have:

- Chicago: A = 0, B = 0
- St. Louis: A = 1, B = 0
- Detroit: A = 0, B = 1

Our regression model is:

$$y = \beta_0 + \beta_1 \cdot A + \beta_2 \cdot B + \varepsilon$$

This model is powerful because each coefficient has a clear interpretation. Chicago is the baseline when A = 0 and B = 0:

$$E(y) = \beta_0$$

The intercept β_0 equals the mean number of hours worked in Chicago. St. Louis is when A = 1 and B = 0:

$$E(y) = \beta_0 + \beta_1$$

The coefficient β_1 measures how much St. Louis differs from Chicago. Detroit is when A = 0 and B = 1:

$$E(y) = \beta_0 + \beta_2$$

The coefficient β_2 measures how much Detroit differs from Chicago. In other words, regression does not estimate each mean separately. Instead, it estimates one mean (Chicago) and then measures how the other groups differ from it.

The following Python program estimates this regression model and computes the sample means. The Python code is available at: https://github.com/KenSzulczyk

```
# ------------------------------------------------
# Regression with Dummy Variables (Experimental Design)
# ------------------------------------------------

# Import required libraries
import pandas as pd
import statsmodels.api as sm

# ------------------------------------------------
# STEP 1: Create the dataset directly in Python
# (No CSV needed since the dataset is small)
# ------------------------------------------------
```

```
data = {
    # Observation number (just for reference)
    "Obs": list(range(1, 16)),

    # Categorical variable: City
    "City": ["Chicago"]*5 + ["St. Louis"]*5 + ["Detroit"]*5,

    # Dummy variable A:
    # A = 1 if St. Louis, 0 otherwise
    "A": [0,0,0,0,0, 1,1,1,1,1, 0,0,0,0,0],

    # Dummy variable B:
    # B = 1 if Detroit, 0 otherwise
    "B": [0,0,0,0,0, 0,0,0,0,0, 1,1,1,1,1],

    # Dependent variable (response)
    "y": [49,53,58,62,53, 74,62,67,74,63, 50,64,60,56,55]
}

# Convert dictionary into a pandas DataFrame
df = pd.DataFrame(data)

# Display the dataset
print("Dataset:")
print(df)

# ------------------------------------------------
# STEP 2: Calculate the mean for each city
# (This is equivalent to group means in ANOVA)
# ------------------------------------------------

city_means = df.groupby("City")["y"].mean()

print("\nCity Means:")
print(city_means)

# Expected results:
# Chicago ≈ 55
# St. Louis ≈ 68
# Detroit ≈ 57

# ------------------------------------------------
# STEP 3: Set up regression model
# y = β0 + β1*A + β2*B + ε
# ------------------------------------------------

# Independent variables (dummy variables)
X = df[["A", "B"]]

# Add a constant (intercept term β0)
# This represents the baseline group (Chicago)
X = sm.add_constant(X)
```

```
# Dependent variable
y = df["y"]

# ------------------------------------------------
# STEP 4: Estimate the regression model and display results
# ------------------------------------------------

model = sm.OLS(y, X).fit()

print("\nRegression Results:")
print(model.summary())
```

The Python output is below:

```
City Means:
City
Chicago      55.0
Detroit      57.0
St. Louis    68.0
Name: y, dtype: float64

Regression Results:
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.585
Model:                            OLS   Adj. R-squared:                  0.516
Method:                 Least Squares   F-statistic:                     8.448
Date:                Fri, 10 Apr 2026   Prob (F-statistic):            0.00513
Time:                        01:57:17   Log-Likelihood:                -44.865
No. Observations:                  15   AIC:                             95.73
Df Residuals:                      12   BIC:                             97.85
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         55.0000      2.408     22.838      0.000      49.753      60.247
A             13.0000      3.406      3.817      0.002       5.579      20.421
B              2.0000      3.406      0.587      0.568      -5.421       9.421
==============================================================================
Omnibus:                        3.332   Durbin-Watson:                   2.224
Prob(Omnibus):                  0.189   Jarque-Bera (JB):                1.213
Skew:                           0.147   Prob(JB):                        0.545
Kurtosis:                       1.638   Cond. No.                         3.73
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

From the output:

- Intercept ($\beta_0 = 55$): The mean for Chicago
- $\beta_1 = 13$: St. Louis managers work 13 more hours than Chicago managers

- $\beta_2 = 2$: Detroit managers work 2 more hours than Chicago managers

We now test the hypothesis:

H_0: $\beta_1 = \beta_2 = 0$ (all group means are equal)
H_1: At least one mean is different

From the regression output:
- F-statistic = 8.448
- p-value = 0.005

Table 6.2 reproduces the ANOVA results from Chapter 2.

Table 6.2. ANOVA for Lazer Manufacturing

ANOVA

Source	Df	SS	MS	F	p-value
SSTR	2	490.00	245.00	8.45	0.005
SSE	12	348.00	29.00		
SST	14	838.00			

Notice that:
- The F-statistic is identical in both methods
- The p-value is identical
- Both approaches lead to the same conclusion

After rejecting the overall F-test, we often want to compare individual groups.
- The t-tests in regression compare each group to the baseline (Chicago)
- The Least Significant Difference (LSD) method compares all pairs of means

Thus:
- Regression answers: How does each group differ from the baseline?
- LSD answers: Which specific pairs of groups differ?

These are related but not identical comparisons.

6.6 Transformations

In many real-world applications, the relationship between the dependent variable and the independent variables is not linear. When this happens, a simple linear regression may provide a poor fit. To address this issue, we can transform the data so that the relationship becomes more linear and easier to model.

Transformations are especially useful when:

- The data show curvature
- The effect of a variable changes at different levels
- The variance of the errors is not constant

Suppose a company tracks how many days employees spend training on a software system and then measures their proficiency.

- At the beginning, employees learn quickly, and their scores increase sharply
- After more training, improvements continue, but at a slower rate

This pattern is known as diminishing marginal returns: each additional day of training contributes less to performance than the previous one.

We estimate two models:

- A standard linear model
- A transformed model using the natural logarithm of Days

The Python code is below for linear and log transformion. The Python code is available at: https://github.com/KenSzulczyk

```
# ------------------------------------------------------
# Linear Regression and Log(X) Regression Example
# Relationship: Days (x) and Score (y)
# ------------------------------------------------------

# Import required libraries
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt

# ------------------------------------------------------
# STEP 1: Manually create the dataset
# ------------------------------------------------------

data = {
    "Obs": list(range(1, 21)),

    # Independent variable (Days)
    "Days": [6,7,9,11,12,19,22,33,40,41,51,53,56,65,76,85,91,104,106,111],

    # Dependent variable (Score)
    "Score":
[67,77,83,130,112,189,162,233,233,265,235,276,301,301,295,367,315,315,296,322]
}

# Convert into DataFrame
df = pd.DataFrame(data)
```

```
# Display dataset
print("Dataset:")
print(df)

# ------------------------------------------------------------
# STEP 2: Linear regression
# Model: y = β0 + β1*x + ε
# ------------------------------------------------------------

X_linear = sm.add_constant(df["Days"])
y = df["Score"]

model_linear = sm.OLS(y, X_linear).fit()

print("\nLinear Regression Results:")
print(model_linear.summary())

# ------------------------------------------------------------
# STEP 3: Log(X) regression
# Model: y = β0 + β1*ln(x) + ε
# This produces a concave (downward-bending) curve
# ------------------------------------------------------------

# Take natural log of Days
df["ln_Days"] = np.log(df["Days"])

X_logx = sm.add_constant(df["ln_Days"])

model_logx = sm.OLS(y, X_logx).fit()

print("\nLog(X) Regression Results:")
print(model_logx.summary())

# ------------------------------------------------------------
# STEP 4: Generate predictions for plotting
# ------------------------------------------------------------

# Sort data for smooth curves
df_sorted = df.sort_values(by="Days")

# Linear predictions
X_linear_sorted = sm.add_constant(df_sorted["Days"])
y_pred_linear = model_linear.predict(X_linear_sorted)

# Log(X) predictions
X_logx_sorted = sm.add_constant(np.log(df_sorted["Days"]))
y_pred_logx = model_logx.predict(X_logx_sorted)

# ------------------------------------------------------------
# STEP 5: Plot data and fitted models
# ------------------------------------------------------------
```

```
plt.figure()

# Scatter plot of actual data
plt.scatter(df["Days"], df["Score"], color="cornflowerblue")

# Linear regression line
plt.plot(df_sorted["Days"], y_pred_linear, color="darkred")

# Log(X) regression line (concave)
plt.plot(df_sorted["Days"], y_pred_logx, color="darkorange")

# Labels and title
plt.xlabel("Days")
plt.ylabel("Score")
plt.title("Linear vs Log(X) Regression")

# Legend
plt.legend(["Data","Linear Fit", "Log(X) Fit"])

# Display plot
plt.show()
```

The Python output is below:

```
Linear Regression Results:
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Score   R-squared:                       0.793
Model:                            OLS   Adj. R-squared:                  0.781
Method:                 Least Squares   F-statistic:                     68.94
Date:                Fri, 10 Apr 2026   Prob (F-statistic):           1.44e-07
Time:                        02:31:36   Log-Likelihood:                -102.84
No. Observations:                  20   AIC:                             209.7
Df Residuals:                      18   BIC:                             211.7
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        112.6651     17.044      6.610      0.000      76.857     148.473
Days           2.3253      0.280      8.303      0.000       1.737       2.914
==============================================================================
Omnibus:                        6.101   Durbin-Watson:                   0.726
Prob(Omnibus):                  0.047   Jarque-Bera (JB):                1.688
Skew:                          -0.084   Prob(JB):                        0.430
Kurtosis:                       1.587   Cond. No.                         106.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.

Log(X) Regression Results:
                            OLS Regression Results
==============================================================================
```

```
Dep. Variable:                  Score   R-squared:                       0.946
Model:                            OLS   Adj. R-squared:                  0.943
Method:                 Least Squares   F-statistic:                     313.3
Date:                Fri, 10 Apr 2026   Prob (F-statistic):           7.84e-13
Time:                        02:31:36   Log-Likelihood:                -89.462
No. Observations:                  20   AIC:                             182.9
Df Residuals:                      18   BIC:                             184.9
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       -106.5154     19.586     -5.438      0.000    -147.664     -65.366
ln_Days       94.2204      5.323     17.701      0.000      83.037     105.403
==============================================================================
Omnibus:                        2.309   Durbin-Watson:                   2.237
Prob(Omnibus):                  0.315   Jarque-Bera (JB):                1.097
Skew:                           0.557   Prob(JB):                        0.578
Kurtosis:                       3.273   Cond. No.                         15.4
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

We write the linear model below:

$$y = \beta_0 + \beta_1 \cdot x + \varepsilon$$

The linear regression produces:

- Adjusted $R^2 = 0.781$

Although the model is statistically significant, the fit is not ideal. When we examine the graph in Figure 6.1:

- The model overestimates scores at low values of Days
- It underestimates in the middle range
- It again overestimates at high values

This pattern indicates a systematic error, suggesting the true relationship is not linear. The log-transformed model is specified below:

$$y = \beta_0 + \beta_1 \cdot ln(x) + \varepsilon$$

Taking the natural logarithm of Days changes the shape of the relationship:

- The log transformation compresses large values of x
- It spreads out smaller values
- The result is a concave (downward-bending) curve

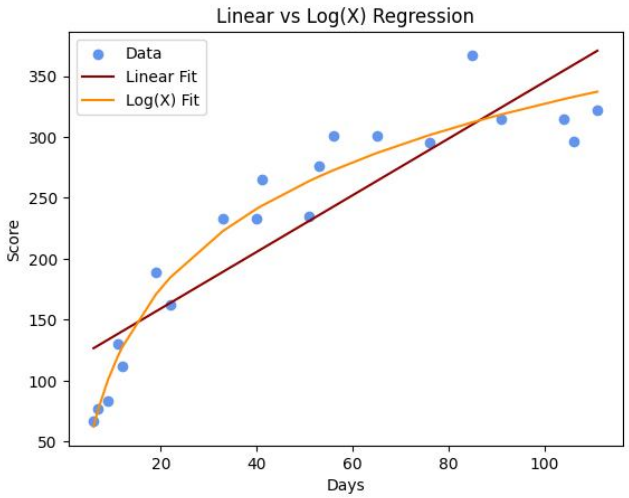


Figure 6.1. Training Days vs. Proficiency

This matches our intuition about diminishing returns.

The regression results show:

- Adjusted $R^2 = 0.943$

This is a substantial improvement over the linear model.

In Figure 6.1:

- The linear model appears as a straight line that misses the curvature in the data
- The log(X) model follows the data more closely, especially at the beginning and end

The transformation allows a simple regression model to capture a nonlinear relationship.

In this example, the dependent variable (Score) is the same in both models. Therefore, we can directly compare the adjusted R^2 values. A higher adjusted R^2 indicates a better fit after accounting for the number of predictors. Since the adjusted R^2 increases from 0.781 to 0.943, the transformed model provides a much better explanation of the data.

Transformations are not limited to logarithms. We can modify the model in several ways to capture more complex relationships. Sometimes, the effect of one variable depends on another variable. This is called an interaction effect and is shown in the regression below:

$$y = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \beta_3 \cdot x_1 \cdot x_2 + \varepsilon$$

The term $x_1 \cdot x_2$ captures how the variables work together. It is common in experimental design and factorial studies. We introduced interaction effects earlier in Chapter 2.

Some relationships follow curved patterns that can be modeled using powers of a variable. For example, a cost function may behave as follows:

- Costs increase slowly at low production levels
- Reach a turning point
- Then increase rapidly at high production levels

A cubic model captures this pattern:

$$y = \beta_0 + \beta_1 \cdot q + \beta_2 \cdot q^2 + \beta_3 \cdot q^3 + \varepsilon$$

Even though the model is nonlinear in q, it is still linear in the parameters (β's). Thus, we can estimate it using least squares.

We can also transform the dependent variable to address issues such as nonconstant variance (heteroskedasticity). Common transformations include:

- Logarithm: $ln(y)$: Useful for stabilizing variance and modeling percentage changes
- Reciprocal: $1/y$: Useful when large values dominate the data
- Square root: $\sqrt{y}$: Often used for count data or moderate skewness

These transformations compress large values and can make the variance more uniform.

- We can apply logarithmic transformations using either the base-10 (common log) or the base e = 2.71828... (natural log)
- We take the reciprocal, 1/y, instead of y
- The square root of y also compresses y nonlinearly.

Key Insight

Transformations allow us to use linear regression to model nonlinear relationships.

- We can transform independent variables to capture curvature
- We can transform the dependent variable to stabilize variance
- As long as the model is linear in the parameters, we can estimate it using ordinary least squares

6.7 Model Selection

In regression analysis, we often face an important question: Should we include or exclude certain variables from the model? Earlier, we used adjusted R^2 to evaluate whether adding a variable improves the model's fit. However, adjusted R^2 alone does not tell us whether the improvement is statistically significant. To formally test this, we use an F-test.

We begin with the demand model for coffee specified earlier in the chapter. This is called the full model because it includes all variables suggested by economic theory:

$$y = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \beta_3 \cdot x_3 + \beta_4 \cdot x_4 + \varepsilon$$

where:

- x_1: Coffee price
- x_2: Tea price
- x_3: Sugar price
- x_4: Income

From the regression results, we observed that sugar price is not statistically significant, and we suspect that tea price may also contribute little to explaining coffee demand.

In practice, we often prefer a parsimonious model. It means a model that explains the data well using as few variables as possible. To test whether tea and sugar prices are necessary, we specify a reduced model that excludes these variables:

$$y = \beta_0 + \beta_1 \cdot x_1 + \beta_4 \cdot x_4 + \varepsilon$$

We want to test whether tea and sugar prices are jointly significant. The hypotheses are below:

H_0: $\beta_2 = \beta_3 = 0$
H_a: At least one slope does not equal zero.

This is a joint hypothesis test, which cannot be answered using individual t-tests alone. We construct an F-statistic by comparing the reduced model and the full model.

$$F = \frac{[SSE(reduced) - SSE(full)]/p}{MSE(full)}$$

where:

- SSE(reduced): error sum of squares from the reduced model
- SSE(full): error sum of squares from the full model
- p: number of restrictions (here, p = 2)
- MSE(full): mean squared error from the full model

Important:

Adding variables to a regression always reduces SSE. Therefore, the reduced model will have a larger SSE. The F-statistic measures whether that increase is large enough to be statistically significant.

The program below performs the partial F-test using nested models. The Python code is available at: https://github.com/KenSzulczyk

```
# ------------------------------------------------
# Demand Estimation for Coffee using OLS Regression
# ------------------------------------------------

# Step 1: Import libraries
import pandas as pd                      # For data handling
import numpy as np                       # For mathematical functions (e.g., log)
import statsmodels.formula.api as smf    # For regression using formulas

# Step 2: Load the dataset from a CSV file
# Make sure "coffee_demand.csv" is in your working directory
df = pd.read_csv("Chapter_06-coffee_demand.csv")

# Step 3: Estimate the demand function using OLS
# The formula syntax allows us to write the model like an equation
# Coffee_consumption is the dependent variable (Q)
# The independent variables include prices and income
model_full = smf.ols(
    formula="Coffee_consumption ~ Coffee_price + Tea_price + Sugar_price +
Annual_income",
    data=df).fit()

# ------------------------------------------------
# Step 4: Test whether Tea_price and Sugar_price can be removed
# (Partial F-test using nested models)
# ------------------------------------------------

# Estimate the reduced model (omit Tea_price and Sugar_price)
model_reduced = smf.ols(
    formula="Coffee_consumption ~ Coffee_price + Annual_income",
    data=df).fit()

# Compare full model vs reduced model
# This performs the joint F-test:
# H0: β_Tea = β_Sugar = 0
```

```
f_test = model_full.compare_f_test(model_reduced)

# Display results
print("\nPartial F-test (Joint Significance Test):")
print("F-statistic:", f_test[0])
print("p-value:", f_test[1])
print("Degrees of freedom:", f_test[2])
```

The output is below:

```
Partial F-test (Joint Significance Test):
F-statistic: 3.7946421163334803
p-value: 0.03436680979653035
Degrees of freedom: 2.0
```

Joint Test (Tea and Sugar Prices)

- F-statistic = 3.795
- p-value = 0.034

Since the p-value is less than 0.05, we reject the null hypothesis at the 5% significance level. Thus, we conclude that at least one of the variables (tea price or sugar price) contributes to the regression. Therefore, we should not remove both variables at the same time.

We next test whether sugar price alone contributes to the model. The Python output is below:

```
Partial F-test (Joint Significance Test):
F-statistic: 0.15160960053572128
p-value: 0.6998465798468301
Degrees of freedom: 1.0
```

Sugar price

- F-statistic = 0.152
- p-value = 0.700

Since the p-value is much greater than 0.05, we fail to reject the null hypothesis. Thus, we conclude that the sugar price does not significantly contribute to the regression and may be removed.

Key Insight

Different fields take different approaches when deciding whether to keep or remove variables: In statistics and machine learning, researchers often favor parsimonious models. Researchers remove insignificant variables to improve simplicity and predictive

performance. In economics, researchers typically emphasizes theory-driven models. Important variables are often retained even if they are not statistically significant

6.8 Multicollinearity

Multicollinearity refers to the presence of correlation among the independent variables in a regression model. In other words, two or more explanatory variables move closely together. Multicollinearity makes it difficult to separate their individual effects on the dependent variable.

As a general guideline, multicollinearity may be considered problematic when the absolute correlation (corr) between two independent variables is high (for example, $|\text{corr}| > 0.7$).

Consider the regression model:

$$y = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \varepsilon$$

Suppose x_1 and x_2 are highly correlated. This means they contain very similar information. As a result, the regression model struggles to determine how much of the change in y is due to x_1 and how much is due to x_2. In simple terms, the model cannot clearly distinguish their separate effects.

Multicollinearity does not violate the assumptions of regression. However, it creates several practical problems:

- Unstable coefficient estimates: Small changes in the data can lead to large changes in the estimated coefficients.
- Large standard errors: This reduces the precision of the estimates.
- Insignificant t-tests: Variables that should be important may appear statistically insignificant.
- Incorrect signs: Coefficients may have unexpected signs (e.g., positive instead of negative).

A common symptom is: The overall model is statistically significant (high F-statistic), but individual variables are not.

For example, we include both income and total spending in a regression. These variables are likely highly correlated. The model cannot determine whether changes in demand come from income or spending. They both move together.

If the goal is prediction, multicollinearity is usually not a serious problem. The model can still predict y accurately. We do not need to interpret individual coefficients. Prediction depends on the combined effect of the variables, not their individual contributions.

If the goal is interpretation, multicollinearity becomes a serious issue. In many economic applications, we want to answer questions such as:

- What is the effect of price on demand?
- What is the effect of income on consumption?

To answer these questions, we must estimate each coefficient reliably. When variables are highly correlated, this becomes difficult.

Several approaches can help reduce multicollinearity:

- Remove one of the correlated variables
- Combine variables into a composite index
- Collect more data, if possible
- Re-specify the model based on economic theory

In practice, economic reasoning should guide which variables to keep.

Key Insight

Multicollinearity does not significantly affect prediction. However, it weakens the interpretation of individual coefficients.

6.9 Residual Analysis

In regression analysis, we make several important assumptions about the error term, ε. These assumptions ensure that our estimates and statistical tests are valid. The key assumptions are:

- The error term has a mean of zero
- The error term has constant variance (homoscedasticity)
- The error terms are independent of one another
- The error term is normally distributed, reflecting random deviations between observed and predicted values

If these assumptions are violated, the regression results—especially hypothesis tests—may be misleading [13].

Residuals (e) are the estimated errors:

$$e = y - \hat{y}$$

They provide valuable information about whether the assumptions of the regression model are satisfied.

In simple linear regression, plotting residuals against x or against $\hat{y}$ provides similar information. However, in multiple regression, there are several explanatory variables, so it is more useful to plot residuals against the fitted values ($\hat{y}$).

Standardized residuals are commonly used to identify unusual observations. These residuals are scaled by their estimated standard deviation. They allow us to compare them across observations.

$$standardized\ residual = \frac{e}{standard\ deviation\ of\ e}$$

As a rule of thumb:

- Most standardized residuals should lie between -2 and $+2$
- Values outside this range may indicate outliers

We use the following program to compute and plot both regular and standardized residuals. The Python code is available at: https://github.com/KenSzulczyk

```
# ------------------------------------------------
# Demand Estimation for Coffee using OLS Regression
# ------------------------------------------------

# Step 1: Import libraries
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import matplotlib.pyplot as plt

# Step 2: Load the dataset
df = pd.read_csv("Chapter_06-coffee_demand.csv")

# Step 3: Display the first few rows
print(df.head())

# Step 4: Estimate the demand function using OLS
model = smf.ols(
    formula="Coffee_consumption ~ Coffee_price + Tea_price + Sugar_price +
Annual_income",
    data=df).fit()

# Step 5: Print regression results
print(model.summary())

# ------------------------------------------------
```

```
# Step 6: Extract residuals
# ------------------------------------------------

# Regular residuals
df["residuals"] = model.resid

# Standardized residuals
influence = model.get_influence()
df["standardized_residuals"] = influence.resid_studentized_internal

# Display first few values
print("\nResiduals:")
print(df[["residuals", "standardized_residuals"]].head())

# ------------------------------------------------
# Step 7: Plot Regular Residuals
# ------------------------------------------------

plt.figure()

# Scatter plot: fitted values vs residuals
plt.scatter(df["Year"], df["residuals"])

# Horizontal reference line at zero
plt.axhline(y=0, color="darkred")

# Labels and title
plt.xlabel("Year")
plt.ylabel("Residuals")
plt.title("Residuals vs Fitted Values")

plt.show()

# ------------------------------------------------
# Step 8: Plot Standardized Residuals
# ------------------------------------------------

plt.figure()

# Scatter plot: fitted values vs standardized residuals
plt.scatter(df['Year'], df["standardized_residuals"])

# Reference lines
plt.axhline(y=0, color="cornflowerblue")
plt.axhline(y=2, color="darkred")
plt.axhline(y=-2, color="darkred")

# Labels and title
plt.xlabel("Year")
plt.ylabel("Standardized Residuals")
plt.title("Standardized Residals vs Fitted Values")

plt.show()
```

The Python output is Figure 6.2. This plot shows the standardized residuals. We observe that two observations near the upper boundary. One observation well below the lower boundary. Thus, we potentially have three outliers.

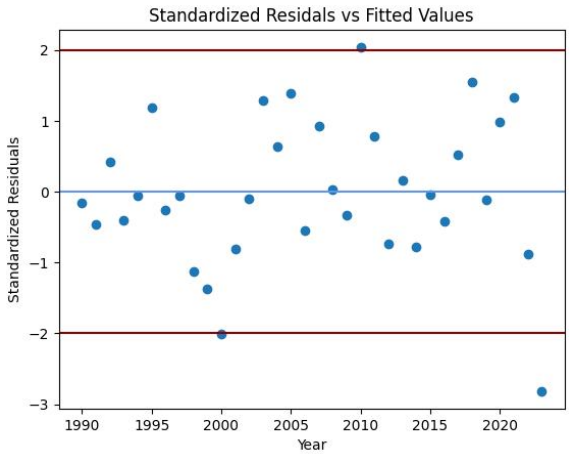


Figure 6.2. Standardized Residuals

Ideally, residuals should behave like random noise:

- No clear pattern
- Constant spread
- Centered around zero

If the residuals display patterns, this indicates problems with the model.

In many business and economic applications, data are collected over time. In such cases, the error in one period may be related to the error in the previous period. This is called autocorrelation or serial correlation.

We can represent this as:

$$\varepsilon_t = \rho \cdot \varepsilon_{t-1} + \omega$$

where:

- ρ is the correlation between consecutive errors, ε_t and ε_{t-1}
- ω is random noise

Types of autocorrelation

- Positive autocorrelation ($\rho > 0$): Residuals tend to have the same sign over time, such as positive followed by positive, or negative followed by negative.

- Negative autocorrelation ($\rho < 0$): Residuals alternate signs. A residual is positive, then negative, positive again, and then negative again.

Autocorrelation violates the assumption that errors are independent. Why does autocorrelation matter?

- Coefficient estimates remain unbiased
- However, standard errors are incorrect
- This leads to unreliable t-tests and p-values

The Durbin-Watson (DW) statistic is used to detect first-order autocorrelation:

$$DW = \frac{\sum(e_t - e_{t-1})^2}{\sum(e_t)^2} = \frac{\sum(e_t - e_{t-1})^2}{SSE}$$

The statistic ranges from 0 to 4:

- DW ≈ 2 → No evidence of autocorrelation
- DW < 2 → Positive autocorrelation
- DW > 2 → Negative autocorrelation

In practice:

- A value near 1 suggests positive autocorrelation
- A value near 3 suggests negative autocorrelation

In our coffee example, the DW statistic is close to 1. It indicates possible positive autocorrelation. Although statistical tables exist, they often include inconclusive regions, making interpretation less straightforward. We plot the residuals for the coffee example in Figure 6.3. Positive autocorrelation is not apparent although the DW test indicates otherwise.

Another common issue is heteroscedasticity, where the variance of the error term is not constant. This often occurs in cross-sectional data. For example, higher-income individuals tend to have more variability in spending. lower-income individuals have more constrained spending. This leads to a pattern where the spread of residuals increases with income. Figure 6.4 shows that the residuals are randomly scattered. The assumption is satisfied. Figure 6.5 shows that the residuals form a funnel shape. Thus, heteroscedasticity is present.

Key Insight

Residual analysis is essential for validating a regression model.

- Residuals should behave like random noise

- Patterns indicate violations of assumptions
- Statistical tests and plots help identify these problems

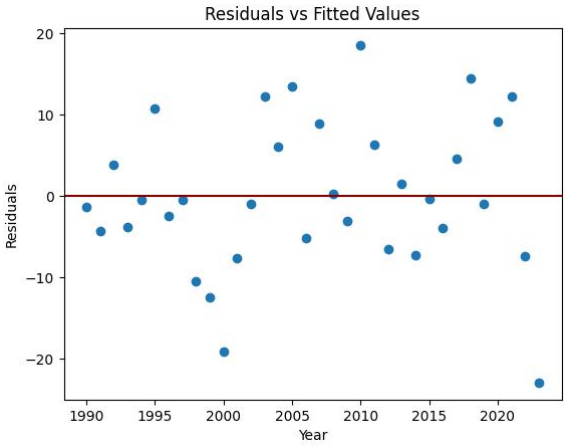


Figure 6.3. Residuals Plots

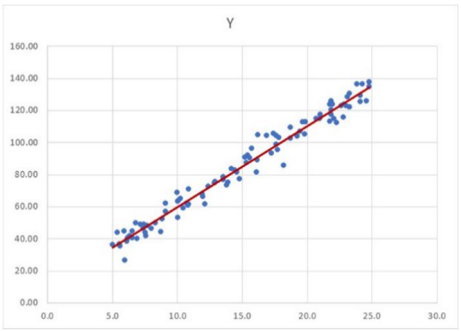


Figure 6.4. Random Residuals

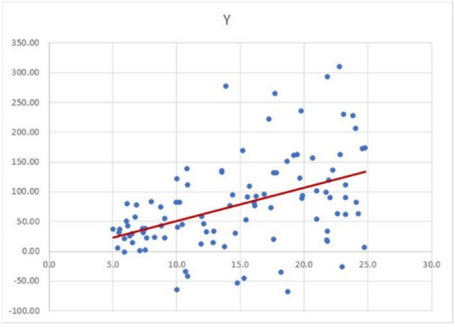


Figure 6.5. Heteroscedastic Residuals

6.10 Summary

In this chapter, we moved beyond simple linear regression and developed the framework of multiple regression, a powerful tool for analyzing relationships in a more realistic setting. Most real-world phenomena are influenced by several factors at the same time, and multiple regression allows us to isolate and measure these effects within a single model.

We began by learning how to interpret coefficients when more than one explanatory variable is present. This requires a shift in thinking: each coefficient now measures the effect of a variable holding all other variables constant. This idea is central to regression analysis and is especially important in fields such as economics and business, where many variables interact simultaneously.

We then examined how to evaluate and improve our models. Adjusted R^2 and the F-test provide tools for determining whether adding variables improves explanatory power. At the same time, we emphasized the importance of parsimony—building models that are as simple as possible while still capturing the essential relationships in the data.

The chapter also introduced several challenges that arise in practice, including multicollinearity, heteroscedasticity, autocorrelation, and outliers. These issues remind us that regression is not just about estimating equations—it is about carefully diagnosing and understanding the data. A good model is not only statistically significant but also meaningful and reliable.

We extended the regression framework further by incorporating dummy variables. They allow us to analyze qualitative factors and connect regression with experimental design and ANOVA. This highlights a key insight: many statistical methods are closely related and can be understood within a unified framework.

Finally, we explored how transformations can improve model fit and how nested model comparisons can guide decisions about adding or removing variables. These tools are essential for building models that are both accurate and interpretable.

6.11 Exercises

Problem 1.

We want to estimate how different factors affect house prices. We collect data on:

- Price (in \$1000s) → dependent variable
- Size (square meters)
- Bedrooms (number of bedrooms)

- Age (years since construction)

Please estimate the regression below and evaluate the results:

$$Price = \beta_0 + \beta_1 \cdot Size + \beta_2 \cdot Bedrooms + \beta_3 \cdot Age + \varepsilon$$

We entered the data manually into Python.

```
# ------------------------------------------------
# Multiple Regression Example: Housing Prices
# (Expanded Dataset)
# ------------------------------------------------

# Import libraries
import pandas as pd
import statsmodels.api as sm

# Create dataset manually
data = {
    "Size": [50,60,80,100,120,140,160,180,70,90,110,130,150,170,200],
    "Bedrooms": [1,2,3,3,4,4,5,5,2,3,3,4,4,5,6],
    "Age": [20,15,10,5,8,3,2,1,12,9,7,6,4,2,1],
    "Price": [120,150,200,250,280,320,360,400,170,220,260,300,340,380,420]
}

# Convert to DataFrame
df = pd.DataFrame(data)

# Display dataset
print(df)
```

Problem 2.

Write a Python program to complete the following tasks:

a. Import the dataset Chapter_06-cubic_cost_function.csv into Python and display the first few observations.

b. Create a scatter plot showing the relationship between total cost (TC) and quantity produced (q). Briefly describe the pattern we observe.

c. Generate two additional variables:

- q^2 (quantity squared)
- q^3 (quantity cubed)

Then compute and display the correlation matrix for q, q^2, q^3, and TC.

d. Estimate the following cubic cost function using ordinary least squares (OLS):

$$TC = \beta_0 + \beta_1 \cdot q + \beta_2 \cdot q^2 + \beta_3 \cdot q^3 + \varepsilon$$

e. Interpret the regression results. In our answer, comment on:

- The statistical significance of the coefficients
- The overall fit of the model
- Whether the estimated relationship is consistent with a typical total cost function

Chapter 7. Multivariate Methods

In many real-world problems, we are interested in more than just a single outcome. Businesses track multiple performance measures. Researchers observe several related responses, and decision-makers must consider how variables interact rather than operate in isolation. This chapter introduces a set of statistical tools designed to handle this complexity by analyzing multiple variables simultaneously and uncovering deeper patterns in the data [26, 27].

We begin with Multivariate Analysis of Variance (MANOVA), which extends ANOVA by allowing us to test for differences across groups using multiple dependent variables at once. We then introduce Analysis of Covariance (ANCOVA), which combines regression and ANOVA to control for additional variables that may influence the outcome. Together, these methods help us make more accurate and meaningful comparisons across groups.

Next, we explore techniques focused on classification and structure. Discriminant analysis helps us classify observations into predefined groups based on their characteristics. On the other hand, cluster analysis identifies natural groupings in the data without prior labels. These methods are widely used in fields such as marketing, finance, and machine learning to segment customers and detect patterns.

Finally, we study logistic regression, a powerful tool for modeling binary outcomes such as success or failure, purchase decisions, or disease occurrence. Logistic regression allows us to estimate probabilities and classify observations while interpreting the influence of explanatory variables.

By the end of this chapter, we will understand how to analyze complex datasets with multiple variables, classify observations, and uncover meaningful relationships that go beyond simple univariate methods.

7.1 Multivariate Analysis of Variance (MANOVA)

Multivariate Analysis of Variance (MANOVA) extends ANOVA by allowing researchers to analyze multiple dependent variables simultaneously. Rather than conducting separate ANOVAs for each outcome, MANOVA evaluates whether groups differ across a combination of dependent variables, taking into account the relationships among them [26].

The independent variables in a MANOVA can be between-subjects, within-subjects, or a combination of both. When a design includes both types of factors, it is referred to as a mixed design. In a mixed design, some factors compare different groups of participants

(between-subjects), while others involve repeated measurements on the same participants (within-subjects).

There are three common uses of MANOVA in practice:

- Between-group MANOVA: This is used when comparing two or more independent groups on multiple dependent variables. For example, researchers may compare teaching methods across several performance outcomes.

- Within-subjects (repeated measures) MANOVA: This is used when the same participants are measured on multiple dependent variables across different conditions or time periods.

- Mixed-design MANOVA: This is used when a study includes both between-subjects factors (e.g., treatment vs. control group) and within-subjects factors (e.g., measurements taken over time), along with multiple dependent variables.

Like ANOVA, MANOVA tests for statistically significant effects. However, instead of analyzing each dependent variable separately, it constructs a weighted linear combination of the dependent variables that maximizes the differences among groups or conditions.

MANOVA does not require the assumption of sphericity. Sphericity means that the variances of the differences between all pairs of within-subject conditions are equal. For example, suppose 10 subjects are measured at three time points. For each subject, we can compute the differences between Time 1 and Time 2, Time 1 and Time 3, and Time 2 and Time 3. Sphericity holds if the variances (or equivalently, the standard deviations) of these three sets of difference scores are approximately equal.

More formal statistical tests for sphericity exist, but they are beyond the scope of this chapter. Although MANOVA does not rely on the sphericity assumption, it is generally more conservative than univariate repeated-measures ANOVA. It means that MANOVA may have less statistical power to detect effects.

MANOVA relies on five important assumptions:

Independence of Observations: Each observation should be independent of the others. For example, one student's performance should not influence another student's performance. Violations of independence can seriously bias the results [28, 29].

Multivariate Normality: The dependent variables, when considered jointly, should follow a multivariate normal distribution within each group of the independent variable. In simpler terms, each dependent variable should be

approximately normally distributed. Their combination should also behave normally within each group [28, 29].

Homogeneity of Variance–Covariance Matrices: The variance–covariance matrices of the dependent variables should be equal across groups. This means that both the variability of each dependent variable and the relationships (covariances) between them are similar for all groups. In other words, the overall spread and structure of the data should be consistent across groups [28, 29].

Absence of Multicollinearity: The dependent variables should not be excessively correlated with one another. If two variables are highly correlated, they provide redundant information. Multicollinearity reduces the effectiveness of the multivariate analysis. Ideally, the dependent variables should be related, but not so strongly that they measure essentially the same construct [28, 29].

Linearity: The relationships between all pairs of dependent variables should be linear within each group of the independent variable. This means that scatterplots of each pair of dependent variables should show a roughly straight-line pattern rather than curves or more complex relationships [28, 29].

MANOVA offers several important advantages over ANOVA. While ANOVA analyzes a single dependent variable, MANOVA allows researchers to examine multiple dependent variables simultaneously.

For example, suppose we want to study how different diets affect both weight and blood pressure. These two outcomes are the dependent variables, while the type of diet—typically a categorical variable—is the independent variable that may influence them.

The primary goal of MANOVA is to determine whether there are statistically significant differences between groups across the combination of dependent variables. In this example, we ask whether one diet leads to greater overall changes in both weight and blood pressure compared to other diets.

A key strength of MANOVA is that it does not examine each dependent variable in isolation. Instead, it considers how the dependent variables vary together across groups. This joint variation is captured by the concept of covariance, which measures how changes in one variable are associated with changes in another. By incorporating these relationships, MANOVA can detect patterns that might be missed if each outcome were analyzed separately.

To perform a MANOVA, the study should be designed so that participants are randomly assigned to treatment groups whenever possible. This helps ensure that any observed differences between groups are due to the treatment rather than pre-existing differences.

In this respect, MANOVA follows the same experimental principles as simpler designs such as ANOVA.

Like ANOVA, MANOVA is a parametric method, but it extends the assumptions to the multivariate case. Specifically, the dependent variables are assumed to follow a multivariate normal distribution within each treatment group, and these distributions share a common variance–covariance matrix.

A variance–covariance matrix summarizes both the variability of each dependent variable and the relationships among them. The diagonal elements of the matrix represent the variances of each variable, while the off-diagonal elements represent the covariances, which indicate how pairs of variables move together. Because covariance between X and Y is the same as between Y and X, the matrix is symmetric, meaning the lower triangle mirrors the upper triangle.

For example, with three dependent variables, such as X, Y, and Z, the covariance matrix contains the variances of each variable along the diagonal and the covariances between each pair (X–Y, X–Z, and Y–Z) in the off-diagonal positions.

MANOVA uses this matrix to evaluate group differences while accounting for how the dependent variables are related to one another. Rather than analyzing each outcome separately, it incorporates these interrelationships to provide a more comprehensive test of group effects.

The variance–covariance matrix for the three dependent variables X, Y, and Z is illustrated below:

$$\begin{bmatrix} var(X) & cov(X,Y) & cov(X,Z) \\ cov(X,Y) & var(Y) & cov(Y,Z) \\ cov(X,Z) & cov(Y,Z) & var(Z) \end{bmatrix}$$

MANOVA is often described as more conservative than ANOVA because it evaluates multiple dependent variables in a single overall test. While this approach reduces the risk of false positives, it can also make it harder to detect significant effects.

If we instead estimate a series of separate (univariate) ANOVAs—one for each dependent variable—we ignore the correlations among the dependent variables. This creates a problem known as the inflation of the experimental (familywise) Type I error rate [30]. For example, if each test is conducted at the 5% significance level, the probability of making at least one false rejection increases as more tests are performed. In contrast, MANOVA accounts for the relationships among the dependent variables and controls this overall error rate within a single analysis.

A balanced design is preferred in MANOVA. It meana that each group has approximately the same number of observations. Balanced data improve the stability and reliability of the estimated variance–covariance matrices. Although ANOVA and MANOVA can sometimes handle unbalanced data, strong imbalances may distort results and reduce statistical power.

In practice, unbalanced data often arise from nonresponse or self-selection. For example, suppose a bank conducts a customer satisfaction survey. Dissatisfied or frustrated customers may be more motivated to respond, while satisfied customers may ignore the survey due to time constraints or lack of interest. As a result, the sample becomes uneven and may not accurately represent the full customer base.

Missing data present an additional challenge. One option is to remove observations or even variables with substantial missing values, but this may result in a loss of valuable information. Another common approach is imputation, such as replacing missing values with the mean or median. However, this method reduces variability in the data and can weaken relationships among variables, potentially affecting the MANOVA results. More advanced imputation techniques are often preferred, but they are beyond the scope of this chapter.

We must be careful when dealing with outliers. An outlier is an observation that is unusually far from the rest of the data and does not follow the general pattern. Outliers can have a strong influence on statistical results, especially in methods like MANOVA that rely on means, variances, and relationships among variables.

When an outlier is detected, the first step is to determine whether it is a data error or a valid but extreme observation. For example, data entry mistakes can easily occur. A value of 17 might be mistakenly recorded as 71. This could lead to unrealistic cases such as a 71-year-old pregnant woman. In such situations, the value should be corrected if the true value can be verified, or removed if it is clearly an error.

Outliers can also arise from unusual but real combinations of variables. For instance, a man who is 198 cm tall and weighs 56 kg may not appear extreme when considering height and weight separately. However, when these variables are examined together, the combination is highly unusual—indicating a multivariate outlier. This is particularly important in MANOVA, where relationships among variables are central to the analysis.

If an outlier is confirmed to be an error, it may be corrected or removed. If it is a valid observation, researchers must decide whether to keep it, transform the data, or use robust methods that reduce its influence. Regardless of the decision, it is essential to report any changes to the data, including whether observations were removed and the reasons for doing so. Transparency ensures that the results remain credible and reproducible.

To illustrate MANOVA, consider an experiment that evaluates different training methods for call center recruits handling travel-related customer queries.

We examine two types of training methods:

- Method 1: Trainees spend part of the first session learning the transportation network using maps and printed timetables available to the public.
- Method 2: Trainees immediately begin using the online information system that they will rely on when responding to customer queries.

In addition to training method, we vary the number of training sessions: recruits receive either one, two, or three sessions. Combining these factors gives us a 2×3 factorial design, resulting in six experimental conditions.

We randomly assign 30 new recruits to these conditions, with five participants in each group. Random assignment helps ensure that any differences in performance are due to the training methods rather than pre-existing differences among recruits.

After completing their assigned training, each recruit takes a performance test. During the test, trainers present a standard set of customer queries over the phone. We record two outcome measures (dependent variables):

- Correct: the number of queries answered correctly within a fixed time period
- Delay: the average time (in seconds) before the recruit begins responding to each query

These two dependent variables capture different aspects of performance: accuracy and response speed. Because they may be related (for example, faster responses might reduce accuracy), MANOVA is an appropriate method to analyze them jointly.

In practice, researchers often apply data transformations to address issues such as skewness or unequal variability. In this example, the researchers take the natural logarithm of the "correct" variable. Other transformations, such as the square root or reciprocal, could also be used depending on the data.

The purpose of a transformation is to reduce skewness and stabilize variability. By compressing large values more than small ones, the transformation reduces the overall spread of the data and helps produce more consistent residuals. This improves the validity of the statistical analysis and makes the assumptions of MANOVA more likely to be satisfied.

We show the Python code to estimate a MANOVA below. The Python code is available at: https://github.com/KenSzulczyk

```
# ==========================================
# MANOVA Example: Training Methods Experiment
# ==========================================

# Import required libraries
import pandas as pd
import numpy as np
from statsmodels.multivariate.manova import MANOVA

# ------------------------------------------
# Step 1: Load the dataset
# ------------------------------------------
# Make sure the file path is correct
df = pd.read_excel('Chapter_07-MANOVA.xlsx')

# Display first few rows to verify data
print(df.head())

# ------------------------------------------
# Step 2: Create transformed variable
# ------------------------------------------
# Take the natural logarithm of 'Correct'
# Add a small constant if needed to avoid log(0)
df['ln_Correct'] = np.log(df['Correct'])

# ------------------------------------------
# Step 3: MANOVA without transformation
# ------------------------------------------
# Model includes:
# - Main effects: Method, Sessions
# - Interaction: Method * Sessions
model_original = MANOVA.from_formula(
    'Correct + Delay ~ Method + Sessions + Method:Sessions',
    data=df
)

print("\nMANOVA Results (Original Variables):")
print(model_original.mv_test())

# ------------------------------------------
# Step 4: MANOVA with transformation
# ------------------------------------------
model_transformed = MANOVA.from_formula(
    'ln_Correct + Delay ~ Method + Sessions + Method:Sessions',
    data=df
)

print("\nMANOVA Results (Log-Transformed Correct):")
print(model_transformed.mv_test())
```

The Python output is below for the untransformed data:

```
MANOVA Results (Original Variables):
                  Multivariate linear model
============================================================

------------------------------------------------------------
       Intercept        Value  Num DF  Den DF F Value  Pr > F
------------------------------------------------------------
          Wilks' lambda 0.1014 2.0000 25.0000 110.7588 0.0000
         Pillai's trace 0.8986 2.0000 25.0000 110.7588 0.0000
 Hotelling-Lawley trace 8.8607 2.0000 25.0000 110.7588 0.0000
    Roy's greatest root 8.8607 2.0000 25.0000 110.7588 0.0000
------------------------------------------------------------

------------------------------------------------------------
          Method         Value  Num DF  Den DF F Value Pr > F
------------------------------------------------------------
           Wilks' lambda 0.6379 2.0000 25.0000  7.0966 0.0036
          Pillai's trace 0.3621 2.0000 25.0000  7.0966 0.0036
  Hotelling-Lawley trace 0.5677 2.0000 25.0000  7.0966 0.0036
     Roy's greatest root 0.5677 2.0000 25.0000  7.0966 0.0036
------------------------------------------------------------

------------------------------------------------------------
         Sessions        Value  Num DF  Den DF F Value Pr > F
------------------------------------------------------------
           Wilks' lambda 0.5298 2.0000 25.0000 11.0947 0.0004
          Pillai's trace 0.4702 2.0000 25.0000 11.0947 0.0004
  Hotelling-Lawley trace 0.8876 2.0000 25.0000 11.0947 0.0004
     Roy's greatest root 0.8876 2.0000 25.0000 11.0947 0.0004
------------------------------------------------------------

------------------------------------------------------------
     Method:Sessions     Value  Num DF  Den DF F Value Pr > F
------------------------------------------------------------
           Wilks' lambda 0.7435 2.0000 25.0000  4.3129 0.0246
          Pillai's trace 0.2565 2.0000 25.0000  4.3129 0.0246
  Hotelling-Lawley trace 0.3450 2.0000 25.0000  4.3129 0.0246
     Roy's greatest root 0.3450 2.0000 25.0000  4.3129 0.0246
============================================================
```

The Python output for the log transformed data is below:

```
MANOVA Results (Log-Transformed Correct):
                  Multivariate linear model
============================================================

------------------------------------------------------------
       Intercept        Value  Num DF  Den DF F Value  Pr > F
------------------------------------------------------------
          Wilks' lambda 0.0966 2.0000 25.0000 116.9467 0.0000
         Pillai's trace 0.9034 2.0000 25.0000 116.9467 0.0000
 Hotelling-Lawley trace 9.3557 2.0000 25.0000 116.9467 0.0000
```

```
    Roy's greatest root 9.3557 2.0000 25.0000 116.9467 0.0000
------------------------------------------------------------


------------------------------------------------------------
          Method         Value  Num DF  Den DF F Value Pr > F
------------------------------------------------------------
           Wilks' lambda 0.6353 2.0000 25.0000  7.1773 0.0034
          Pillai's trace 0.3647 2.0000 25.0000  7.1773 0.0034
  Hotelling-Lawley trace 0.5742 2.0000 25.0000  7.1773 0.0034
     Roy's greatest root 0.5742 2.0000 25.0000  7.1773 0.0034
------------------------------------------------------------


------------------------------------------------------------
         Sessions        Value  Num DF  Den DF F Value Pr > F
------------------------------------------------------------
           Wilks' lambda 0.5347 2.0000 25.0000 10.8764 0.0004
          Pillai's trace 0.4653 2.0000 25.0000 10.8764 0.0004
  Hotelling-Lawley trace 0.8701 2.0000 25.0000 10.8764 0.0004
     Roy's greatest root 0.8701 2.0000 25.0000 10.8764 0.0004
------------------------------------------------------------


------------------------------------------------------------
     Method:Sessions     Value  Num DF  Den DF F Value Pr > F
------------------------------------------------------------
           Wilks' lambda 0.7909 2.0000 25.0000  3.3053 0.0533
          Pillai's trace 0.2091 2.0000 25.0000  3.3053 0.0533
  Hotelling-Lawley trace 0.2644 2.0000 25.0000  3.3053 0.0533
     Roy's greatest root 0.2644 2.0000 25.0000  3.3053 0.0533
============================================================
```

The log transformation has little impact on the overall results, as the conclusions remain largely unchanged. Most of the multivariate test statistics are statistically significant at the 5% level, except the interaction term (Method × Sessions) in the log-transformed model. This suggests that, after transformation, there is weaker evidence that the effect of training method depends on the number of sessions.

In most applications, we are not particularly interested in the intercept. The intercept represents the overall mean of the dependent variables when no predictors are included in the model. As such, it typically provides little substantive insight in experimental settings where the focus is on group differences.

For each factor in MANOVA, statistical software reports four alternative test statistics. These tests differ in how they combine information from the dependent variables and in how sensitive they are to violations of assumptions:

> **Wilks' Lambda:** It measures the proportion of total variance in the dependent variables not explained by the independent variable. Smaller values indicate stronger group differences. It is one of the most commonly reported statistics, but it is more sensitive to violations of assumptions, particularly multivariate normality and homogeneity of covariance matrices.

Pillai's Trace: It measures the proportion of explained variance in the dependent variables accounted for by the independent variable. It is generally considered the most robust statistic, especially when assumptions such as equal covariance matrices or multivariate normality are violated, or when sample sizes are small or unequal. Because of this robustness, many researchers prefer Pillai's Trace as the default test.

Hotelling-Lawley Trace: This statistic is based on the ratio of explained to unexplained variance across the dependent variables. It is conceptually similar to Wilks' Lambda but tends to be more powerful when group differences are large. However, it relies more heavily on the assumption of multivariate normality and can be less stable when sample sizes are small.

Roy's Largest Root: It focuses on the single largest dimension of group separation among the dependent variables. In other words, it tests whether there is one dominant linear combination of dependent variables that strongly differentiates the groups. It can be very powerful when such a dominant effect exists, but it is also the most sensitive to assumption violations and may increase the risk of Type I errors if used indiscriminately.

Key Insight

MANOVA evaluates group differences by considering both mean differences and the relationships (covariance) among dependent variables. In contrast, ANOVA analyzes each dependent variable separately and does not account for how the variables move together.

By incorporating these relationships, MANOVA can detect patterns that may be missed when variables are examined one at a time, providing a more comprehensive understanding of group effects.

7.2 Analysis of Covariance (ANCOVA)

Analysis of Covariance (ANCOVA) is a statistical technique that allows researchers to include both categorical variables (factors) and continuous variables (covariates) in a single model [30, 31]. It combines ideas from ANOVA (comparing group means) and regression analysis (modeling relationships with continuous variables).

ANCOVA relies on several important assumptions:

Linearity: The relationship between the dependent variable and the covariate should be approximately linear within each group of the independent variable. If

this relationship is not linear, it may indicate more complex underlying processes. Thus, the model may not be appropriate.

Homogeneity of Regression Slopes: The relationship between the covariate and the dependent variable should be the same across all groups. In other words, the regression lines for each group should be parallel. If the slopes differ, it suggests an interaction between the covariate and the treatment, which requires a different model specification.

Homogeneity of Variance (Homoscedasticity): The variability of the residuals should be similar across all levels of the independent variable and across the range of the covariate. The data points should not become more widely spread as values increase. Violations of this assumption can lead to unreliable statistical conclusions.

A covariate is a continuous variable that is not the main focus of the study but may still influence the dependent variable. It is sometimes referred to as a control variable or a nuisance variable.

By including covariates in the model, ANCOVA:

- Adjusts group comparisons for pre-existing differences
- Reduces unexplained variation (noise)
- Improves the precision of estimated treatment effects

Researchers can include one or more covariates to ensure that the observed differences between groups are not driven by these external factors.

We use ANCOVA when:

- We want to compare multiple groups, like ANOVA
- We also have continuous variables that may influence the outcome
- We want to control for these variables to isolate the true effect of the treatment

In short, ANCOVA helps us make fairer comparisons between groups.

Real world examples include the following:

- Marketing: Evaluate the impact of a new marketing campaign on sales while controlling for advertising spending.
- Medical research: Compare different treatments for reducing cholesterol while controlling for patient age and baseline cholesterol levels.

Consider a teacher evaluating a new teaching method. Suppose we include intelligence score as a covariate. If the regression lines for each teaching method are parallel, then the effect of intelligence is consistent across groups. This satisfies the homogeneity of regression slopes assumption, and ANCOVA is appropriate. In this case, ANCOVA adjusts the group means by removing the effect of intelligence, allowing a fair comparison of teaching methods. Figure 7.1 illustrates parallel regression lines for the new and current teaching methods.

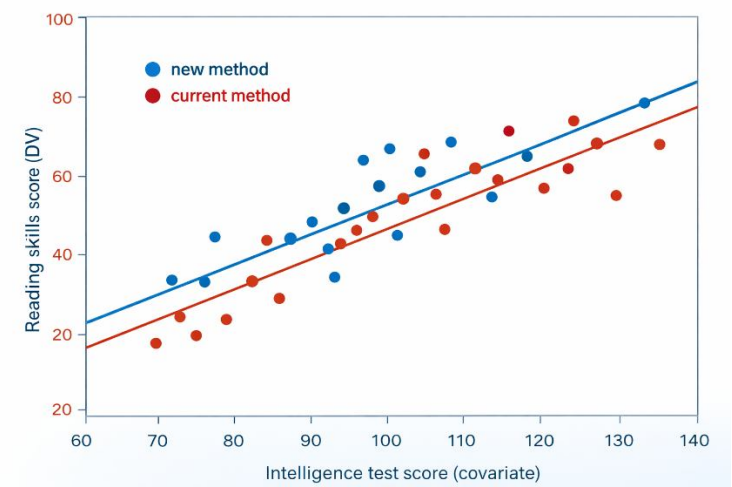


Figure 7.1. Comparing Teaching Methods

In some cases, the relationship between the covariate and the outcome differs across groups. This means the slopes are not parallel. Thus, it indicates an interaction effect.

For example, treatment groups may respond differently depending on pre-test scores. Students with higher initial ability may benefit more from one method than another. In this situation, the assumption of homogeneity of slopes is violated. We should include an interaction term between the covariate and the treatment. If the interaction is statistically significant, ANCOVA without interaction is not appropriate. Figure 7.2 illustrates non-parallel lines, showing interaction between treatment and covariate.

A teacher wants to determine whether three different studying techniques affect exam scores. However, students already differ in their current performance, which may influence the results.

The variables are:

- Factor (independent variable): Studying technique
- Covariate: Current grade in the class
- Dependent variable: Exam score

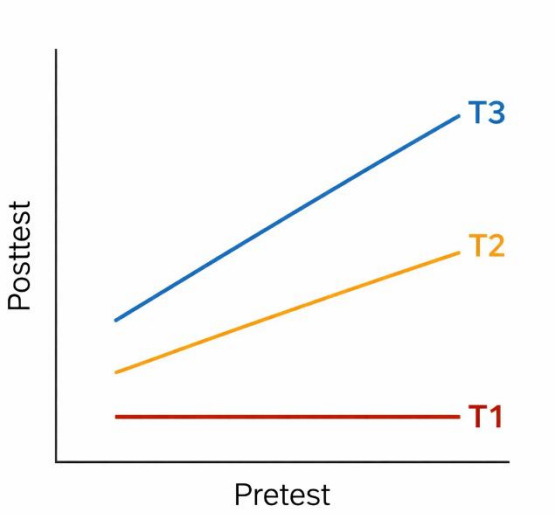


Figure 7.2. Treatment Covariate Interaction

The current grade reflects a student's existing ability and study habits. By including it as a covariate, the teacher controls for these differences and obtains a more accurate estimate of the effect of each studying technique.

Using ANCOVA, the teacher tests whether the adjusted mean exam scores differ across the three techniques. If the null hypothesis is rejected, we conclude that at least one studying technique leads to significantly different performance, after controlling for students' prior achievement.

The Python code for the ANCOVA is shown below. The Python code is available at: https://github.com/KenSzulczyk

```
# ============================================
# ANCOVA Example: Study Techniques
# ============================================

# --------------------------------------------
# Step 1: Import required libraries
# --------------------------------------------
import numpy as np
import pandas as pd

# --------------------------------------------
# Step 2: Create the dataset
# --------------------------------------------
# We simulate data for three study techniques (A, B, C)
# Each student has:
# - current_grade (covariate)
# - exam_score (dependent variable)
```

```
df = pd.DataFrame({
    'technique': np.repeat(['A', 'B', 'C'], 5),
    'current_grade': [
        67, 88, 75, 77, 85,
        92, 69, 77, 74, 88,
        96, 91, 88, 82, 80
    ],
    'exam_score': [
        77, 89, 72, 74, 69,
        78, 88, 93, 94, 90,
        85, 81, 83, 88, 79
    ]
})

# View the dataset
print("Dataset:")
print(df)

# ------------------------------------------
# Step 3: Install and import ANCOVA library
# ------------------------------------------
# Run this line once in your terminal or notebook
# pip install pingouin

from pingouin import ancova

# ------------------------------------------
# Step 4: Perform ANCOVA
# ------------------------------------------
# dv = dependent variable (exam_score)
# covar = covariate (current_grade)
# between = factor (technique)

results = ancova(
    data=df,
    dv='exam_score',
    covar='current_grade',
    between='technique'
)

# Display results
print("\nANCOVA Results:")
print(results)
```

The Python output is below:

```
ANCOVA Results:
          Source          SS  DF         F     p_unc       np2
0      technique  390.575130   2  4.809973  0.031556  0.466536
1  current_grade    4.193886   1  0.103296  0.753934  0.009303
2       Residual  446.606114  11       NaN       NaN       NaN
```

We reject the null hypothesis for the teaching method. It indicates that there are statistically significant differences in exam scores across the three techniques after controlling for current grade. In other words, the choice of teaching method has a meaningful impact on student performance.

The current grade (covariate) is not statistically significant. This suggests that, within this sample, prior performance does not have a strong independent effect on exam scores once the teaching method is taken into account.

Key Insight

ANCOVA improves upon ANOVA by adjusting for important background variables. This allows researchers to isolate the true effect of the treatment and make more accurate and meaningful comparisons between groups.

7.3 Discriminant Analysis

Discriminant analysis is a statistical technique used to classify observations into predefined groups and to identify the variables that best separate those groups [27, 29]. It plays an important role in statistics, econometrics, and modern machine learning.

There are two key ideas in discriminant analysis:

Classification: Assigning observations to known categories with the smallest possible error

Separation: Identifying the combinations of variables that best distinguish between groups

Classification focuses on placing each observation into a known group based on its characteristics. The goal is to minimize classification errors. For example, suppose we want to predict whether a convicted criminal will reoffend within five years. We could use variables such as:

- Number of previous convictions
- Age
- Time spent in rehabilitation

Using these variables, discriminant analysis constructs a rule that assigns individuals to either:

- Likely to reoffend

- Unlikely to reoffend

Separation focuses on understanding how groups differ from one another. Instead of only assigning observations, we also want to interpret the structure of the data. For example, consider accident survivors who experience:

- Severe trauma
- Mild trauma
- No trauma

We may use variables such as social support, counseling time, and job satisfaction to determine which combinations of factors best separate these groups.

In modern applications, discriminant analysis is closely related to machine learning. A common approach is to split the dataset into:

- Training set (about 80%) → used to estimate the model
- Testing set (about 20%) → used to evaluate accuracy

The model learns patterns from the training data and is then tested on new, unseen data [32]. This process helps us evaluate how well the model generalizes.

One important assumption of classical discriminant analysis is that the variance–covariance matrices are equal across groups. Statistical tests, such as Box's M test, can be used to assess this assumption.

Suppose we want to classify individuals as male or female based on height (h) and weight (w). In general, males tend to be taller and heavier than females. The question is: Can we draw a line that separates the two groups with minimal classification error in Figure 7.3?

Discriminant analysis constructs functions that help us assign observations to groups. The discriminant functions for males and females might be:

$$Discriminant\ (male) = -435.945 + 4.488 \cdot h + 1.1117 \cdot w$$

$$Discriminant\ (female) = -388.785 + 4.241 \cdot h + 1.041 \cdot w$$

To classify an individual, we compute both scores. Then we assign the individual to the group with the higher value. For example, a person is 184 cm tall and weighs 67 kg.

$$Discriminant\ (male) = -435.945 + 4.488 \cdot 184 + 1.1117 \cdot 67 = 464.33$$

$$Discriminant\ (female) = -388.785 + 4.241 \cdot 184 + 1.041 \cdot 67 = 461.31$$

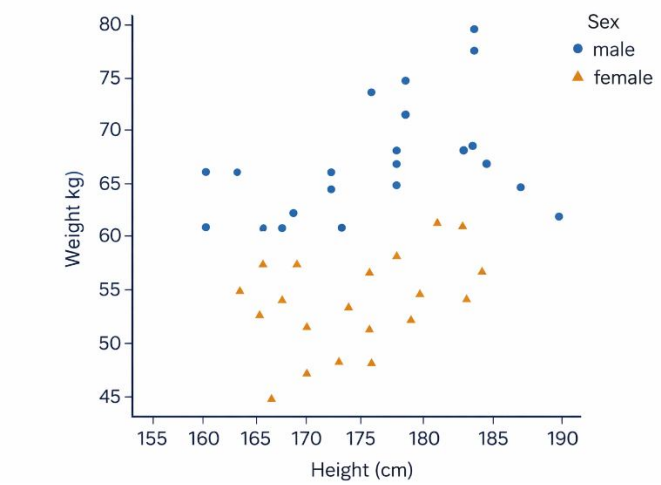


Figure 7.3. Males vs. Female Groups

Since the male score is higher, the individual is classified as male. The male is correctly classified.

In another example, a person is 170 cm tall and weighs 67 kg:

$$Discriminant\ (male) = -435.945 + 4.488 \cdot 170 + 1.1117 \cdot 67 = 402.05$$

$$Discriminant\ (female) = -388.785 + 4.241 \cdot 170 + 1.041 \cdot 67 = 401.93$$

The model classifies this individual as male, even though the person is female. This is a classification error. It is unavoidable in most real-world problems.

We now apply discriminant analysis to the well-known Iris dataset. This dataset is widely used in statistics and machine learning because it contains clear group structure. Each observation includes four measurements: sepal length, sepal width, petal length, and petal width. The goal is to classify each flower into one of three species. Figure 7.4 shows these characteristics on an Iris flower.

We divide the dataset into: training set (80%) and testing set (20%).

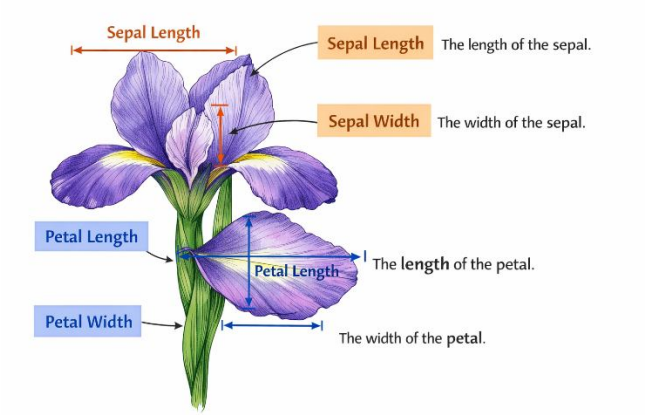


Figure 7.4. Iris Flower Features

The Python code to analyze the Iris problem is shown below. The Python code is available at: https://github.com/KenSzulczyk

```
# ============================================
# Discriminant Analysis Example: Iris Dataset
# ============================================

# --------------------------------------------
# Step 1: Import required libraries
# --------------------------------------------
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

# --------------------------------------------
# Step 2: Load the dataset
# --------------------------------------------
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

# Define column names
columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']

# Read dataset
df = pd.read_csv(url, names=columns)

print("First five observations:")
print(df.head())
```

```
# ------------------------------------------
# Step 3: Prepare the data
# ------------------------------------------
# Separate features (X) and target variable (y)
X = df.iloc[:, 0:4].values
y = df.iloc[:, 4].values

# Convert categorical labels into numeric values
le = LabelEncoder()
y = le.fit_transform(y)

# ------------------------------------------
# Step 4: Explore the data
# ------------------------------------------
# Visualize distributions of each feature
plt.figure(figsize=(12, 6))

for i, feature in enumerate(columns[:-1]):
    plt.subplot(2, 2, i + 1)
    sns.histplot(data=df, x=feature, hue='class', kde=True)
    plt.title(f'{feature} Distribution')

plt.tight_layout()
plt.show()

# ------------------------------------------
# Step 5: Split data into training and testing sets
# ------------------------------------------
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# ------------------------------------------
# Step 6: Apply Linear Discriminant Analysis (LDA)
# ------------------------------------------
lda = LinearDiscriminantAnalysis(n_components=2)

# Fit LDA on training data and transform
X_train_lda = lda.fit_transform(X_train, y_train)
X_test_lda = lda.transform(X_test)

# ------------------------------------------
# Step 7: Visualize LDA components
# ------------------------------------------
lda_df = pd.DataFrame(X_train_lda, columns=['LDA1', 'LDA2'])
lda_df['Class'] = y_train

sns.scatterplot(
    data=lda_df,
    x='LDA1',
    y='LDA2',
    hue='Class',
    palette='deep'
```

```
)

plt.title("LDA Projection of Training Data")
plt.show()

# -----------------------------------------
# Step 8: Classification using Random Forest
# -----------------------------------------
model = RandomForestClassifier(max_depth=2, random_state=0)

model.fit(X_train_lda, y_train)

# Predict on test data
y_pred = model.predict(X_test_lda)

# -----------------------------------------
# Step 9: Evaluate model performance
# -----------------------------------------
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print(f"\nAccuracy: {accuracy:.2f}")

# Plot confusion matrix
plt.figure(figsize=(6, 6))
sns.heatmap(
    conf_matrix,
    annot=True,
    fmt="d",
    cbar=False,
    square=True,
    cmap='Blues'
)

plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

Before modeling, we examine the data using histograms in Figure 7.5. We observe that the sepal length and width overlap across species. The petal length and width show clearer separation. This suggests that petal measurements are more informative for classification

We apply Linear Discriminant Analysis (LDA) to reduce the four variables into two components that maximize separation between species. One species is clearly separated from the others in Figure 7.6. The remaining two species show some overlap. This demonstrates how LDA finds the directions that best distinguish groups.

We also apply a Random Forest classifier. It combines multiple decision trees to improve prediction accuracy. It has the advantage for higher accuracy and robustness. The trade-off is it has less interpretability compared to simpler models.

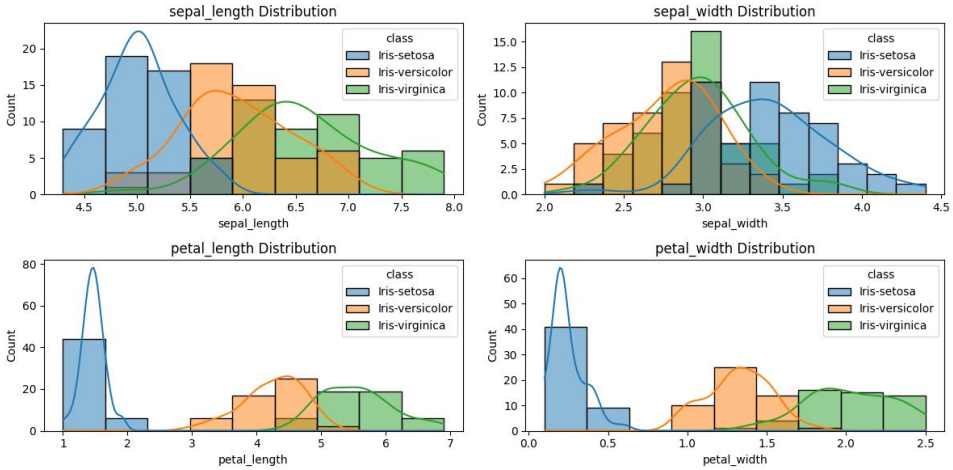


Figure 7.5. Iris Feature Distributions

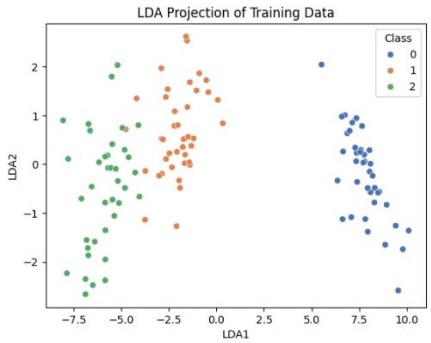


Figure 7.6. Linear Discriminant Analysis in Two Dimensions

We evaluate the model using a confusion matrix in Figure 7.7. The diagonal elements represent correctly classified flowers. The off-diagonal elements represent misclassifications. The results show that almost all flowers are correctly classified. Only one observation is misclassified. This indicates excellent model performance, largely due to the strong separation between species.

Key Insight

Discriminant analysis provides both:

- A classification tool (predicting group membership)
- A dimension reduction tool (understanding how variables separate groups)

In practice, it helps us answer two important questions:

- Which group does this observation belong to?
- What characteristics distinguish these groups?

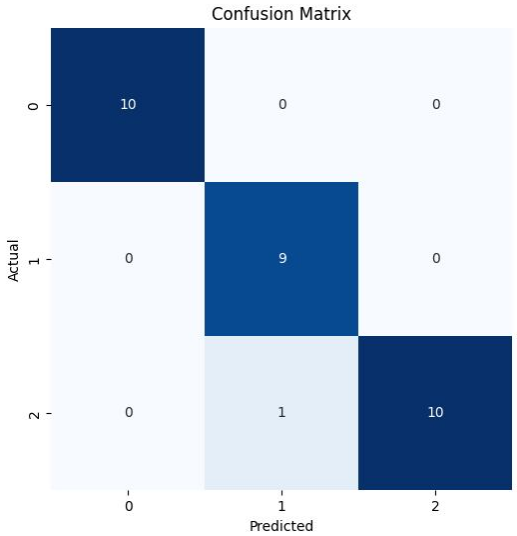


Figure 7.7. Confusion Matrix

7.4 Cluster Analysis

Cluster analysis is a statistical technique used to group observations based on similarity [32, 33]. Unlike many of the methods we have studied so far, we do not know the group memberships in advance. Instead, we use an algorithm to discover patterns and organize the data into clusters.

Because the group labels are unknown, cluster analysis is an example of unsupervised learning. In contrast, methods such as discriminant analysis are examples of supervised learning, where we already know the group for each observation and use that information to build a classification rule.

A key concept in cluster analysis is distance. Distance measures how similar or dissimilar two observations are:

- Small distance → observations are similar
- Large distance → observations are dissimilar

For example, consider the distances in kilometers between five U.S. cities shown in Figure 7.8. Cities that are geographically close are considered more similar, while those far apart are less similar.

	Chicago	Detroit	St. Louis	Nashville	Cincinnati
Chicago	0				
Detroit	380	0			
St. Louis	420	740	0		
Nashville	640	860	500	0	
Cincinnati	410	370	500	400	0

Figure 7.8. Distances between U.S. Cities (in kilometers)

From the table, we observe that Chicago, Detroit, and Cincinnati are relatively close and form a natural group. Nashville is farther away from the others and is less similar. This idea of grouping based on distance is the foundation of cluster analysis.

There are several clustering algorithms, but two of the most common are:

K-means clustering: We specify the number of clusters k in advance. The algorithm assigns observations to clusters based on distance.

Hierarchical clustering: The number of clusters is not specified initially. The algorithm builds clusters step by step. It allows the data to determine the structure.

We illustrate clustering using the Iris dataset, which contains three species: Iris setosa, Iris versicolor, and Iris virginica. Before applying K-means, we standardize the variables by subtracting the mean and dividing by the standard deviation. This ensures that all variables are on the same scale (mean = 0, standard deviation = 1). Without standardization, variables with larger units would dominate the distance calculations.

K-means works as follows: We choose the number of clusters k. We assign each observation to the nearest cluster center (centroid). We update the centroids based on cluster members. Finally, we repeat until the clusters stabilize. The goal is to group observations so that those within the same cluster are as similar as possible.

We know the Iris dataset has three types: Iris setosa, Iris virginica and Iris versicolor. We use K-means. We subtract the mean and divide by the standard deviation to standardize the variables. All variables have the same scaling, mean = 0 and std dev. = 1. K-means surrounds points with a centroid and groups observations together that have the smallest distances between each other.

A common challenge in K-means is selecting the appropriate number of clusters. One popular approach is the Elbow Method. We compute inertia, which is the within-cluster sum of squared distances, for different values of k. As k increases, inertia decreases (clusters become tighter). However, improvements eventually slow down. We look for the point where the curve begins to bend—this is the "elbow," indicating a good choice for k [32].

In the Iris dataset, the elbow occurs at k = 3 in Figure 7.9. This matches the known number of species. However, because K-means uses random starting points, different runs may occasionally suggest slightly different results.

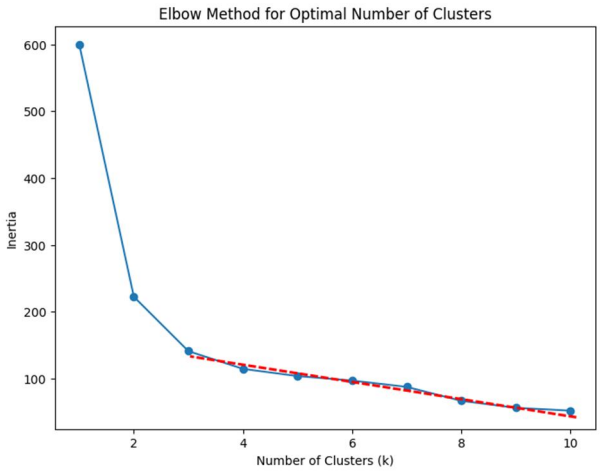


Figure 7.9. Elbow Method for Clustering

Another way to evaluate clustering is the silhouette score, which measures how well-separated the clusters are. Values range from –1 to 1. Higher values indicate better-defined clusters.

For the Iris dataset:

- k = 2: 0.5818
- k = 3: 0.4799
- k = 4: 0.3850

We run the Python program three times and change k to either 2, 3, or 4.

The silhouette score suggests that two clusters are most clearly separated. This makes sense because one species, Iris setosa, is very distinct from the other two. However, we know that the dataset actually contains three species. Therefore, even though two clusters provide stronger separation, choosing three clusters better reflects the true structure of the data.

The Python code to analyze the Iris problem is shown below. The Python code is available at: https://github.com/KenSzulczyk

```
# ==========================================
# Cluster Analysis Example: Iris Dataset
# ==========================================

# ------------------------------------------
# Step 1: Import required libraries
# ------------------------------------------
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.cluster import KMeans
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score

# ------------------------------------------
# Step 2: Load the dataset
# ------------------------------------------
iris = load_iris()

# Features (independent variables)
X = iris.data

# Convert to DataFrame for easier handling
df = pd.DataFrame(X, columns=iris.feature_names)

print("First five observations:")
print(df.head())

# ------------------------------------------
# Step 3: Standardize the data
# ------------------------------------------
# Important: K-means is sensitive to scale
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# ------------------------------------------
# Step 4: Determine optimal number of clusters (Elbow Method)
# ------------------------------------------
inertia = []

for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=7)
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)

# Plot Elbow curve
plt.figure()
```

```
plt.plot(range(1, 11), inertia, marker='o')
plt.title('Elbow Method for Optimal Number of Clusters')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.show()

# -----------------------------------------
# Step 5: Fit K-means with chosen number of clusters
# -----------------------------------------
# From the elbow plot, we typically choose k = 3
optimal_k = 3

kmeans = KMeans(n_clusters=optimal_k, random_state=42)
kmeans.fit(X_scaled)

# Predicted cluster labels
cluster_labels = kmeans.labels_

# -----------------------------------------
# Step 6: Evaluate clustering performance
# -----------------------------------------
sil_score = silhouette_score(X_scaled, cluster_labels)
print(f"\nSilhouette Score for {optimal_k} clusters: {sil_score:.4f}")

# -----------------------------------------
# Step 7: Add cluster labels to dataset
# -----------------------------------------
df['Cluster'] = cluster_labels

print("\nFirst few rows with cluster assignments:")
print(df.head())

# -----------------------------------------
# Step 8: Visualize clusters
# -----------------------------------------
sns.pairplot(df, hue='Cluster', palette='Set1')
plt.suptitle('Cluster Analysis of Iris Flowers', y=1.02)
plt.show()
```

The Python output for the Silhouette score is below:

```
Silhouette Score for 3 clusters: 0.4799

First few rows with cluster assignments:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   Cluster
0        1
```

```
1         2
2         2
3         2
4         1
```

Key Insight

Cluster analysis allows the data to reveal its own structure. While statistical measures such as inertia and silhouette score provide guidance, domain knowledge and interpretation are equally important when deciding on the number of clusters.

7.5 Logistic Regression

Logistic regression is a special type of regression model used when the dependent variable y takes only two possible values, typically 0 or 1 [34]. These are called binary outcomes. Many real-world variables are binary. For example:

- Gender (male / female)
- Passing a course (pass / fail)
- Declaring bankruptcy (yes / no)
- Having a disease (present / absent)

In these cases, we use explanatory variables (independent variables) to estimate the probability that y = 1. Logistic regression is therefore widely used for classification problems, where we assign observations to one of two groups.

A standard multiple regression model is written as:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon$$

However, this model is not appropriate for binary outcomes because it can produce predicted values less than 0 or greater than 1. These are not valid probabilities.

To solve this problem, the logistic regression transforms the linear model into a new variable:

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon$$

This linear combination is then passed through the logistic (sigmoid) function, which ensures that predicted values lie between 0 and 1:

$$E(y) = \frac{e^z}{1 + e^z} = \frac{1}{1 + e^{-z}}$$

This function produces an S-shaped curve, where probabilities approach 0 for small values of z and approach 1 for large values of z.

We can write the model as:

$$E(y) = P(y = 1|x_1, x_2 \cdots x_p)$$

This represents the probability that the outcome equals 1 given the explanatory variables.

For example, suppose we have only one explanatory variable in Figure 7.10:

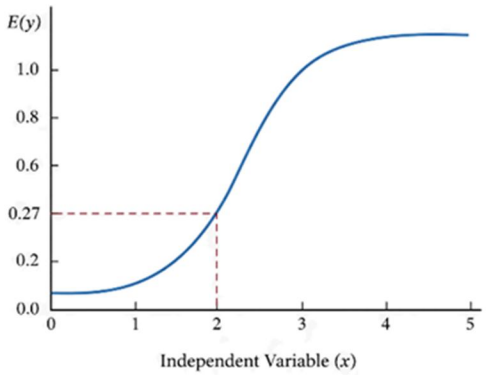


Figure 7.10. Logistic Regression Model

- If x > 3, the probability of y = 1 is high
- If x < 2, the probability of y = 1 is low
- Between 2 and 3 is a transition region, where classification is uncertain

In practice, we classify observations as:

- y = 1 if predicted probability ≥ 0.5
- y = 0 if predicted probability < 0.5

Instead of working directly with probabilities, logistic regression is often interpreted using odds:

$$odds = \frac{P(y = 1)}{P(y = 0)} = \frac{P(y = 1)}{1 - P(y = 1)}$$

Remember, we only have two events. All events, two in this case, adds up to one as shown below:

$$P(y = 1) + P(y = 0) = 1$$

The odds represent how likely an event is to occur relative to it not occurring.

For example: If the odds are 2 to 1, then:

$$P(y = 1) = \frac{2}{2 + 1} = 0.67$$

An odds ratio compares the odds before and after a change in an explanatory variable:

$$odds\ ratio = \frac{odds_1}{odds_0}$$

For instance, increasing x_1 by one unit (holding other variables constant) changes the odds of the outcome. The coefficient β_1 tells us how strongly this variable affects the odds.

Logistic regression can also be written in log-odds form, called the logit model [3, 34]:

$$ln \frac{P}{1 - P} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$$

This transformation makes the model linear in the parameters, which allows us to estimate it using maximum likelihood.

Unlike linear regression, logistic regression does not report an R^2 or F-statistic in the usual sense. This is because the model is nonlinear and does not decompose variance in the same way. Instead, we use measures such as pseudo R^2 and likelihood-based tests.

We apply logistic regression to study whether individuals develop diabetes. The dependent variable is:

- y = 1: has diabetes
- y = 0: does not have diabetes

The explanatory variables include:

- Pregnant – number of pregnancies
- Insulin – insulin level
- BMI – body mass index
- Age – age of the individual
- Glucose – blood glucose level
- BP – blood pressure

- Pedigree – genetic predisposition

The Python code for the logistic regression is shown below. The Python code is available at: https://github.com/KenSzulczyk

```
# ==========================================
# Logistic Regression Example: Diabetes Dataset
# ==========================================

# ------------------------------------------
# Step 1: Import required libraries
# ------------------------------------------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import classification_report

import statsmodels.api as sm

# ------------------------------------------
# Step 2: Load and clean the dataset
# ------------------------------------------

# Load dataset
df = pd.read_excel('Chapter_07-diabetes.xlsx')

# Convert all columns to numeric (important for modeling)
df = df.apply(pd.to_numeric)

# Preview data
print("First five observations:")
print(df.head())

# ------------------------------------------
# Step 3: Define features (X) and target (y)
# ------------------------------------------
feature_cols = [
    'pregnant', 'insulin', 'bmi',
    'age', 'glucose', 'bp', 'pedigree'
]

X = df[feature_cols]   # Independent variables
y = df['outcome']      # Dependent variable

# ------------------------------------------
# Step 4: Split into training and testing sets
# ------------------------------------------
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=16
)

# ----------------------------------------
# Step 5: Estimate logistic regression (sklearn)
# ----------------------------------------
model = LogisticRegression(max_iter=1000)

# Fit the model
model.fit(X_train, y_train)

# Predict on test data
y_pred = model.predict(X_test)

# ----------------------------------------
# Step 6: Evaluate model performance
# ----------------------------------------
# Confusion matrix
conf_matrix = metrics.confusion_matrix(y_test, y_pred)

print("\nConfusion Matrix:")
print(conf_matrix)

# ----------------------------------------
# Step 7: Visualize confusion matrix
# ----------------------------------------
plt.figure()
sns.heatmap(
    pd.DataFrame(conf_matrix),
    annot=True,
    fmt='g',
    cmap = 'Blues'
)

plt.title('Confusion Matrix')
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
plt.show()

# ----------------------------------------
# Step 8: Classification report
# ----------------------------------------
target_names = ['Without Diabetes', 'With Diabetes']

print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=target_names))

# ----------------------------------------
# Step 9: Estimate Logit model (statsmodels)
# ----------------------------------------
# Add constant for intercept
X_train_sm = sm.add_constant(X_train)
```

```
# Fit Logit model
logit_model = sm.Logit(
    y_train.astype(float),
    X_train_sm.astype(float)
).fit()

# Display results
print("\nLogit Model Summary:")
print(logit_model.summary())
```

Before estimating the model, we divide the dataset:

- 75% training data → used to estimate the model
- 25% testing data → used to evaluate prediction accuracy

The model does not see the test data during training. This ensures that we evaluate how well the model performs on new, unseen data.

The confusion matrix summarizes classification results in Figure 7.11:

- Diagonal elements → correct predictions
- Off-diagonal elements → classification errors

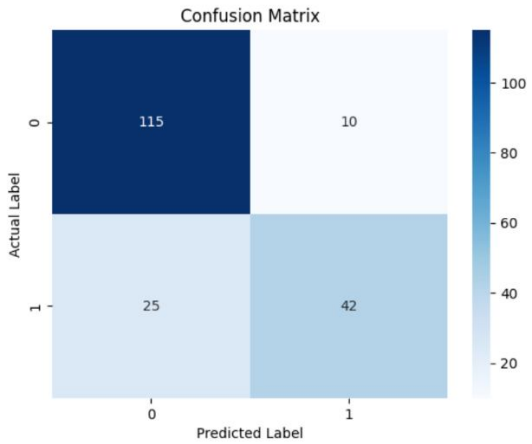


Figure 7.11. Confusion Matrix

In this case:

- 25 diabetic individuals were incorrectly classified as non-diabetic
- 10 non-diabetic individuals were incorrectly classified as diabetic

The classification report on the next page shows:

- Accuracy ≈ 82%
- The model correctly identifies:
  - 81% of diabetic cases
  - 82% of non-diabetic cases

This indicates that the model performs reasonably well, though it is slightly better at identifying non-diabetic individuals.

```
Classification Report:
                  precision    recall  f1-score   support

Without Diabetes       0.82      0.92      0.87       125
   With Diabetes       0.81      0.63      0.71        67

        accuracy                           0.82       192
       macro avg       0.81      0.77      0.79       192
    weighted avg       0.82      0.82      0.81       192
```

We have the estimated logistic output below:

```
Optimization terminated successfully.
         Current function value: 0.490919
         Iterations 6

Logit Model Summary:
                           Logit Regression Results
==============================================================================
Dep. Variable:                outcome   No. Observations:                  576
Model:                          Logit   Df Residuals:                      568
Method:                           MLE   Df Model:                            7
Date:                Mon, 06 Apr 2026   Pseudo R-squ.:                  0.2410
Time:                        05:01:33   Log-Likelihood:                -282.77
converged:                       True   LL-Null:                       -372.56
Covariance Type:            nonrobust   LLR p-value:                2.396e-35
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -7.8196      0.793     -9.866      0.000      -9.373      -6.266
pregnant       0.0914      0.037      2.484      0.013       0.019       0.164
insulin       -0.0014      0.001     -1.469      0.142      -0.003       0.000
bmi            0.0908      0.016      5.548      0.000       0.059       0.123
age            0.0128      0.011      1.155      0.248      -0.009       0.035
glucose        0.0325      0.004      8.033      0.000       0.025       0.040
bp            -0.0137      0.006     -2.244      0.025      -0.026      -0.002
pedigree       0.8962      0.340      2.632      0.008       0.229       1.564
==============================================================================
```

From the estimated logit model:

- Pregnant, BMI, glucose, and pedigree have positive and statistically significant coefficients
    - These variables increase the probability of having diabetes
- Blood pressure (BP) has a negative and significant coefficient
    - Higher BP is associated with a lower probability of diabetes in this model
- Insulin and age are not statistically significant
    - These variables do not have a clear independent effect, possibly due to correlation with other variables (multicollinearity)

Key Insight

Logistic regression allows us to:

- Estimate probabilities for binary outcomes
- Classify observations into groups
- Interpret how explanatory variables affect the likelihood of an event

It is one of the most widely used tools in statistics, economics, and machine learning for decision-making under uncertainty.

7.6 Summary

In this chapter, we expanded our statistical toolkit to address situations involving multiple variables and more complex data structures. We began with MANOVA, which allows us to test for group differences across several dependent variables simultaneously while accounting for their relationships. We then introduced ANCOVA, which improves comparisons by controlling for important covariates and reducing unwanted variation.

We also explored methods for classification and pattern recognition. Discriminant analysis provided a framework for assigning observations to known groups. Meanwhile, cluster analysis revealed natural groupings within the data without prior assumptions. These techniques are essential in modern data analysis. Researchers must identify structure and segmentation, and they are often as important as estimating relationships.

Finally, logistic regression offered a practical approach to modeling binary outcomes and making predictions. By interpreting probabilities, odds, and coefficients, we gained insight into how explanatory variables influence the likelihood of an event.

A key theme throughout this chapter is that real-world data are rarely simple. By incorporating multiple variables, controlling for external factors, and recognizing patterns in the data, we can produce more accurate, insightful, and actionable results. These methods form the foundation for more advanced topics in statistics, econometrics, and

machine learning. They are widely used in both academic research and industry applications.

7.7 Exercises

Problem 1.

In this exercise, we will apply Multivariate Analysis of Variance (MANOVA) using the well-known Iris dataset. This dataset, originally introduced by Ronald Fisher in 1936, is widely used for statistical analysis, machine learning, and data visualization.

The dataset contains measurements on iris flowers from three different species. For each observation, the following four variables are recorded:

- Sepal length
- Sepal width
- Petal length
- Petal width

These four variables will serve as the dependent variables in your analysis. The independent variable is the species of the iris flower, which classifies each observation into one of three groups.

The task is to estimate a MANOVA using the four dependent variables and species as the independent variable. Then we test whether there are significant differences among species using the multivariate test statistics, e.g., Pillai's Trace, Wilks' Lambda.

Problem 2.

In this exercise, we analyze a dataset on teenage gambling behavior using both regression and ANOVA techniques. The dataset was originally obtained from an R library.

The data frame contains the following variables:

Dependent Variable

- Gamble: Annual expenditure on gambling (measured in pounds per year)

Categorical Variable (Factor): Sex:

- 0 = Male
- 1 = Female

Covariates (Continuous Variables)

- Status: Socioeconomic status score based on parents' occupation
- Income: Weekly income (in pounds)
- Verbal: Verbal ability score (number of words correctly defined out of 12)

a. Estimate a multiple regression model using Gamble as the dependent variable and Sex, Status, Income, and Verbal as explanatory variables. Interpret the results.

b. Perform an ANOVA-style analysis by treating Sex as the factor and Status, Income, and Verbal as covariates, i.e., estimate an ANCOVA model. Compare the results with those from part (a).

Problem 3.

In this exercise, we will use discriminant analysis to classify borrowers as either likely to default or unlikely to default on a loan. A bank wants to evaluate loan applications in order to reduce the risk of default. Each observation in the dataset represents one borrower.

The dependent variable (group) is the borrower's default:

- 1 = Default (high risk)
- 0 = No default (low risk)

Some explanatory variables include:

- duration – loan duration (in months)
- amount – loan amount
- age – borrower's age
- installment_rate – percentage of income used for loan payments
- number_credits – number of existing loans

a. Compute summary statistics for the explanatory variables. Compare borrowers who default with those who do not.

b. Select one key variable and visualize its distribution using a histogram. Compare the distributions across the two groups.

c. Estimate a discriminant model using Linear Discriminant Analysis (LDA).

- Split the data into training (80%) and testing (20%) sets
- Use the model to classify borrowers
- Construct and interpret a confusion matrix

Problem 4.

A retail company wants to better understand its customers in order to improve marketing strategies. The company collected data on customer spending behavior but does not know how customers should be grouped.

Our task is to use cluster analysis to identify natural groupings of customers.

Each observation represents one customer. The variables include:

- Age – customer's age
- Annual Income (k$) – annual income in thousands of dollars
- Spending Score (1–100) – a score assigned by the store based on spending behavior

a. We explore the data, display the first five observations, and compute summary statistics.

b. We create a scatter plot of annual income vs spending score. What patterns do we observe?

c. We determine the number of clusters. We use the Elbow Method to identify the optimal number of clusters.

d. We plot the clusters using a scatter plot. We use different colors for each cluster.

e. We describe each cluster in plain language. For example: "High income, high spending." How could the company use these clusters for marketing?

Problem 5.

In this exercise, we will use logistic regression to analyze customer behavior and predict whether a customer will leave a company, called churn. We use a real-world dataset from IBM that is widely used in business analytics and machine learning.

A telecommunications company wants to understand why customers cancel their service. Each observation represents one customer.

The dependent variable is the churn and defined below:

- 1 = Customer leaves
- 0 = Customer stays

These explanatory variables help explain why customers leave and are defined below:

- tenure – number of months the customer has stayed
- MonthlyCharges – monthly bill amount
- TotalCharges – total amount paid
- Contract – type of contract (month-to-month, one year, two year)
- InternetService – type of internet service
- PaymentMethod – how the customer pays
- SeniorCitizen – whether the customer is elderly

Please estimate the logistic regression for churn. Perform the analysis to determine what causes customers to leave this company. The dataset is Chapter_07-churn_data.csv

Chapter 8. Machine Learning and Neural Networks

This chapter introduces neural networks, one of the most powerful and widely used tools in modern data analysis and artificial intelligence. Unlike traditional statistical models, which often rely on strong assumptions and provide easily interpretable coefficients, neural networks are designed to learn complex patterns directly from data [35]. While this flexibility allows them to achieve high predictive accuracy, it also makes them more difficult to interpret. It leads to their common description as "black box" models [36].

We begin with the fundamental building block of neural networks: the perceptron [37]. By understanding how a single artificial neuron processes information, we can gradually build toward more complex network structures. We then introduce activation functions. They allow neural networks to model nonlinear relationships, and explore how multiple neurons can be combined into layers to form deep neural networks.

Next, we turn to practical implementation using Keras, a user-friendly framework for building neural networks in Python. Through hands-on examples, we demonstrate how to construct, train, and evaluate neural networks. In particular, we compare neural networks to familiar models, such as linear and logistic regressions, highlighting both their strengths and limitations.

Finally, we extend our analysis to sequential data using recurrent neural networks (RNNs) and Long Short-Term Memory (LSTM) models. These models are especially useful for time series and other ordered data, where past observations influence future outcomes. By the end of this chapter, we will understand not only how neural networks work, but also when and why they should be used.

8.1 Perceptron

The perceptron is the fundamental building block of neural networks. Neural networks are often described as "black boxes" [36]. This means that, while they can produce highly accurate predictions, it is difficult to interpret exactly how they arrive at those results. Unlike regression models, neural networks do not provide easily interpretable outputs such as coefficients, t-statistics, or p-values. Instead, we input data into the model and receive predictions as output, with limited insight into the internal decision-making process.

Neural networks operate by forming connections between artificial neurons. These connections can be saved and shared. It allows researchers to replicate results. However, modern neural networks may contain thousands or even millions of parameters, making it

impractical to report all of them in a traditional academic paper [35]. As a result, researchers often focus on model performance rather than individual parameter interpretation.

Artificial neural networks (ANNs) are inspired by biology [38]. To understand this idea, consider a biological neuron, shown in Figure 8.1.

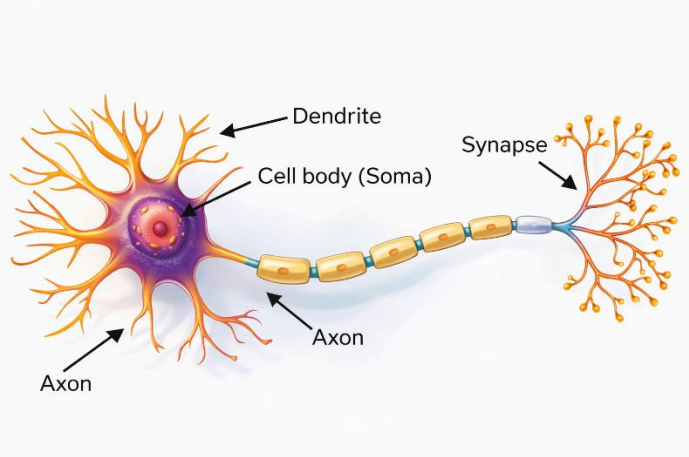


Figure 8.1. Biological Neuron

A biological neuron consists of three main components:

- Dendrites: receive electrical signals from other neurons
- Cell body (soma): processes incoming signals
- Axon: transmits signals to other neurons

At the end of the axon are synapses, which are small gaps between neurons. Chemical signals cross these gaps to activate neighboring neurons. This complex network allows the brain and nervous system to process information efficiently.

We mimic this biological structure using an artificial neuron, called a perceptron, as shown in Figure 8.2.

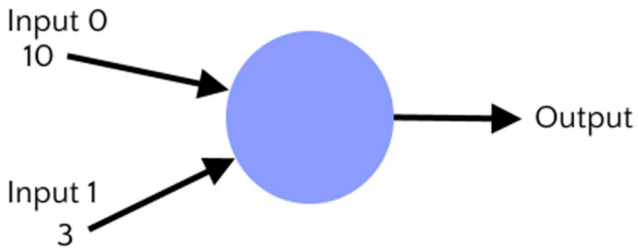


Figure 8.2. Artificial Neuron

The perceptron consists of:

- Inputs (x_1, x_2, ..., x_n): Similar to dendrites
- Weights (w_1, w_2, ..., wn): Strength of each input
- Cell body: Computes a weighted sum
- Output: Sends a signal to the next neuron

For example, suppose we have the following:

Input 1 = 10, weight = 0.6
Input 2 = 3, weight = −0.5

The weighted sum is:

$$0.6 \cdot 10 - 0.5 \cdot 3 = 4.5$$

This value is passed into the neuron in Figure 8.3. The neuron does not automatically pass this value forward. Instead, it uses an activation function to determine the output.

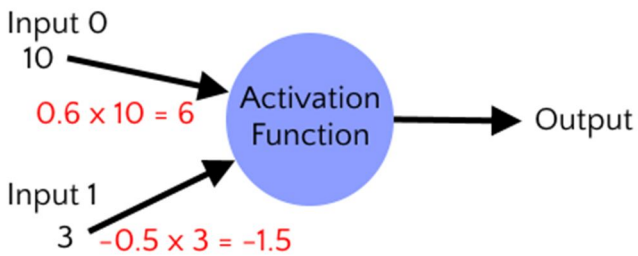


Figure 8.3. Neuron Inputs

A simple activation rule is:

If the weighted sum > 0 → output = 1
If the weighted sum ≤ 0 → output = 0

In our example, the sum is 4.5, so the neuron outputs 1. Thus, it "fires." However, this function is very limited. Small changes in inputs do not affect the output unless the threshold is crossed. This lack of sensitivity makes learning difficult.

What happens if all weights are zero? Then the weighted sum is always zero, and the neuron will never activate. To solve this problem, we introduce a bias term (b). The bias shifts the activation threshold and allows the neuron to activate even when inputs are small or negative.

The perceptron is written mathematically as:

$$z = b + \sum_{i=0}^{n} w_i \cdot x_i$$

where:

- x_i = inputs
- w_i = weights
- b = bias

We show this in Figure 8.4.

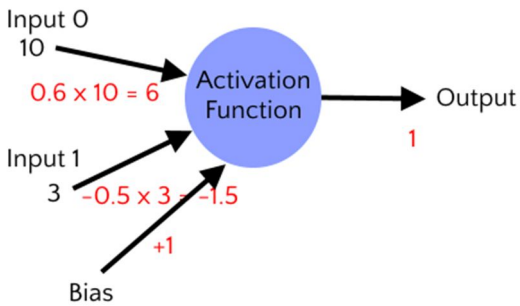


Figure 8.4. Neuron with Bias

This equation should look familiar. It is very similar to a multiple regression model shown below [39]:

$$\beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \cdots + \beta_n \cdot x_n$$

- Bias (b) → Intercept (β_0)
- Weights (w_i) → Slopes (β_i)

The key difference is that neural networks apply an activation function after this linear combination.

We construct a neural network by connecting multiple perceptrons together as shown in Figure 8.5.

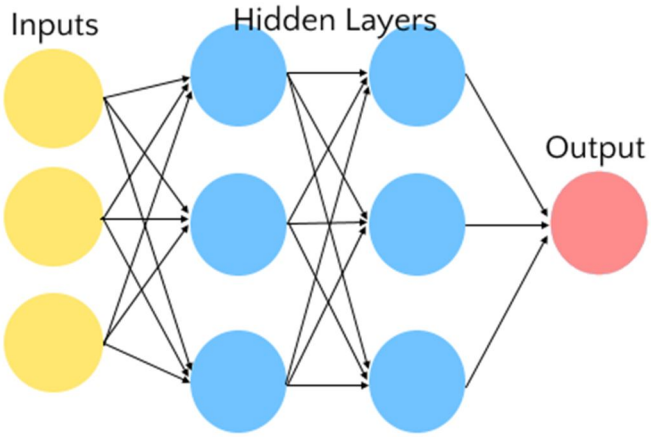


Figure 8.5. Neural Network Structure

- Input layer: Receives raw data
- Hidden layers: Process information through multiple transformations
- Output layer: Produces final predictions

If a network has three or more layers, it is often called a deep neural network.

We discuss the activation function. Our previous activation function was a simple function that output 0 or 1. Remember, the sum of the weights times the input. If the result is positive, the cell outputs a one. If the result is negative, the cell outputs a zero. This is not a good function. Thus, small changes are not reflected. We show the binary activation function in Figure 8.6.

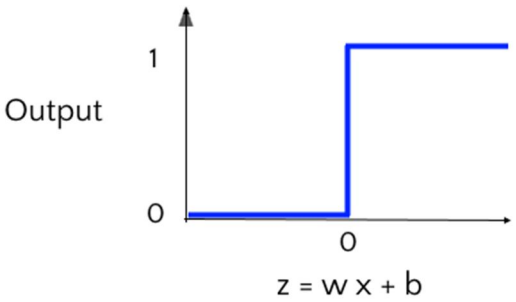


Figure 8.6. Binary Activation Function

The binary function is too rigid. We need smoother functions that allow gradual changes. We can use the sigmoid function as defined below [40]:

$$f(x) = \frac{1}{1 + e^{-x}} = \frac{1}{1 + e^{-wx-b}}$$

- Outputs values between 0 and 1
- Smooth and continuous
- Used in logistic regression in Chapter 7

We show the sigmoid function as the red line in Figure 8.7.

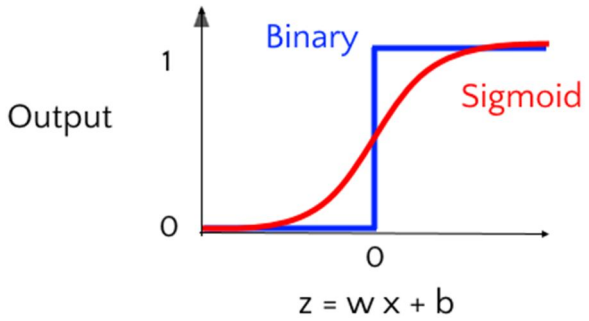


Figure 8.7. Binary vs. Sigmoid Functions

We can use the hyperbolic tangent function, or tanh(x), which closely resembles the sigmoid function. Specifically, the hyperbolic tangent is defined as the hyperbolic sine (sinh) divided by the hyperbolic cosine (cosh). These functions often arise in engineering applications.

$$\tanh x = \frac{\sinh x}{\cosh x}$$

$$\sinh x = \frac{e^x - e^{-x}}{2}$$

$$\cosh x = \frac{e^x + e^{-x}}{2}$$

- Outputs values between −1 and 1
- Centered at zero, which often improves learning

One of the commonly used activation functions is the rectified linear unit (ReLU) [41]. It is a relatively simple function. We define it below with the weights and bias.

$$f(x) = max(0, z) = max(0, wx + b)$$

- If input is negative → output = 0
- If input is positive → output = input

ReLU is widely used because it is:

- Simple
- Computationally efficient
- Effective in deep networks

It behaves similarly to a diode in electronics. It allows current to pass only in one direction. We show the output of a ReLU in Figure 8.8.

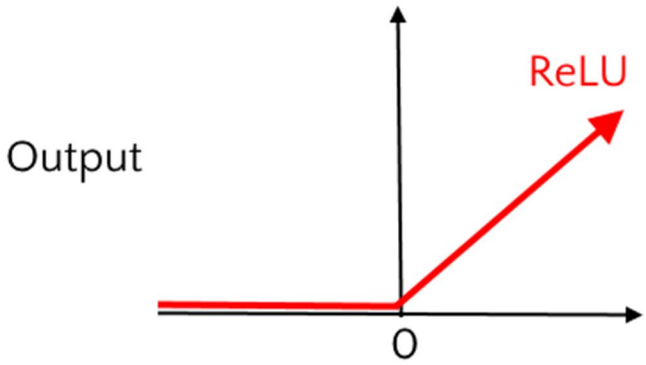


Figure 8.8. Rectified Linear Unit (ReLU) Activation Function

Key Insight

A perceptron combines inputs, weights, and a bias into a weighted sum, then applies an activation function to produce an output. By connecting many perceptrons together, we build powerful neural networks capable of modeling complex relationships.

8.2. Keras Basics

In this section, we introduce Keras, a powerful and user-friendly tool for building neural networks in Python. Keras was developed in 2015 by François Chollet as part of a research initiative known as the Open-ended Neuro-Electronic Intelligent Robot Operating System (ONEIROS) [42].

The name Keras comes from the ancient Greek word kéras, meaning “horn,” which historically symbolized strength and power. This is an appropriate name, as Keras provides a powerful yet accessible way to build deep learning models.

What is Keras? Keras is a free, open-source Python API used for developing neural networks and deep learning models. It is designed with three main goals:

- User-friendly: Simple and intuitive syntax

- Modular: Easy to build and modify models
- Extensible: Works across different platforms and devices

Keras allows us to quickly construct neural networks by stacking layers together and defining how information flows through the model. It runs on top of back-end engines such as TensorFlow, which handle the heavy computational work [43].

To build a neural network in Keras, we typically follow a few key steps:

- Define the model structure
- Add layers (input, hidden, output)
- Compile the model
- Train (fit) the model using data

The most common approach is to use a sequential model. Layers are added one after another in a linear stack. Once the model is defined, we train it using data. Two important concepts are:

Epoch: An epoch is one complete pass through the entire training dataset. If we train for 10 epochs, the model sees the full dataset 10 times. More epochs allow the model to learn better patterns, but too many can lead to overfitting.

Batch: Instead of feeding all data at once, we often divide the dataset into smaller groups called batches.

- Improves computational efficiency
- Helps stabilize learning
- Common in large datasets

To illustrate how Keras works, we will construct a small neural network:

- Input layer: 4 neurons
- Hidden layer: 4 neurons
- Output layer: 1 neuron

This structure is intentionally simple and is useful for learning how neural networks are built. However, it is important to recognize an important limitation: Using a neural network to estimate a simple linear regression is inefficient.

Why?

- Linear regression provides a direct, interpretable solution

- Neural networks require more computation and tuning
- The results are harder to interpret

Why use this example?

Even though this example is inefficient, it serves an important purpose:

- It demonstrates how neural networks are constructed
- It shows how data flows through layers
- It prepares us for more complex models later

We can think of this example as a learning tool, not a practical application.

The Python code is below. The Python code is available at: https://github.com/KenSzulczyk

```
# ---------------------------------------
# PART 1: IMPORT LIBRARIES
# ---------------------------------------

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Keras (modern usage)
from keras.models import Sequential
from keras.layers import Dense, Input

# Evaluation metric
from sklearn.metrics import mean_squared_error

# ---------------------------------------
# PART 2: CREATE SIMULATED DATA
# ---------------------------------------

# We simulate a simple linear relationship:
# y = mx + b + noise

m = 2   # slope
b = 3   # intercept

# Create 100 evenly spaced values between 0 and 50
x = np.linspace(0, 50, 100)

# Set a random seed so results are reproducible
np.random.seed(101)

# Create random noise (mean = 0, standard deviation = 4)
noise = np.random.normal(0, 4, size=len(x))
```

```
# Generate y values
y = m * x + b + noise

# ----------------------------------------
# PART 3: VISUALIZE THE DATA
# ----------------------------------------

plt.scatter(x, y)
plt.title("Simulated Data: y = mx + b + noise")
plt.xlabel("x (Input)")
plt.ylabel("y (Output)")
plt.show()

# ----------------------------------------
# PART 4: BUILD THE NEURAL NETWORK
# ----------------------------------------

# Create a sequential model
model = Sequential()

# Modern Keras: define input layer explicitly
model.add(Input(shape=(1,)))  # One input variable

# Hidden layers
model.add(Dense(4, activation='relu'))  # First hidden layer
model.add(Dense(4, activation='relu'))  # Second hidden layer

# Output layer (for regression)
model.add(Dense(1, activation='linear'))

# Compile the model
# - loss: Mean Squared Error (MSE)
# - optimizer: Adam (efficient training algorithm)
model.compile(loss='mse', optimizer='adam')

# Display model structure
print("\nModel Summary:")
model.summary()

# ----------------------------------------
# PART 5: TRAIN THE MODEL
# ----------------------------------------

# Train the model
# - epochs = number of times the model sees the data
history = model.fit(x, y, epochs=200, verbose=0)

# ----------------------------------------
# PART 6: PLOT THE LOSS FUNCTION
# ----------------------------------------

# Extract loss values
```

```
loss = history.history['loss']

# Create epoch index
epochs = range(len(loss))

# Plot loss over time
plt.plot(epochs, loss)
plt.title("Training Loss (MSE) Over Epochs")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.show()

# ----------------------------------------
# PART 7: MAKE PREDICTIONS
# ----------------------------------------

# Create new input values
x_for_predictions = np.linspace(0, 50, 100)

# Predict output values
y_pred = model.predict(x_for_predictions)

# ----------------------------------------
# PART 8: VISUALIZE RESULTS
# ----------------------------------------

# Plot actual data
plt.scatter(x, y, label="Actual Data")

# Plot neural network predictions
plt.plot(x_for_predictions, y_pred, 'r', label="Neural Network Prediction")

plt.title("Neural Network Fit")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.show()

# ----------------------------------------
# PART 9: EVALUATE MODEL PERFORMANCE
# ----------------------------------------

# Predict using original x values
y_pred_train = model.predict(x)

# Calculate Mean Squared Error
mse = mean_squared_error(y, y_pred_train)

print("\nMean Squared Error:", mse)
```

The randomly generated linear regression is shown in Figure 8.9.

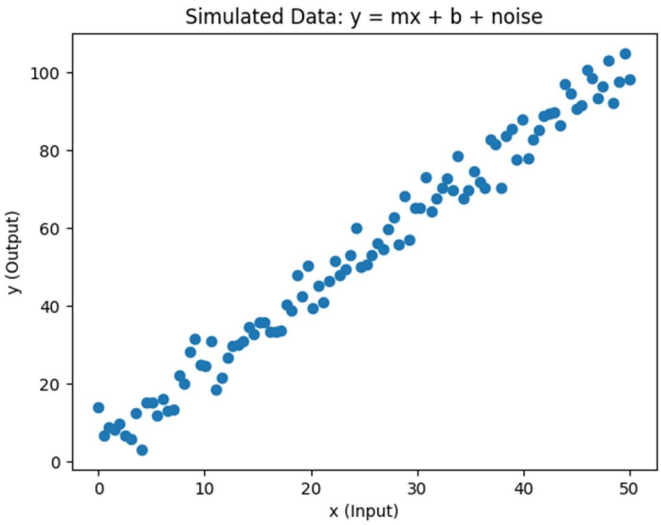


Figure 8.9. Linear Regression Data

The model structure and parameter count are shown in Figure 8.10.

Model Summary:
Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 4)	8
dense_1 (Dense)	(None, 4)	20
dense_2 (Dense)	(None, 1)	5

Total params: 33 (132.00 B)
Trainable params: 33 (132.00 B)
Non-trainable params: 0 (0.00 B)

Figure 8.10. Neural Network for Regression

The plot of the loss function. The rapid decline in Figure 8.11 indicates how quickly the neural network is learning.

The loss curve illustrates how the model improves during training by plotting the mean squared error (MSE) against the number of epochs. At the beginning of training, the loss is typically high because the model's predictions are far from the actual values. As training progresses, the optimizer—such as Adam—updates the model's parameters to reduce the error, causing the loss to decline [44]. Over time, the curve flattens, indicating convergence as further improvements become smaller.

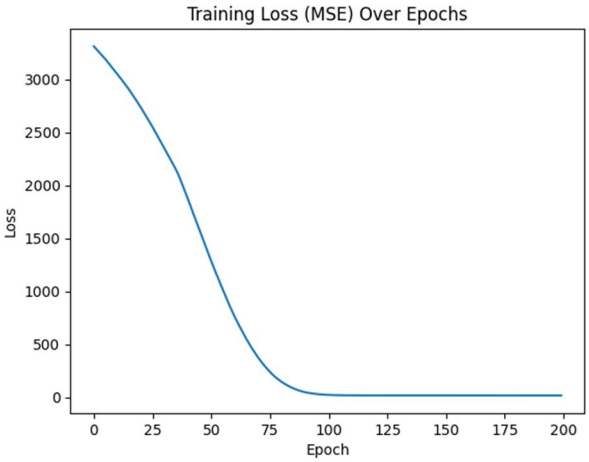


Figure 8.11. Regression Loss Function

This use of MSE is closely related to its role in linear regression, where it also measures the average squared difference between observed and predicted values [39]. However, in regression, MSE is typically computed once from a closed-form solution and used for statistical inference. In neural networks it is evaluated repeatedly and minimized iteratively to guide the learning process.

The neural network's linear regression prediction is shown in Figure 8.12.

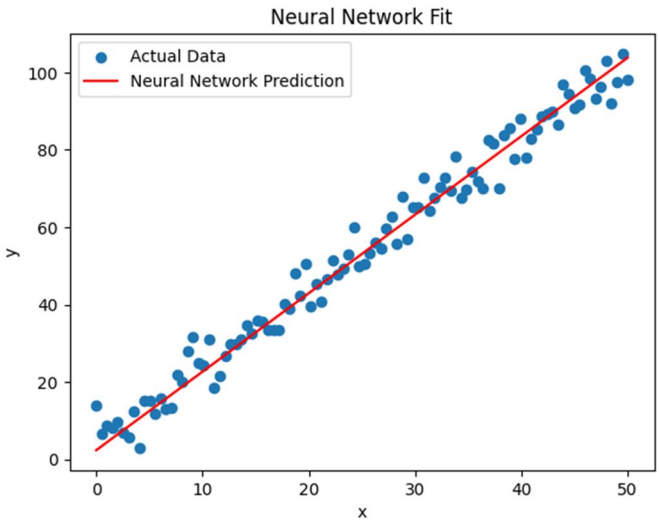


Figure 8.12. Regression Predictions

The mean squared error (MSE) evaluates the model's performance. The output is shown on the next page:

```
Mean Squared Error: 18.08652600410997
```

If we rerun this model and the MSE decreases, we know the model's predictions have improved.

Key Insight

Keras provides a simple and powerful framework for building neural networks. While it can be used for basic models, its true strength lies in handling complex, nonlinear relationships that traditional methods cannot easily capture.

8.3 Case Study: Diabetes Data

In this section, we examine whether individuals from the Pima Indian dataset are likely to develop diabetes. In Chapter 7, we analyzed this same problem using logistic regression. Now, we apply a neural network and compare the results.

The response (dependent) variable is defined as:

- 1 = individual has diabetes
- 0 = individual does not have diabetes

The model uses several health-related variables as inputs:

- Pregnant – number of times pregnant
- Insulin – insulin level in the blood
- BMI (Body Mass Index) – a measure of body fat based on height and weight
- Age – older individuals are generally at higher risk of diabetes
- Glucose – blood glucose concentration (a key indicator of diabetes)
- BP (Blood Pressure) – diastolic blood pressure
- Pedigree – a measure of genetic predisposition to diabetes

These variables capture both biological and genetic risk factors.

Because we have seven input variables, the neural network requires seven input neurons. Unlike logistic regression, neural networks do not have a fixed structure. We must choose the number of hidden layers and neurons. In this example, we use:

- Three hidden layers
- Eight neurons in each layer

- ReLU activation function for all hidden layers

The output layer contains: One neuron with a sigmoid activation function. This is similar to logistic regression, where the model produces a probability between 0 and 1. We classify the outcome using a threshold:

- If probability ≥ 0.5 → predict 1 (diabetes)
- If probability < 0.5 → predict 0 (no diabetes)

The following Python program builds and estimates the neural network. The Python code is available at: https://github.com/KenSzulczyk

```
# ============================================================
# NEURAL NETWORK: DIABETES CLASSIFICATION
# ============================================================
# This version improves accuracy by:
# 1. Using the correct loss function (binary cross-entropy)
# 2. Simplifying the network (reduces overfitting)
# 3. Adding validation data
# 4. Using early stopping to prevent overtraining
# ============================================================

# =========================
# 1. IMPORT LIBRARIES
# =========================
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Scikit-learn tools
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix, mean_squared_error

# Keras (Neural Network)
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Input
from keras.callbacks import EarlyStopping

# =========================
# 2. LOAD AND CLEAN DATA
# =========================

# Load dataset
pima = pd.read_excel("Chapter_07-diabetes.xlsx")

# Convert all data to numeric type
pima = pima.astype(float)
```

```
# =========================
# 3. DEFINE FEATURES (X) AND TARGET (y)
# =========================

# Input variables (features)
feature_cols = ['pregnant', 'insulin', 'bmi', 'age', 'glucose', 'bp', 'pedigree']
X = pima[feature_cols]

# Output variable (0 = no diabetes, 1 = diabetes)
y = pima['outcome']

# =========================
# 4. SPLIT DATA INTO TRAINING AND TESTING
# =========================

# 75% training data, 25% testing data
# random_state ensures results are reproducible
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=16
)

# =========================
# 5. SCALE THE INPUT DATA
# =========================

# Neural networks perform better when inputs are scaled

# IMPORTANT:
# Fit scaler ONLY on training data (to avoid data leakage)
scaler = StandardScaler()
scaler.fit(X_train)

# Apply transformation
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

# =========================
# 6. BUILD THE NEURAL NETWORK
# =========================

# Create a sequential model (layers added one after another)
model = Sequential()

# Define input layer explicitly
model.add(Input(shape=(7,)))

# First hidden layer
model.add(Dense(8, activation='relu'))

# Second hidden layer
model.add(Dense(8, activation='relu'))
```

```
# Third hidden layer
model.add(Dense(8, activation='relu'))

# Output layer (binary classification → sigmoid)
model.add(Dense(1, activation='sigmoid'))

# Compile the model
# Binary cross-entropy is the correct loss for classification
model.compile(
    loss='binary_crossentropy',
    optimizer='adam'
)

# Show model structure
model.summary()

# =========================
# 7. TRAIN THE MODEL
# =========================

# Early stopping prevents overfitting:
# Training stops when validation loss stops improving
early_stop = EarlyStopping(
    monitor='val_loss',          # watch validation loss
    patience=20,                # wait 20 epochs before stopping
    restore_best_weights=True   # keep best model
)

# Train the model
history = model.fit(
    X_train_scaled,
    y_train,
    epochs=200,
    validation_split=0.2,       # 20% of training data used for validation
    callbacks=[early_stop],
    verbose=1
)

# =========================
# 8. PLOT TRAINING PERFORMANCE
# =========================

# Extract loss values
train_loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(train_loss) + 1)

# Plot training vs validation loss
plt.figure()
plt.plot(epochs, train_loss, label="Training Loss")
plt.plot(epochs, val_loss, label="Validation Loss")
plt.xlabel("Epochs")
```

```
plt.ylabel("Loss")
plt.title("Training vs Validation Loss")
plt.legend()
plt.show()

# =========================
# 9. MAKE PREDICTIONS
# =========================

# Predict probabilities (values between 0 and 1)
y_pred_prob = model.predict(X_test_scaled)

# Convert probabilities to class labels (0 or 1)
# Threshold = 0.5
y_pred = (y_pred_prob >= 0.5).astype(int)

# =========================
# 10. EVALUATE MODEL PERFORMANCE
# =========================

# Mean Squared Error (for comparison with earlier models)
mse = mean_squared_error(y_test, y_pred_prob)

# Accuracy (main classification metric)
accuracy = accuracy_score(y_test, y_pred)

# Confusion matrix
conf_m = confusion_matrix(y_test, y_pred)

# =========================
# 11. DISPLAY RESULTS
# =========================

print("Mean Squared Error:", mse)
print("Accuracy:", accuracy)

# Plot confusion matrix
plt.figure(figsize=(6, 6))
sns.heatmap(conf_m, annot=True, fmt="d", cmap="Blues", cbar=False, square=True)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

Figure 8.13 illustrates the structure of the neural network. The network contains:

- 32 neurons across all layers
- 217 parameters to estimate

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 8)	64
dense_1 (Dense)	(None, 8)	72
dense_2 (Dense)	(None, 8)	72
dense_3 (Dense)	(None, 1)	9

Total params: 217 (868.00 B)
Trainable params: 217 (868.00 B)
Non-trainable params: 0 (0.00 B)

Figure 8.13. Neural Network Architecture

These parameters are the weights and biases that the model learns during training.

This neural network includes two important modifications that improve performance:

Binary Cross-Entropy Loss Function: Instead of mean squared error (MSE), we use binary cross-entropy. This loss function is specifically designed for classification problems and measures how well the predicted probabilities match the actual outcomes [40].

Early Stopping: Training stops automatically when the validation loss no longer improves. This prevents the model from overfitting [35]. It occurs when the model memorizes the training data instead of learning general patterns.

To better understand how the neural network learns, we plot both training loss and validation loss.

- Training loss measures how well the model fits the data it is trained on. The model directly adjusts its weights to minimize this loss.
- Validation loss measures how well the model performs on unseen data. This data is not used to update the model, making it a good indicator of how well the model generalizes.

In this example, we reserve 20% of the training data for validation.

Figure 8.14 shows that the neural network learns quickly, and both losses decrease initially. Around epoch 80, the validation loss stops improving. At this point, training stops automatically. This is an important concept: Continuing to train beyond this point can actually reduce model performance because the model begins to overfit the data.

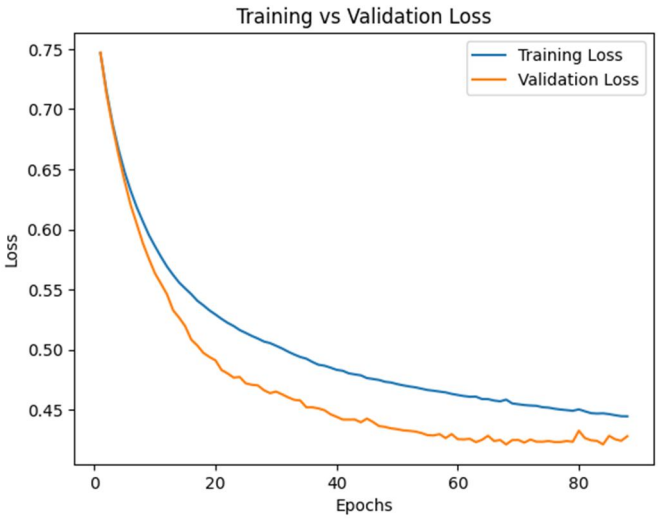


Figure 8.14. Training vs. Validation Loss

The neural network achieves an accuracy of approximately 81%. Figure 8.15 presents the confusion matrix. The results indicate:

- The model performs well in identifying individuals without diabetes
- The model is less accurate in identifying individuals with diabetes

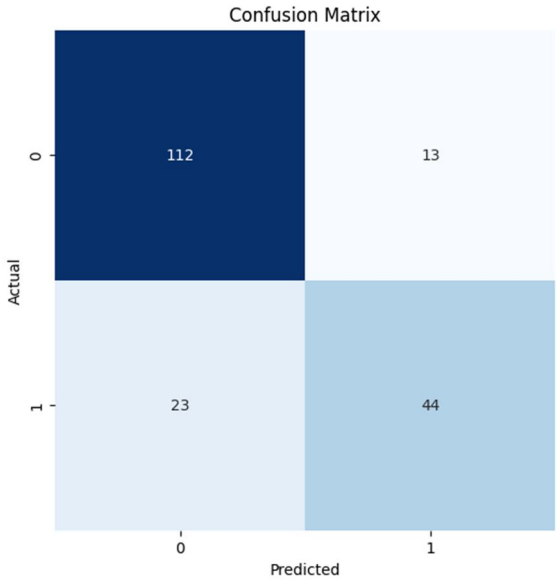


Figure 8.15. Diabetes Confusion Matrix

This is a common issue in medical classification problems, where detecting positive cases is often more difficult [32].

Interestingly, the neural network performs about the same as the logistic regression model from Chapter 7. This provides an important insight: More complex models do not always produce better results.

Neural networks require:

- Larger datasets
- Careful tuning of parameters
- Proper model design

When the dataset is relatively small and relationships are fairly simple, logistic regression can perform just as well or even better.

Important Note for Students: Students may obtain slightly different results when running the program. This is normal because:

- Neural networks start with random initial weights
- The training process is stochastic (randomized)

As a result, accuracy may vary slightly across runs. Some students may obtain higher or lower accuracy than reported here.

Key Insight

This example highlights an important lesson: Neural networks are powerful, but they are not always superior to simpler models. Understanding the data and choosing the appropriate model is more important than using the most complex method.

8.4 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are designed to work with sequential data, where the order of observations matters [35]. Many real-world datasets are naturally sequential. Examples include:

- Time series data – observations ordered over time, e.g., monthly sales
- Sentences – sequences of words in a specific order
- Natural language processing (NLP) – analyzing text such as social media posts or news articles. For example, researchers may construct indices of investor sentiment using posts from Twitter and other platforms. Words like “recession,” “layoffs,” or “bankruptcy” may signal negative sentiment.
- Audio data – sequences of sound waves

- Vehicle trajectories – sequences of positions or directions over time
- Music – sequences of notes and rhythms

Consider a simple numerical sequence:

[1,2,3,4,5,6]

If we shift this sequence forward in time, we obtain:

[2,3,4,5,6,7]

The goal of a neural network in this context is to learn the pattern in the data so that it can predict future values. In other words, the model learns how sequences evolve over time.

In a standard neural network, such as a perceptron, each neuron:

- Receives inputs from other neurons
- Multiplies each input by a weight
- Adds a bias
- Produces an output

However, these models do not remember past information. Each input is treated independently.

A recurrent neuron is different. It feeds its output back into itself. It retains information from previous time steps. This feedback loop gives the network a form of memory. When we visualize this process over time, we often "unroll" the neuron. This means we represent the same neuron at different time steps, connected in a sequence. Because of this memory feature, recurrent neurons are often called memory cells.

Recurrent neuron networks (RNNs) are flexible: they can process sequences as inputs and produce either a single output or a sequence of outputs.

Instead of using a single recurrent neuron, we typically use a layer of recurrent neurons.

- The input X feeds into the layer
- The neurons produce an output y
- This output is fed back into the layer for the next time step

When we input a sequence, the network processes it step by step. The output at each step depends not only on the current input, but also on all previous inputs. In this way, the network builds a memory of the sequence. We show this in Figure 8.16.

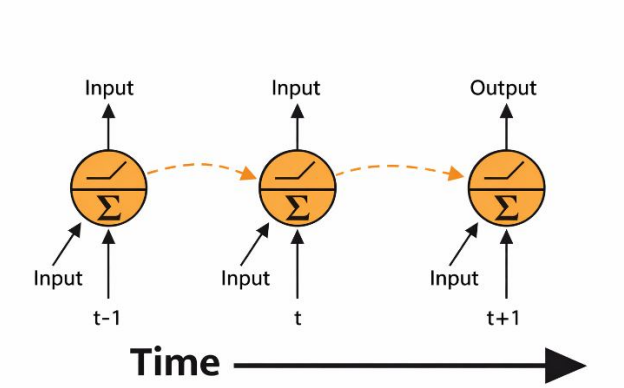


Figure 8.16. Recurrent Neuron

RNNs are very flexible. They can handle different types of input and output structures:

1. Sequence → Single Output
   - Input: a sequence, e.g., words in a product review
   - Output: a single value, e.g., positive or negative sentiment

2. Sequence → Sequence
   - Input: a sequence, e.g., time series data
   - Output: another sequence, e.g., future forecasts

3. Single Input → Sequence Output
   - Input: a single value or word, e.g., "Hello"
   - Output: a sequence, e.g., "How are you?"

Although RNNs can remember past information, they have an important limitation: As the sequence becomes longer, the network gradually "forgets" earlier inputs. This happens because information is lost as it passes through many time steps. As a result, basic RNNs struggle to capture long-term dependencies [45].

To solve this problem, researchers developed the Long Short-Term Memory (LSTM) network [46]. An LSTM is a special type of RNN that is designed to remember information over long periods of time. We show the LSTM in Figure 8.17.

Each LSTM cell includes:
- Current input X_t
- Previous hidden state h_{t-1}
- Cell state C_t, which acts as long-term memory

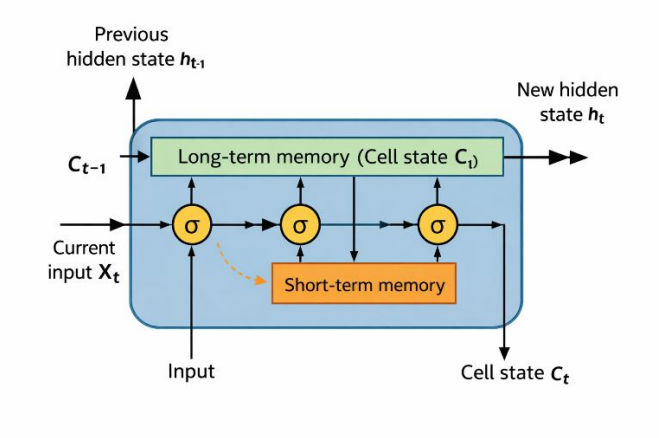


Figure 8.17. Long Short-Term Memory (LSTM) Cell

The key idea is that the LSTM can decide what to remember and what to forget. This allows it to preserve important information over many time steps. While the mathematical details are complex, modern libraries such as Keras make LSTMs easy to implement.

We now apply an LSTM model to forecast alcohol sales. The data are obtained from the Federal Reserve Bank of St. Louis (FRED), which provides a large collection of economic time series.

For cross-sectional data, we can randomly split observations into training and testing sets. However, this approach does not work for time series data, because the order of observations matters.

Instead, we:

- Use earlier observations for training
- Use later observations for testing

In this example:

- Training data: observations up to index 384
- Testing data: the remaining observations, the last 24 months

This approach mimics real-world forecasting.

Neural networks perform better when data are scaled [47]. Two common methods are:

- Standardization (mean = 0, standard deviation = 1)
- Min–max scaling (values between 0 and 1)

In this example, we use min–max scaling:

$$Scaled\ value = \frac{x - min}{max - min}$$

This ensures that all inputs are on the same scale, improving model performance. The Python code is below. The Python code is available at: https://github.com/KenSzulczyk

```
# ============================================================
# IMPROVED LSTM NEURAL NETWORK: ALCOHOL SALES FORECASTING
# ============================================================
# Improvements:
# 1. Uses 24 months of history (captures longer patterns)
# 2. Uses tanh activation (better for LSTM)
# 3. Adds a second LSTM layer (deeper learning)
# 4. Uses dropout (reduces overfitting)
# 5. Uses early stopping (prevents overtraining)
# 6. Evaluates model using RMSE
# ============================================================

# =========================
# 1. IMPORT LIBRARIES
# =========================
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from statsmodels.tsa.seasonal import seasonal_decompose

from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout
from keras.layers import Input
from keras.callbacks import EarlyStopping

from tensorflow.keras.preprocessing.sequence import TimeseriesGenerator
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error

# =========================
# 2. LOAD AND PREPARE DATA
# =========================

df = pd.read_csv(
    'Chapter_08-alcohol_sales.csv',
    index_col='Date',
    parse_dates=True
)

df.index.freq = 'MS'
```

```
df.columns = ['Sales']

print(df.head())

# =========================
# 3. VISUALIZE DATA
# =========================

df.plot(figsize=(12, 8), title="Alcohol Sales Over Time")
plt.show()

# We decompose the sales into the long term trend and the seasonal variation
results = seasonal_decompose(df['Sales'])
# We plot the data to look at it
# Add a semicolon if we get two graphs
results.plot()
plt.show()

# =========================
# 4. TRAIN-TEST SPLIT
# =========================

train = df.iloc[:384].copy()
test = df.iloc[384:].copy()

# =========================
# 5. SCALE DATA
# =========================

scaler = MinMaxScaler()
scaler.fit(train)

scaled_train = scaler.transform(train)
scaled_test = scaler.transform(test)

# =========================
# 6. TIME SERIES GENERATOR
# =========================

# Use 24 months instead of 12
n_input = 24
n_features = 1

train_generator = TimeseriesGenerator(
    scaled_train,
    scaled_train,
    length=n_input,
    batch_size=10
)

# Validation generator (for monitoring performance)
test_generator = TimeseriesGenerator(
    scaled_test,
```

```
    scaled_test,
    length=n_input,
    batch_size=1
)

# =========================
# 7. BUILD IMPROVED LSTM MODEL
# =========================

model = Sequential()

# Define input layer (24 months, 1 feature)
model.add(Input(shape=(n_input, n_features)))

# First LSTM layer
model.add(LSTM(
    36,
    activation='tanh',
    return_sequences=True
))

# Second LSTM layer
model.add(LSTM(36, activation='tanh'))

# Output layer
model.add(Dense(1))

# Compile model
model.compile(
    optimizer='adam',
    loss='mse'
)

model.summary()

# =========================
# 8. TRAIN MODEL (WITH EARLY STOPPING)
# =========================

early_stop = EarlyStopping(
    monitor='val_loss',
    patience=10,
    restore_best_weights=True
)

history = model.fit(
    train_generator,
    epochs=100,
    validation_data=test_generator,
    callbacks=[early_stop],
    verbose=1
)
```

```
# ========================
# 9. PLOT TRAINING PERFORMANCE
# ========================

train_loss = history.history['loss']
val_loss = history.history['val_loss']

plt.figure()
plt.plot(train_loss, label="Training Loss")
plt.plot(val_loss, label="Validation Loss")
plt.title("Training vs Validation Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()

# ========================
# 10. MAKE FORECASTS
# ========================

test_predictions = []

first_eval_batch = scaled_train[-n_input:]
current_batch = first_eval_batch.reshape((1, n_input, n_features))

for i in range(len(test)):

    current_pred = model.predict(current_batch, verbose=0)[0]
    test_predictions.append(current_pred)

    current_batch = np.append(
        current_batch[:, 1:, :],
        [[current_pred]],
        axis=1
    )

# ========================
# 11. INVERSE TRANSFORM
# ========================

true_predictions = scaler.inverse_transform(test_predictions)

# ========================
# 12. EVALUATE MODEL
# ========================

test['Predictions'] = true_predictions

# Root Mean Squared Error (best for forecasting)
rmse = np.sqrt(mean_squared_error(test['Sales'], test['Predictions']))

print("RMSE:", rmse)
```

```
# ========================
# 13. PLOT RESULTS
# ========================

test.plot(figsize=(10, 6), title="Actual vs Predicted Alcohol Sales")
plt.show()

# ========================
# 14. SAVE / LOAD MODEL
# ========================

# model.save('Improved_LSTM_Model.h5')

# from keras.models import load_model
# new_model = load_model('Improved_LSTM_Model.h5')
```

Figure 8.18 shows the monthly alcohol sales data. Two important patterns are visible:

- A long-term upward trend
- A seasonal pattern that repeats each year

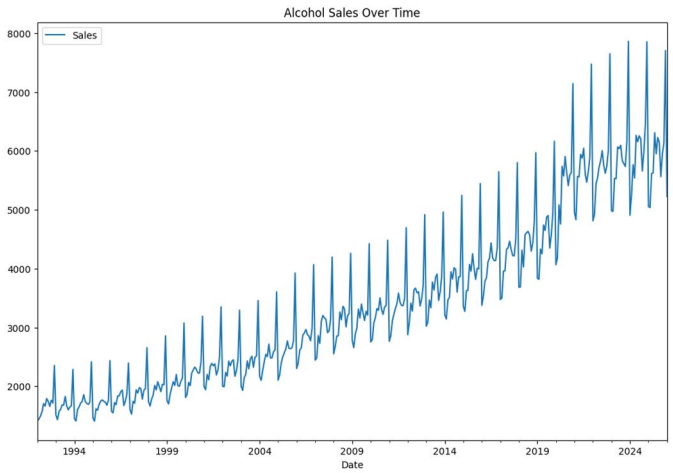


Figure 8.18. Monthly Alcohol Sales

To better understand the data, we decompose the series into three components:

- Trend: long-term movement over time
- Seasonality: repeating patterns within each year
- Residual (noise): random fluctuations

Python treats each component as an object, allowing us to analyze them separately.

Figure 8.19 shows that alcohol sales increased sharply after 2019, which may reflect changes during the COVID-19 pandemic.

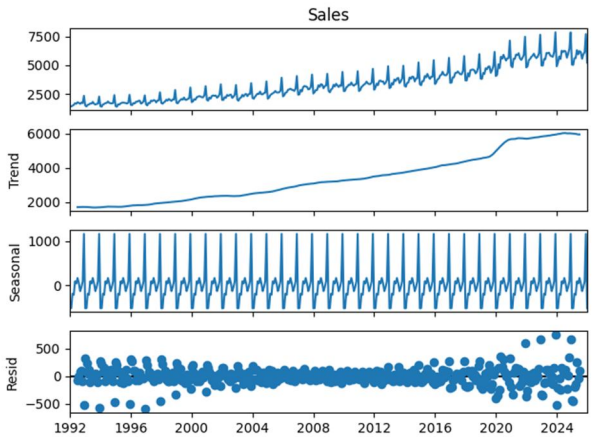


Figure 8.19. Alcohol Sales Decomposition

Figure 8.20 shows the model summary. The network contains 16,021 parameters, which are estimated during training.

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 24, 36)	5,472
lstm_1 (LSTM)	(None, 36)	10,512
dense (Dense)	(None, 1)	37

Total params: 16,021 (62.58 KB)
Trainable params: 16,021 (62.58 KB)
Non-trainable params: 0 (0.00 B)

Figure 8.20. Neural Network Architecture

Figure 8.21 plots the training and validation loss: Training loss remains low and stable. Validation loss fluctuates, indicating some difficulty in generalizing.

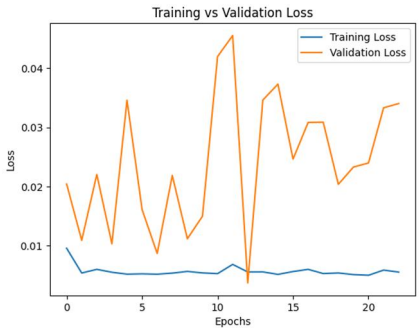


Figure 8.21. Training vs. Validation Loss (Alcohol Sales)

Figure 8.22 shows the forecast from January 2024 to January 2026. The model successfully captures the long-term upward trend. However, it struggles to fully capture the seasonal fluctuations. This is a common challenge in time series forecasting, especially when seasonal patterns are strong.

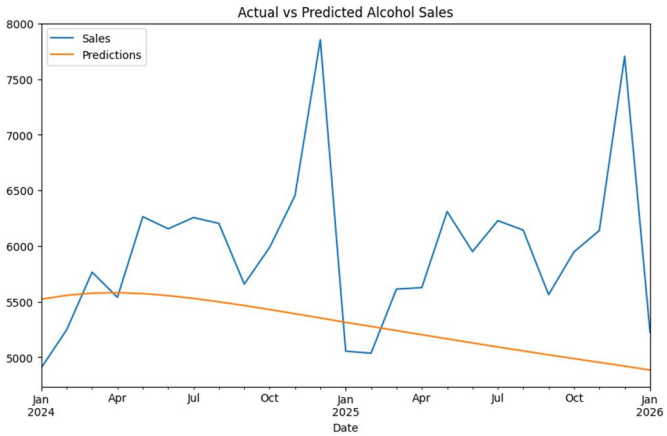


Figure 8.22. Alcohol Sales Forecast (2024 - 2026)

The final part of the code allows us to save the trained model. This is useful for:

- Reproducing results
- Sharing models with other researchers
- Avoiding the need to retrain the model

The model is saved as a .h5 file and can be reloaded later.

```
# ==========================
# 14. SAVE / LOAD MODEL
# ==========================

model.save('Improved_LSTM_Model.h5')

from keras.models import load_model
new_model = load_model('Improved_LSTM_Model.h5')
```

Key Insight

LSTM models are powerful because they can capture long-term dependencies in sequential data. However, they do not automatically capture all patterns, especially seasonality, without careful design and tuning.

8.5 Summary

In this chapter, we explored the foundations and applications of neural networks. We began with the perceptron. We show how inputs, weights, and biases combine to produce outputs through activation functions. By connecting multiple perceptrons, we constructed neural networks capable of modeling complex and nonlinear relationships.

We then introduced Keras as a practical tool for building and training neural networks. Through examples, we demonstrated how neural networks learn by minimizing a loss function over multiple epochs and how performance can be evaluated using metrics such as mean squared error and accuracy. Importantly, we saw that neural networks are not always superior to simpler models. In some cases, such as the diabetes example, logistic regression performed just as well. Thus, we emphasize the importance of choosing the appropriate model for the data.

The chapter concluded with recurrent neural networks and LSTM models. They extend neural networks to sequential data. These models allow us to capture patterns over time, making them especially useful for forecasting and natural language processing. However, we also observed that even advanced models like LSTM may struggle to capture all aspects of the data, such as strong seasonal patterns, without careful design and tuning.

The key lesson from this chapter is that neural networks are powerful tools, but they require thoughtful application [35]. Successful modeling depends not only on the complexity of the method, but also on understanding the structure of the data, selecting appropriate features, and carefully evaluating model performance.

8.6 Exercises

Problem 1.

In this exercise, we will use a simple neuronal network to analyze customer behavior and predict whether a customer will leave a company, called churn. We use a real-world dataset from IBM that is widely used in business analytics and machine learning. We used a logistic regression to estimate this problem in Chapter 7.

A telecommunications company wants to understand why customers cancel their service. Each observation represents one customer.

The dependent variable is the churn and defined below:

- 1 = Customer leaves
- 0 = Customer stays

These explanatory variables help explain why customers leave and are defined below:

- tenure – number of months the customer has stayed
- MonthlyCharges – monthly bill amount
- TotalCharges – total amount paid
- Contract – type of contract (month-to-month, one year, two year)
- InternetService – type of internet service
- PaymentMethod – how the customer pays
- SeniorCitizen – whether the customer is elderly

Please train a neuronal network to predict churn. The dataset is Chapter_07-churn_data.csv

Problem 2.

Inflation remains an important macroeconomic concern in the United States. In this exercise, we will use a neural network to model and forecast inflation dynamics. Using monthly data from FRED, construct a time series of the U.S. inflation rate (e.g., based on the Consumer Price Index).

This task requires the following:

- Build a feedforward neural network or LSTM model to forecast inflation.
- Use the model to generate out-of-sample forecasts for the last two years of the dataset.
- Ensure that these forecasts are truly out-of-sample, i.e., not used in training.
- Plot the actual vs. predicted inflation rates over the forecast period. Please comment on the forecast.

References

[1] Freedman, D., Pisani, R., & Purves, R. (2007). *Statistics* (4th ed.). W. W. Norton & Company.

[2] Stevens, S. S. (1946). On the theory of scales of measurement. *Science*, 103(2684), 677–680.

[3] Wooldridge, J. M. (2020). *Introductory econometrics: A modern approach* (7th ed.). Cengage Learning.

[4] Casella, G., & Berger, R. L. (2002). *Statistical inference* (2nd ed.). Duxbury.

[5] Neyman, J., & Pearson, E. S. (1933). On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society A*, 231(694–706), 289–337.

[6] Pearson, K. (1896). Mathematical contributions to the theory of evolution. *Philosophical Transactions of the Royal Society A*, 187, 253–318.

[7] de Moivre, A. (1733). *The doctrine of chances*. London.

[8] Lindeberg, J. W. (1922). Eine neue Herleitung des Exponentialgesetzes in der Wahrscheinlichkeitsrechnung. *Mathematische Zeitschrift*, 15, 211–225.

[9] Gosset, W. S. (1908). The probable error of a mean. *Biometrika*, 6(1), 1–25.

[10] Pearson, K. (1900). On the criterion that a given system of deviations… *Philosophical Magazine*, 50(302), 157–175.

[11] Fisher, R. A. (1925). *Statistical methods for research workers*. Oliver & Boyd.

[12] Montgomery, D. C. (2019). *Design and analysis of experiments* (10th ed.). Wiley.

[13] Kutner, M. H., Nachtsheim, C. J., Neter, J., & Li, W. (2005). *Applied linear statistical models* (5th ed.). McGraw-Hill.

[14] Fisher, R. A. (1935). *The design of experiments*. Oliver & Boyd.

[15] Gujarati, D. N., & Porter, D. C. (2009). *Basic econometrics* (5th ed.). McGraw-Hill.

[16] Stock, J. H., & Watson, M. W. (2020). *Introduction to econometrics* (4th ed.). Pearson.

[17] Greene, W. H. (2018). *Econometric analysis* (8th ed.). Pearson.

[18] Guido van Rossum, & Drake, F. L. (2009). *Python 3 reference manual*. CreateSpace.

[19] McKinney, W. (2017). *Python for data analysis* (2nd ed.). O'Reilly Media.

[20] Google. (2023). Google Colaboratory. https://colab.research.google.com/

[21] Python Software Foundation. (2026). *Python documentation*. https://docs.python.org/3/

[22] Oliphant, T. E. (2006). *A guide to NumPy*. Trelgol Publishing.

[23] Hunter, J. D. (2007). *Matplotlib: A 2D graphics environment*. Computing in Science & Engineering, 9(3), 90–95.

[24] Montgomery, D. C., Peck, E. A., & Vining, G. G. (2012). *Introduction to linear regression analysis* (5th ed.). Wiley.

[25] Draper, N. R., & Smith, H. (1998). *Applied regression analysis* (3rd ed.). Wiley.

[26] Tabachnick, B. G., & Fidell, L. S. (2019). *Using multivariate statistics* (7th ed.). Pearson.

[27] Johnson, R. A., & Wichern, D. W. (2018). *Applied multivariate statistical analysis* (6th ed.). Pearson.

[28] Hair, J. F., Black, W. C., Babin, B. J., & Anderson, R. E. (2019). *Multivariate data analysis* (8th ed.). Cengage.

[29] Stevens, J. P. (2012). *Applied multivariate statistics for the social sciences* (5th ed.). Routledge.

[30] Field, A. (2024). *Discovering statistics using IBM SPSS statistics* (5th ed.). Sage.

[31] Rutherford, A. (2011). *ANOVA and ANCOVA: A GLM approach*. Wiley.

[32] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *Linear model selection and regularization*. New York, NY: Springer US.

[33] Everitt, B. S., Landau, S., Leese, M., & Stahl, D. (2011). *Cluster analysis* (5th ed.). Wiley.

[34] Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression* (3rd ed.). Wiley.

[35] Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning* (Vol. 1, No. 2, pp. 1-800). Cambridge: MIT press.

[36] Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5), 206-215.

[37] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6), 386.

[38] McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4), 115-133.

[39] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning*. New York: Springer

[40] Bishop, C. M., & Nasrabadi, N. M. (2006). *Pattern recognition and machine learning* (Vol. 4, No. 4, p. 738). New York: Springer.

[41] Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In Proceedings of the *27th international conference on machine learning* (ICML-10) (pp. 807-814).

[42] Chollet, F. (2015). *Keras*. https://keras.io

[43] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zheng, X. (2016). {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation* (OSDI 16), pp. 265-283.

[44] Kingma, D. P., & Ba, J. (2014). Adam: *A method for stochastic optimization*. arXiv preprint arXiv:1412.6980.

[45] Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2), 157-166.

[46] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735-1780.

[47] Géron, A. (2022). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. "O'Reilly Media, Inc."

Answers to Exercises

Chapter 1 Answers

Problem 1.

An investor is deciding whether to diversify their portfolio by investing in two sectors: technology and energy. The annual returns (%) for two representative stocks over five years are given in the table below:

Year	Tech Stock (%)	Energy Stock (%)
1	10	4
2	18	12
3	-6	-3
4	20	8
5	9	7

a. Please compute the following for each stock: minimum, mean, maximum, and standard deviation.

We calculate the descriptive statistics in the table below.

Year	Tech Stock (%)	Energy Stock (%)	Dev. Tech	Dev. Energy	Cov
1	10	4	-0.2	-1.6	0.32
2	18	12	7.8	6.4	49.92
3	-6	-3	-16.2	-8.6	139.32
4	20	8	9.8	2.4	23.52
5	9	7	-1.2	1.4	-1.68
Minimum	-6	-3			
Average	10.2	5.6			
Maximum	20	12			
Variance			105.2	31.3	
Std. Dev.			10.3	5.6	
Cov					52.85
Correlation					0.92

b. Compute the correlation coefficient between the two stocks.

The Pearson correlation is calculated in the table in a. and equals 0.92.

c. Based on your results:

- Which stock offers higher average returns?
- Which stock is riskier?
- Does diversification appear beneficial?

The Tech stock offers higher average returns but is also more risky, as indicated by its higher standard deviation. The Tech and Energy stocks are highly positively correlated. Therefore, diversification would provide limited benefits, since the returns tend to move together.

Problem 2.

A national survey reports that adults spend an average of 6.5 hours per day on screens. A researcher suspects that remote workers spend more time on screens than this average. A sample of 36 remote workers shows an average screen time of 7.1 hours per day. Assume the population standard deviation is 1.8 hours.

a. State the null and alternative hypotheses.

H_o:$\mu \leq 6.5$
H_a: $\mu > 6.5$

b. Compute the test statistic.

$$z = \frac{\bar{x} - \mu}{\sigma/\sqrt{n}} = \frac{7.1 - 6.5}{1.8/\sqrt{36}} = 2.0$$

c. Compute the p-value.

p-value =0.023 by using Excel formula below:

=1-NORM.DIST(2,0,1,1) = 0.023

d. At $\alpha = 0.05$, state your conclusion.

Using 95% confidence, we reject the null hypothesis and conclude that remote workers spend more time on screens than the national average.

e. Repeat the test using the critical value approach.

Since the calculated z exceeds the critical value of 1.64, we reject the null hypothesis. We conclude that remote workers spend more time on screens than the national average.

Problem 3.

A technology company wants to compare the average battery life of two smartphone models.

Model A uses a new battery design
Model B uses the current standard battery

The company wants to know whether the mean battery life for the new design lasts longer than the standard battery. The table contains the data.

Model A (New Battery)	Model B (Standard Battery)
$n_1 = 36$	$n_2 = 49$
$\bar{x}_1 = 11.8$ hours	$\bar{x}_2 = 10.9$ hours
$\sigma_1 = 2.4$ hours	$\sigma_2 = 2.8$ hours

a. State the hypotheses

H_0: $\mu_1 \leq \mu_2$
H_0: $\mu_1 > \mu_2$

b. Compute the test statistic

$$\sigma_p = \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}} = \sqrt{\frac{2.4^2}{36} + \frac{2.8^2}{49}} = 0.566$$

$$z = \frac{\bar{x}_1 - \bar{x}_2 - (\mu_1 - \mu_2)}{\sigma_p} = \frac{11.8 - 10.9 - 0}{0.566} = 1.590$$

c. Compute the p-value

=1-NORM.DIST(1.590,0,1,1) = 0.056

d. What is our conclusion using $\alpha = 0.05$?

Since the p-value exceeds the α, we fail to reject the null hypothesis. We do not have sufficient evidence to conclude that the two models have different battery lives.

Problem 4.

A university wants to compare the average exam scores of two teaching methods.

Method A uses traditional lectures
Method B uses interactive learning

The university wants to determine whether the mean scores differ between the two methods.

Teaching Method Data

Method A (Traditional)	Method B (Interactive)
$n_1 = 25$	$n_2 = 30$
$\bar{x}_1 = 78.6$	$\bar{x}_2 = 74.2$
$s_1 = 10.5$	$s_2 = 9.8$

a. State the hypotheses

$H_0: \mu_1 = \mu_2$
$H_0: \mu_1 \neq \mu_2$

b. Compute the test statistic

$$s_p = \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}} = \sqrt{\frac{10.5^2}{25} + \frac{9.8^2}{30}} = 2.759$$

$$df = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{1}{n_1 - 1}\left(\frac{s_1^2}{n_1}\right)^2 + \frac{1}{n_2 - 1}\left(\frac{s_2^2}{n_2}\right)^2} = \frac{\left(\frac{10.5^2}{25} + \frac{9.8^2}{30}\right)^2}{\frac{1}{24}\left(\frac{10.5^2}{25}\right)^2 + \frac{1}{29}\left(\frac{9.8^2}{30}\right)^2} = \frac{61.889}{0.810 + 0.353} = 53.2$$

$$t = \frac{\bar{x}_1 - \bar{x}_2 - (\mu_1 - \mu_2)}{s_p} = \frac{78.6 - 74.2 - 0}{2.759} = 1.595$$

c. Compute the p-value.

We use the Excel formula below to compute the p-value.

$$=T.DIST.RT(1.595,53) = 0.058$$

d. What is our conclusion using $\alpha = 0.05$?

We fail to reject the null hypothesis. We conclude that the two teaching methods do not differ.

Problem 5.

A retail company is working to improve the consistency of its customer experience. Management believes that reducing variability in customer satisfaction is just as important as increasing the average score.

Customer satisfaction is measured on a 100-point scale. Based on past data, the company expects a population standard deviation of $\sigma = 10$ or lower.

After implementing a new employee training program, the company collects a sample of 15 customer satisfaction scores:

88	84	91	79	85
87	82	90	86	83
89	81	92	78	84

a. What is the sample mean customer satisfaction score?

$$\bar{x} = 85.27$$

b. What is the sample variance?

$$s^2 = 18.21$$

c. What is the sample standard deviation?

$$s = 4.27$$

d. Conduct a hypothesis test to determine whether the variability has changed after the training program. Use a significance level of 0.05.

- State the null and alternative hypotheses

We have to convert the standard deviation into a variance:

H_o: $\sigma^2 \leq 100$
H_a: $\sigma^2 > 100$

- Compute the test statistic

$$\chi^2 = \frac{(n-1)\cdot s^2}{\sigma^2} = \frac{(15-1)\cdot 18.21}{100} = 2.549$$

- State the decision rule

Reject H_o if $\chi^2 > 23.685$. Since $2.549 < 23.685$, the test statistic falls in the non-rejection region. We use Excel to get the critical chi-square value:

=CHISQ.INV.RT(0.05,14) = 23.685

- Provide your conclusion

We fail to reject the null hypothesis at the 5% significance level. There is insufficient evidence to conclude that the population variance is greater than 100.

Problem 6.

A manufacturing company operates two production lines that produce the same product. Management wants to ensure that both lines maintain consistent quality. In particular, they are concerned about variability in product thickness, since high variability can lead to defects.

To evaluate consistency, the company collects samples from each production line:

Production Line 1:	$n_1 = 21$,	$s_1^2 = 8.2$	$df_1 = 20$
Production Line 2:	$n_2 = 26$,	$s_2^2 = 3.0$	$df_2 = 25$

Management wants to test whether the variability of the two production lines is the same. The hypotheses are below:

H_o: $\sigma_1^2 = \sigma_2^2$
H_a: $\sigma_1^2 \neq \sigma_2^2$

a. Use the p-value approach to test the hypotheses at the 5% significance level.

Hint: Place the larger sample variance in the numerator and conduct a right-tailed test. Remember to use half the α in the upper tail.

$$F = \frac{8.2}{3.0} = 2.733$$

The p-value is less than 0.05. Therefore, we reject the null hypothesis and conclude that the variability between the two production lines differs. We get the p-value from Excel:

=2*F.DIST.RT(2.733,20,25) = 0.019

b. Repeat the test using the critical value approach.

The critical F-value is 2.300. Since the calculated F-statistic exceeds the critical value, we reject the null hypothesis. Each sample "loses" one degree of freedom because we estimate both the mean and variance. The following Excel formula was used:

=F.INV.RT(0.025,20,25) = 2.300

Chapter 2 Answers

Problem 1

A researcher wants to examine whether different study environments affect student performance. The experiment includes one factor:

Factor (Study Environment):

A1 : Quiet Room
A2 : Coffee Shop
A3 : Library

The response variable is test score. Each environment is tested with five students. The table includes the average and sample variance for each study location. The grand mean equals 80.07.

Table: Exam Scores for Study Environment

	Quiet Room	Coffee Shop	Library
	78	81	80
	82	79	81
	80	83	79
	79	80	82
	81	78	78
Sample Mean	80.0	80.2	80.0
Sample Variance	2.5	3.7	2.5

Please calculate the ANOVA and determine whether the study place affects a student's test scores.

Table: The ANOVA for Exam Scores and Study Environment

ANOVA Source	df	SS	MS	F	p-value
SSTR	2	0.133	0.067	0.02	0.977
SSE	12	34.800	2.900		
SST	14	34.93			

We fail to reject the null hypothesis and conclude that the study environment does not influence exam scores.

Problem 2

Four machines are tested. Five operators serve as blocks. The table shows the data.

Table: Five Operators and Four Machines

Operator	M1	M2	M3	M4	Block Average
1	72	80	74	69	73.750
2	75	82	76	70	75.750
3	73	81	75	71	75.000
4	74	83	77	72	76.500
5	76	84	78	73	77.750
Treatment Average	74.000	82.000	76.000	71.000	
Treatment Variance	2.500	2.500	2.500	2.500	
Treatment Count	5	5	5	5	
Overall Average	75.750				
Overall Variance	19.145				

Overall Count	20

a. Construct an analysis of variance table. What is your conclusion using α=0.1?

Table: ANOVA for Operators and Machines

Analysis of Variance (ANOVA)

Source	df	Sum of Squares	Mean SS	F-stat	p-value
Treatment	3	323.75	107.92	370.00	0.00
Blocks	4	36.50	9.13	31.29	0.00
Error	12	3.50	0.29		
Total	19	363.75			

We reject the null hypothesis using 90% confidence and conclude at least one population mean differs from the others.

b. At the **α = 0.1** level of significance, use Fisher's LSD procedure to test for the equality of the means. Please fill in the missing numbers in the table below.

Pair	Diff	t-critical	LSD	Decision
M1-M2	8.00	1.782	0.609	Reject H0
M1-M3	2.00	1.782	0.609	Reject H0
M1-M4	3.00	1.782	0.609	Reject H0
M2-M3	6.00	1.782	0.609	Reject H0
M2-M4	11.00	1.782	0.609	Reject H0
M3-M4	5.00	1.782	0.609	Reject H0

Using a 90% confidence level, we conclude that all pairs differ from each other except M1-M3.

c. What is the Type I error for the experiment?

Exp-wise Type 1 error: = $(1 - (1 - 0.01)^6) = 0.469$

Problem 3

A coffee shop chain wants to study how sales performance depends on both drink prices and store ambiance. The company believes that these two factors may jointly influence customer behavior.

The experiment includes the following factors:

Factor A (Price Level):
- A1: Low Price
- A2: High Price

Factor B (Store Ambiance):
- B1: Basic Interior
- B2: Modern Interior

The response variable is daily revenue in hundreds of dollars, where higher values indicate better performance. Each treatment combination is tested twice, providing replication for estimating experimental error. Table 2.17 shows the data. The grand mean equals 49.50.

Table: Data for Coffee Prices and Shop Environment

Environment	Low Price	High Price	Average
Basic	42.00	38.00	40.25
Basic	45.00	36.00	
Modern	55.00	60.00	58.75
Modern	58.00	62.00	
Average	50.00	49.00	

(a) Please construct an ANOVA with an interaction term between price and environment.

The interaction term is calculated in the table below:

Table: The Interaction Term for Coffee Prices and Shop Environment

Environment	Low Price	High Price	
Basic	15.1250	15.125	
Modern	15.1250	15.125	
		SSAB	60.500

Table: ANOVA for Coffee Prices and Shop Environment

ANOVA					
Source	df	SS	MS	F	p-value
SSA (Environment)	1	684.5	684.5	210.6	0.0
SSB (Price)	1	2.0	2.0	0.6	0.5
SSAB (Interaction)	1	60.5	60.5	18.6	0.0
SSE	4	13.0	3.3		
SST	7	760.0			

The price is not statistically significant. However, the coffee shop environment and the interaction between coffee price and environment are statistically significant.

Chapter 3 Answers

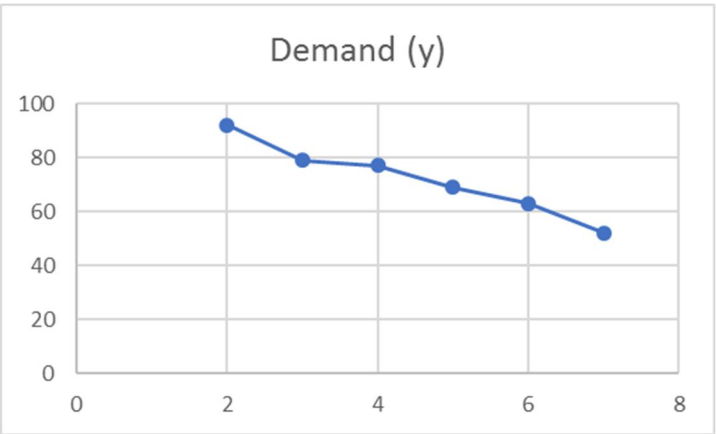
Problem 1.

We want to estimate the relationship between coffee demand and its price. We have the following six observations. We also include the descriptive statistics.

Estimate the Coffee Demand

Observation	Price (x)	Demand (y)	Predicted y	Residuals	Resid²
1	2	92	90.3	1.7	2.9
2	3	79	83.0	-4.0	15.8
3	4	77	75.7	1.3	1.8
4	5	69	68.3	0.7	0.4
5	6	63	61.0	2.0	3.9
6	7	52	53.7	-1.7	2.9
Average	4.5	72		SSE	27.8
Variance	3.5	192.8			
Covariance	-25.6				

a. Plot the data. Please comment on its shape.



We have a negative relationship between quantity demanded for coffee and its price. It reflects the Law of Demand.

b. Please estimate the slope and intercept. Also, estimate the standard errors, t-statistics, and p-values. Put them in the table below. Perform hypothesis testing for the slope and intercept.

Table: Regression Estimates

Coefficient	Parameter Estimate	Std Error	t-stat	p-value
intercept	104.9	3.0	34.6	0.0
slope	-7.3	0.6	-11.6	0.0

- The p-value for the slope lies below an $\alpha = 0.05$. Thus, the population slope does not equal zero.
- The p-value for the intercept lies below an $\alpha = 0.05$. Thus, the population slope does not equal zero.

c. Calculate the ANOVA for coffee demand. Please evaluate the F statistic. Please place the residuals in the coffee data table.

Table: The ANOVA for Coffee Demand

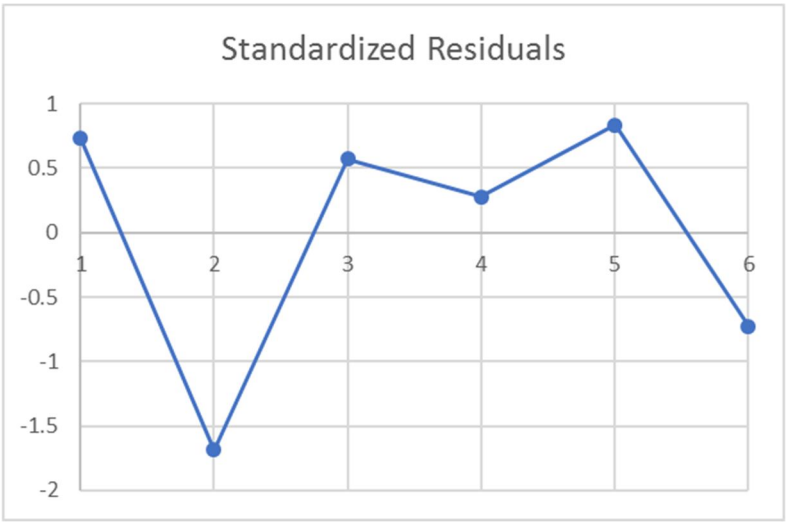
	ANOVA				
	df	SS	MS	F	p-value
SSR	1	936.2	936.2	134.8	0.0
SSE	4	27.8	6.9		
SST	5	964.0			

The p-value lies below an $\alpha = 0.05$. Thus, at least one slope does not equal zero. Of course, we only have one slope.

d. Please calculate the R^2 and Pearson correlation. Please comment on the magnitudes.

The R^2 is 0.97 and the Pearson correlation is -0.99. Thus, the coffee price explains most of the variation in coffee demand.

e. Please plot the standardized residuals. Does it appear that any observation is an outlie?



Standardized residuals lie between -2 and 2. It appears we do not have any outliers.

Problem 2.

A business collected data over 12 consecutive quarters. Revenue (y) and advertising expenditures (x) are both measured in thousands of dollars.

Sum(x_i)	337.80	Cov(x_i,y_i)	118.43
Sum(y_i)	1,092.00	Var(x_i)	43.83
N	12	Var(y_i)	418.91
SSE	1,088.29		
Σ (e_t)	0.00		

(a) Please calculate the ANOVA for this example.

ANOVA

	df	SS	MS	F	p-value
SSR	1	3,519.72	3,519.72	32.34	0.0
SSE	10	1,088.29	108.83		
SST	11	4,608.01			

(b) Please calculate the R^2.

$$R^2 = \frac{SST - SSE}{SST} = \frac{(\ 4{,}608.01\ -\ 1{,}088.29\ )}{4{,}608.01} = 0.764$$

(c) Please comment on the R^2 statistic.

The regression explains 76.4% of the variation in revenue. This indicates that advertising expenditures are an important predictor of revenue. The model provides a reasonably strong fit to the data.

(d) Please calculate the Pearson correlation coefficient. Please comment on it.

$$r_{xy} = (\text{sign of } b_1)\sqrt{R^2} = +\sqrt{0.764} = 0.874$$

This indicates a strong positive linear relationship between advertising and revenue.

(e) Please calculate the least squares estimate b_1.

$$b_1 = \frac{cov(x_i, y_i)}{var(x_i)} = \frac{118.43}{43.83} = 2.70$$

(f) Please calculate the least squares estimate b_0.

$$b_0 = \bar{y} - b_1\bar{x} = \frac{1{,}092.00}{12} - 2.70\left(\frac{337.80}{12}\right) = 14.995$$

(g) Please calculate the standard errors, t statistics, and p-values for the intercept and slope. Perform hypothesis testing.

Coefficient	Parameter Estimate	Std Error	t-stat	p-value
intercept	14.995	13.71	1.09	0.30
slope	2.70	0.48	5.69	0.0

- Slope: Statistically significant at the 5% level ($p < 0.05$)
- Intercept: Not statistically significant ($p > 0.05$)

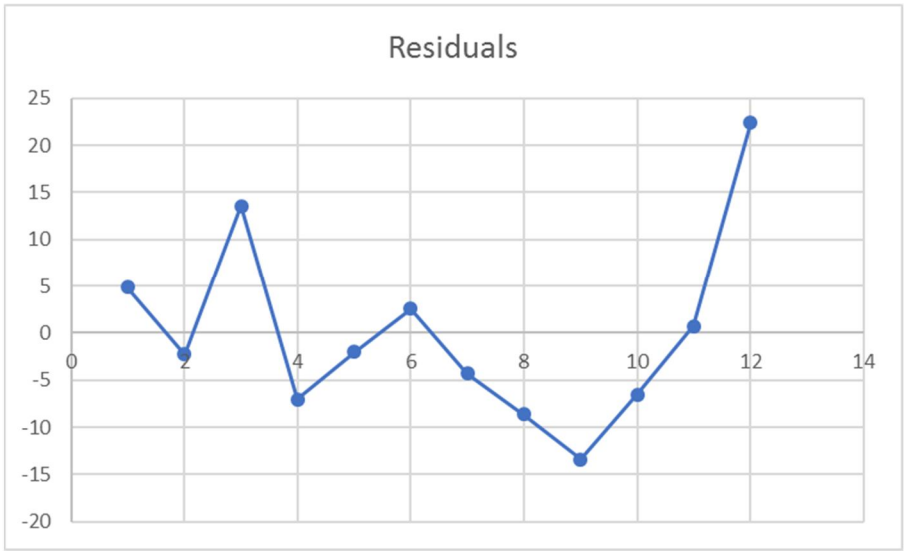
Thus, advertising expenditures have a statistically significant effect on revenue.

(h) Please calculate the marginal effect of b_1 or $\Delta y/\Delta x \left(or\ dy/dx\right)$ and explain what it means.

$$\frac{dy}{dx} = 2.70$$

Since both variables are measured in thousands of dollars, a $1,000 increase in advertising expenditure is associated with an approximate $2,700 increase in revenue, on average.

(i) Please plot the residuals and comment on the plot.



The residual plot suggests that the residuals are not randomly scattered around zero. Instead, there appears to be a pattern in the residuals.

This is important because one of the key assumptions of regression is that residuals behave like random noise. When residuals display a systematic pattern, it indicates that at least one model assumption has been violated.

Chapter 4 Answers

Problem 1.

We revisit the end-of-chapter problem from Chapter 1. An investor is deciding whether to diversify their portfolio by investing in two sectors: technology and energy. The annual returns (%) for two representative stocks over five years are given in the Python code below:

```
# Libraries
import pandas as pd
```

```
# Create the dataset using pandas
data = {
    "Year": [1, 2, 3, 4, 5],
    "Tech": [10, 18, -6, 20, 9],
    "Energy": [4, 12, -3, 8, 7]
}

df = pd.DataFrame(data)

# Display the dataset
print("Dataset:")
print(df)
print()


# Descriptive statistics
print("Descriptive Statistics:")
print(df[["Tech", "Energy"]].agg(["mean", "min", "max", "std","skew","kurt"]))
print()

# Correlation
correlation = df["Tech"].corr(df["Energy"])

print("Correlation between Tech and Energy:")
print(correlation)
```

a. Please compute the following for each stock: minimum, mean, maximum, standard deviation, skewness, and kurtosis.

The output is below:

```
Dataset:
   Year  Tech  Energy
0     1    10       4
1     2    18      12
2     3    -6      -3
3     4    20       8
4     5     9       7

Descriptive Statistics:
           Tech     Energy
mean  10.200000   5.600000
min   -6.000000  -3.000000
max   20.000000  12.000000
std   10.256705   5.594640
skew  -1.095731  -0.860020
kurt   1.239374   1.175576

Correlation between Tech and Energy:
0.9210112879540522
```

b. Compute the correlation coefficient between the two stocks.

The correlation is in the output in a. It equals 0.921.

c. Based on your results:

- Which stock offers higher average returns?
- Which stock is riskier?
- Does diversification appear beneficial?

The Tech stock offers higher average returns but is also more risky, as indicated by its higher standard deviation. The Tech and Energy stocks are highly positively correlated. Therefore, diversification would provide limited benefits, since the returns tend to move together.

Problem 2.

A national survey reports that adults spend an average of 6.5 hours per day on screens. A researcher suspects that remote workers spend more time on screens than this average. A sample of 36 remote workers shows an average screen time of 7.1 hours per day. Assume the population standard deviation is 1.8 hours.

a. Compute the test statistic.

b. Compute the p-value.

c. At $\alpha = 0.05$, state your conclusion.

The Python code is below:

```
# Import the library
import math
from scipy.stats import norm

# Given data
x_bar = 7.1      # sample mean
mu = 6.5         # population mean
sigma = 1.8      # population standard deviation
n = 36           # sample size
alpha = 0.05     # significance level

# Step 1: Compute the test statistic (z)
z = (x_bar - mu) / (sigma / math.sqrt(n))

print("Z-statistic:", round(z, 2))

# Step 2: Compute the p-value (right-tailed test)
```

```
p_value = 1 - norm.cdf(z)

print("P-value:", round(p_value, 3))

# Step 3: Conclusion
if p_value < alpha:
    print("Reject the null hypothesis.")
else:
    print("Fail to reject the null hypothesis.")
```

The output is below:

```
Z-statistic: 2.0
P-value: 0.023
Reject the null hypothesis.
```

Problem 3.

A technology company wants to compare the average battery life of two smartphone models.

Model A uses a new battery design
Model B uses the current standard battery

The company wants to know whether the mean battery life for the new design lasts longer than the standard battery. Table contains the data.

Model A (New Battery)	Model B (Standard Battery)
$n_1 = 36$	$n_2 = 49$
$\bar{x}_1 = 11.8$ hours	$\bar{x}_2 = 10.9$ hours
$\sigma_1 = 2.4$ hours	$\sigma_2 = 2.8$ hours

a. Calculate the pooled standard error.

b. Compute the test statistic

c. Compute the p-value

d. What is our conclusion using $\alpha = 0.05$?

The Python program is below:

```
# Import libraries
import math
```

```
from scipy.stats import norm

# Given data
x1_bar = 11.8   # sample mean (Model A - new battery)
x2_bar = 10.9   # sample mean (Model B - standard battery)

sigma1 = 2.4    # population standard deviation (Model A)
sigma2 = 2.8    # population standard deviation (Model B)

n1 = 36         # sample size (Model A)
n2 = 49         # sample size (Model B)

alpha = 0.05

# Step 1: Compute the standard error
std_error = math.sqrt((sigma1**2)/n1 + (sigma2**2)/n2)

print("Standard Error:", round(std_error, 3))

# Step 2: Compute the test statistic (z)
z = (x1_bar - x2_bar) / std_error

print("Z-statistic:", round(z, 2))

# Step 3: Compute the p-value (RIGHT-tailed test)
p_value = 1 - norm.cdf(z)

print("P-value:", round(p_value, 4))

# Step 4: Conclusion
if p_value < alpha:
    print("Reject the null hypothesis.")
else:
    print("Fail to reject the null hypothesis.")
```

The Python output is below:

```
Standard Error: 0.566
Z-statistic: 1.59
P-value: 0.0558
Fail to reject the null hypothesis.
```

Since the p-value exceeds the α, we fail to reject the null hypothesis. We do not have sufficient evidence to conclude that the two models have different battery lives.

Problem 4.

A university wants to compare the average exam scores of two teaching methods.

Method A uses traditional lectures
Method B uses interactive learning

The university wants to determine whether the mean scores differ between the two methods.

Method A (Traditional)	Method B (Interactive)
$n_1 = 25$	$n_2 = 30$
$\bar{x}_1 = 78.6$	$\bar{x}_2 = 74.2$
$s_1 = 10.5$	$s_2 = 9.8$

a. State the hypotheses

b. Compute the test statistic

c. Compute the p-value

d. What is our conclusion using $\alpha = 0.05$?

The Python program is below:

```
# ------------------------------------------------
# Two-Sample t-Test (Welch's Test)
# Comparing Teaching Methods
# ------------------------------------------------

# Step 1: Import libraries
import numpy as np
from scipy import stats

# ------------------------------------------------
# Step 2: Input sample statistics
# ------------------------------------------------

# Method A (Traditional)
n1 = 25
x1_bar = 78.6
s1 = 10.5

# Method B (Interactive)
n2 = 30
x2_bar = 74.2
s2 = 9.8

# ------------------------------------------------
# Step 3: State the hypotheses
# ------------------------------------------------
```

```
print("H0:  1 =  2")
print("Ha:  1 ≠  2")

# ------------------------------------------------
# Step 4: Compute test statistic
# ------------------------------------------------

# Standard error
se = np.sqrt((s1**2 / n1) + (s2**2 / n2))

# t-statistic
t_stat = (x1_bar - x2_bar) / se

print("\nt-statistic:", t_stat)

# ------------------------------------------------
# Step 5: Degrees of freedom (Welch approximation)
# ------------------------------------------------

df = ((s1**2 / n1 + s2**2 / n2)**2) / \
     (((s1**2 / n1)**2 / (n1 - 1)) + ((s2**2 / n2)**2 / (n2 - 1)))

print("Degrees of freedom:", df)

# ------------------------------------------------
# Step 6: Compute p-value (two-tailed test)
# ------------------------------------------------

p_value = 2 * (1 - stats.t.cdf(abs(t_stat), df))

print("p-value:", p_value)

# ------------------------------------------------
# Step 7: Decision
# ------------------------------------------------

alpha = 0.05

if p_value < alpha:
    print("Reject H0: The mean scores differ between the two methods.")
else:
    print("Fail to reject H0: No significant difference in mean scores.")
```

The Python output is below:

```
H0:  1 =  2
Ha:  1 ≠  2

t-statistic: 1.5948590439478274
Degrees of freedom: 49.78142106740131
p-value: 0.11707366659053142
Fail to reject H0: No significant difference in mean scores.
```

Problem 5.

A retail company is working to improve the consistency of its customer experience. Management believes that reducing variability in customer satisfaction is just as important as increasing the average score.

Customer satisfaction is measured on a 100-point scale. Based on past data, the company expects a population standard deviation of $\sigma = 10$.

After implementing a new employee training program, the company collects a sample of 15 customer satisfaction scores. The Python code to solve this problem is below:

```
# Libraries
import pandas as pd
from scipy.stats import chi2

# Create the dataset
data = {
    "Score": [88, 84, 91, 79, 85,
              87, 82, 90, 86, 83,
              89, 81, 92, 78, 84]
}

df = pd.DataFrame(data)

# Descriptive statistics
print("Descriptive Statistics:")
print(df["Score"].agg(["mean", "var", "std"]))
print()

# Extract values for hypothesis test
alpha = 0.05
n = len(df)
s2 = df["Score"].var()
sigma2 = 100

# Right-tail critical value
critical_value = chi2.ppf(1 - alpha, n-1)

print("Critical chi-square value:")
print(critical_value)
print(" ")

# Chi-square test statistic
chi_square = (n - 1) * s2 / sigma2

print("Chi-square test statistic:")
print(chi_square)
```

a. What is the sample mean customer satisfaction score?

b. What is the sample variance?

c. What is the sample standard deviation?

d. Conduct a hypothesis test to determine whether the variability has changed after the training program. Use a significance level of 0.05.
- State the null and alternative hypotheses
- Compute the test statistic
- State the decision rule
- Provide your conclusion

The output is below:

```
Descriptive Statistics:
mean    85.266667
var     18.209524
std      4.267262
Name: Score, dtype: float64

Critical chi-square value:
23.684791304840576

Chi-square test statistic:
2.5493333333333332
```

We fail to reject the null hypothesis at the 5% significance level. There is insufficient evidence to conclude that the population variance is greater than 100.

Problem 6.

A manufacturing company operates two production lines that produce the same product. Management wants to ensure that both lines maintain consistent quality. In particular, they are concerned about variability in product thickness, since high variability can lead to defects.

To evaluate consistency, the company collects samples from each production line:

Production Line 1: $n_1 = 21$ $s_1^2 = 8.2$
Production Line 2: $n_2 = 26$ $s_2^2 = 3.0$

Management wants to test whether the variability of the two production lines is the same. Please use Python to solve this problem. The hypotheses are below:

H_o: $\sigma_1^2 = \sigma_2^2$
H_a: $\sigma_1^2 \neq \sigma_2^2$

a. Use the p-value approach to test the hypotheses at the 5% significance level.

Hint: Place the larger sample variance in the numerator and conduct a right-tailed test.

b. Repeat the test using the critical value approach.

The Python code is below:

```
# Load the library
from scipy.stats import f

# Data
n1 = 21
s1_sq = 8.2

n2 = 26
s2_sq = 3.0

alpha = 0.05

# Step 1: Put larger variance in numerator
F = s1_sq / s2_sq

print("F-statistic:")
print(F)
print()

# Step 2: Degrees of freedom
df1 = n1 - 1
df2 = n2 - 1

print("Degrees of freedom:")
print("df1 = ", df1)
print("df2 = ", df2)
print()

# Right-tail probability
p_value_right = 1 - f.cdf(F, df1, df2)

# Two-tailed p-value
p_value = 2 * p_value_right

print("p-value:")
print(p_value)

# Two-tailed test use alpha/2 in right tail
critical_value = f.ppf(1 - alpha/2, df1, df2)
```

```
print("Critical F-value:")
print(critical_value)
print()
```

The output is below:

```
F-statistic:
2.733333333333333

Degrees of freedom:
df1 =  20
df2 =  25

p-value:
0.018530027285460937
Critical F-value:
2.3004547884921696
```

The critical F-value is 2.30. Since the calculated F-statistic exceeds the critical value, we reject the null hypothesis. Each sample "loses" one degree of freedom because we estimate the mean before computing the variance.

Chapter 5 Answers

Problem 1

A researcher wants to examine whether different study environments affect student performance. The experiment includes one factor:

Factor (Study Environment):

- A1 : Quiet Room
- A2 : Coffee Shop
- A3 : Library

The response variable is test score. Each environment is tested with five students.

Please calculate the ANOVA and determine whether the study place affects a student's test scores.

The Python code is below.

```
# Import the required library
from scipy import stats
```

```
# ------------------------------
# Step 1: Enter the data
# ------------------------------

# Test scores for each study environment
quiet_room = [78, 82, 80, 79, 81]
coffee_shop = [81, 79, 83, 80, 78]
library = [80, 81, 79, 82, 78]

# ------------------------------
# Step 2: Perform One-Way ANOVA
# ------------------------------

# f_oneway() compares the means of two or more groups
# It returns:
#   F-statistic → ratio of between-group variance to within-group variance
#   p-value     → probability of observing results if means are equal

F_stat, p_value = stats.f_oneway(quiet_room, coffee_shop, library)

# ------------------------------
# Step 3: Print results
# ------------------------------

print("ANOVA Results")
print("F-statistic:", F_stat)
print("p-value:", p_value)

# ------------------------------
# Step 4: Decision Rule
# ------------------------------

alpha = 0.05  # significance level

print("\nDecision at alpha =", alpha)

if p_value < alpha:
    print("Reject the null hypothesis.")
    print("Conclusion: Study environment affects test scores.")
else:
    print("Fail to reject the null hypothesis.")
    print("Conclusion: No significant difference in test scores across environments.")
```

The Python output is below:

```
ANOVA Results
F-statistic: 0.022988505747126433
p-value: 0.9773166464994383

Decision at alpha = 0.05
Fail to reject the null hypothesis.
Conclusion: No significant difference in test scores across environments.
```

Problem 2

Four machines are tested. Five operators serve as blocks.

a. Construct an analysis of variance table. What is your conclusion using $\alpha = 0.1$?

b. At the $\alpha = 0.1$ level of significance, use Fisher's LSD procedure to test for the equality of the means. Please fill in the missing numbers.

c. What is the Type I error for the experiment?

The Python program is below:

```
# Import libraries
import pandas as pd
import statsmodels.api as sm
from statsmodels.formula.api import ols
from itertools import combinations
from scipy import stats
import numpy as np

# ------------------------------
# Step 1: Enter the data
# ------------------------------

# Create dataset in long format (required for statsmodels)
data = pd.DataFrame({
    'Operator': [1,1,1,1, 2,2,2,2, 3,3,3,3, 4,4,4,4, 5,5,5,5],
    'Machine': ['M1','M2','M3','M4'] * 5,
    'Score': [
        72,80,74,69,
        75,82,76,70,
        73,81,75,71,
        74,83,77,72,
        76,84,78,73
    ]
})

# Convert to categorical variables
data['Operator'] = data['Operator'].astype('category')
data['Machine'] = data['Machine'].astype('category')

# ------------------------------
# Step 2: Fit ANOVA model
# ------------------------------

# Randomized Block Design:
# Score = Machine effect + Block (Operator) effect
model = ols('Score ~ C(Machine) + C(Operator)', data=data).fit()
```

```
# Generate ANOVA table
anova_table = sm.stats.anova_lm(model, typ=2)

print("ANOVA Table:")
print(anova_table)

# ------------------------------
# Step 3: Decision (alpha = 0.1)
# ------------------------------

alpha = 0.10

p_value = anova_table.loc['C(Machine)', 'PR(>F)']

print("\nDecision for Machine Effect at alpha =", alpha)

if p_value < alpha:
    print("Reject the null hypothesis.")
    print("Conclusion: Machine means are significantly different.")
else:
    print("Fail to reject the null hypothesis.")
    print("Conclusion: No significant difference among machines.")

# ------------------------------
# Step 4: Fisher's LSD Test
# ------------------------------

# Get Mean Square Error (MSE) from ANOVA table
MSE = anova_table.loc['Residual', 'sum_sq'] / anova_table.loc['Residual', 'df']

# Number of observations per treatment (balanced design)
n = 5

# Critical t-value (two-tailed)
df_error = anova_table.loc['Residual', 'df']
t_crit = stats.t.ppf(1 - alpha/2, df_error)

# Calculate LSD
LSD = t_crit * np.sqrt(2 * MSE / n)

print("\nFisher's LSD value:", LSD)

# Calculate group means
means = data.groupby('Machine', observed=True)['Score'].mean()

print("\nMachine Means:")
print(means)

# Pairwise comparisons
print("\nPairwise Comparisons (Fisher's LSD):")

for m1, m2 in combinations(means.index, 2):
    diff = abs(means[m1] - means[m2])
```

```
    significant = "Significant" if diff > LSD else "Not Significant"

    print(f"{m1} vs {m2}: |Difference| = {diff:.3f} → {significant}")

# ------------------------------
# Step 5: Experiment-wise Type I Error
# ------------------------------

c = 6  # number of pairwise comparisons

alpha_experiment = 1 - (1 - alpha)**c

print()
print("Experimentwise Type I Error:", alpha_experiment)
print()
```

The Python output is below:

```
ANOVA Table:
             sum_sq    df           F        PR(>F)
C(Machine)   323.75   3.0  370.000000  4.369038e-12
C(Operator)   36.50   4.0   31.285714  2.905950e-06
Residual       3.50  12.0         NaN           NaN

Decision for Machine Effect at alpha = 0.1
Reject the null hypothesis.
Conclusion: Machine means are significantly different.

Fisher's LSD value: 0.6087670944506464

Machine Means:
Machine
M1    74.0
M2    82.0
M3    76.0
M4    71.0
Name: Score, dtype: float64

Pairwise Comparisons (Fisher's LSD):
M1 vs M2: |Difference| = 8.000 → Significant
M1 vs M3: |Difference| = 2.000 → Significant
M1 vs M4: |Difference| = 3.000 → Significant
M2 vs M3: |Difference| = 6.000 → Significant
M2 vs M4: |Difference| = 11.000 → Significant
M3 vs M4: |Difference| = 5.000 → Significant

Experimentwise Type I Error: 0.46855899999999995
```

Problem 3

A coffee shop chain wants to study how sales performance depends on both drink prices and store ambiance. The company believes that these two factors may jointly influence customer behavior.

The experiment includes the following factors:

Factor A (Price Level):
- A1: Low Price
- A2: High Price

Factor B (Store Ambiance):
- B1: Basic Interior
- B2: Modern Interior

The response variable is daily revenue in hundreds of dollars, where higher values indicate better performance. Each treatment combination is tested twice, providing replication for estimating experimental error.

(a) Please construct an ANOVA with an interaction term between price and environment.

The Python code is below:

```
# Import libraries
import pandas as pd
import statsmodels.api as sm
from statsmodels.formula.api import ols

# ------------------------------
# Step 1: Enter the data
# ------------------------------

# We enter the data in "long format"
# Each row is one observation

data = pd.DataFrame({
    'Price': [
        'Low', 'Low', 'High', 'High',
        'Low', 'Low', 'High', 'High'
    ],
    'Environment': [
        'Basic', 'Basic', 'Basic', 'Basic',
        'Modern', 'Modern', 'Modern', 'Modern'
    ],
    'Revenue': [
        42, 45, 38, 36,   # Basic
```

```
        55, 58, 60, 62    # Modern
    ]
})

# Convert to categorical variables
data['Price'] = data['Price'].astype('category')
data['Environment'] = data['Environment'].astype('category')

# ------------------------------
# Step 2: Fit the ANOVA model
# ------------------------------

# Model with interaction term:
# Revenue = Price + Environment + Price*Environment

model = ols('Revenue ~ C(Price) * C(Environment)', data=data).fit()

# ------------------------------
# Step 3: Generate ANOVA table
# ------------------------------

anova_table = sm.stats.anova_lm(model, typ=2)

print("Two-Way ANOVA Table (with Interaction):")
print(anova_table)

# ------------------------------
# Step 4: Interpret results
# ------------------------------

alpha = 0.05

print("\nDecision at alpha =", alpha)

for effect in ['C(Price)', 'C(Environment)', 'C(Price):C(Environment)']:
    p_value = anova_table.loc[effect, 'PR(>F)']

    if p_value < alpha:
        print(f"{effect}: Significant effect (Reject H0)")
    else:
        print(f"{effect}: Not significant (Fail to reject H0)")
```

The Python output is below:

```
Two-Way ANOVA Table (with Interaction):
                        sum_sq   df           F    PR(>F)
C(Price)                   2.0  1.0    0.615385  0.476621
C(Environment)           684.5  1.0  210.615385  0.000131
C(Price):C(Environment)   60.5  1.0   18.615385  0.012501
Residual                  13.0  4.0         NaN       NaN

Decision at alpha = 0.05
C(Price): Not significant (Fail to reject H0)
```

```
C(Environment): Significant effect (Reject H0)
C(Price):C(Environment): Significant effect (Reject H0)
```

Problem 4.

We want to estimate the relationship between coffee demand and its price. We have the following six observations.

a. Plot the data.

b. Please estimate the simple linear regression between coffee demand and coffee price.

c. Calculate the ANOVA for coffee demand. Please evaluate the F statistic. Then comment on the coffee's intercept and slope.

d. Please comment on the R^2.

e. Please plot the standardized residuals. Does it appear that any observation is an outlie?

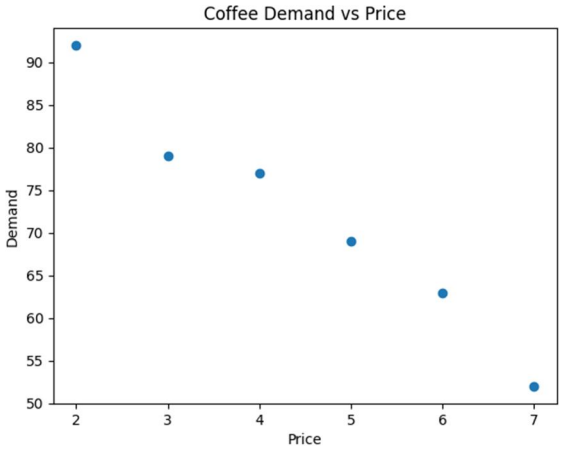
The Python code is below:

```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.formula.api import ols

# ------------------------------
# Step 1: Enter the data
# ------------------------------

data = pd.DataFrame({
    'Price': [2, 3, 4, 5, 6, 7],
    'Demand': [92, 79, 77, 69, 63, 52]
})

# ------------------------------
# (a) Plot the data
# ------------------------------

plt.scatter(data['Price'], data['Demand'])
plt.title("Coffee Demand vs Price")
plt.xlabel("Price")
plt.ylabel("Demand")
plt.show()

# ------------------------------
```

```
# (b) Estimate regression model
# ------------------------------

# Add constant (intercept)
X = sm.add_constant(data['Price'])
y = data['Demand']

model = sm.OLS(y, X).fit()

print()
print(model.summary())
print()

# Clean output (avoids small-sample warning)
print("\nRegression Results:")
print("Intercept:", model.params['const'])
print("Slope:", model.params['Price'])

print("\nStandard Errors:")
print(model.bse)

print("\nR-squared:", model.rsquared)

# ------------------------------
# (c) ANOVA and F-statistic
# ------------------------------

anova_model = ols('Demand ~ Price', data=data).fit()
anova_table = sm.stats.anova_lm(anova_model)

print("\nANOVA Table:")
print(anova_table)

# Future-proof extraction using labels
F_stat = anova_table.loc['Price', 'F']
p_value = anova_table.loc['Price', 'PR(>F)']

print("\nF-statistic:", F_stat)
print("p-value:", p_value)

# ------------------------------
# Interpretation of coefficients
# ------------------------------

print("\nInterpretation:")
print("The slope indicates how demand changes with price.")
print("A negative slope suggests that higher prices reduce demand.")

# ------------------------------
# (d) R-squared
# ------------------------------

print("\nR-squared Interpretation:")
```

```
print("R-squared measures how much variation in demand is explained by price.")
print()

# -------------------------------
# (e) Standardized residuals
# -------------------------------

# Get influence measures
influence = model.get_influence()

# Standardized residuals
standardized_residuals = influence.resid_studentized_internal

# Plot standardized residuals
plt.scatter(data['Price'], standardized_residuals)
plt.axhline(0, color="darkblue")
plt.axhline(2, linestyle='--', color="darkred")
plt.axhline(-2, linestyle='--', color="darkred")

plt.title("Standardized Residuals")
plt.xlabel("Price")
plt.ylabel("Standardized Residuals")
plt.show()

# Print residuals
print("\nStandardized Residuals:")
for i, res in enumerate(standardized_residuals, start=1):
    print(f"Observation {i}: {res:.3f}")

# -------------------------------
# Outlier check
# -------------------------------

print("\nOutlier Check:")
if any(abs(r) > 2 for r in standardized_residuals):
    print("There may be outliers (|residual| > 2).")
else:
    print("No strong evidence of outliers.")
```

The graph of the data is on the next page:

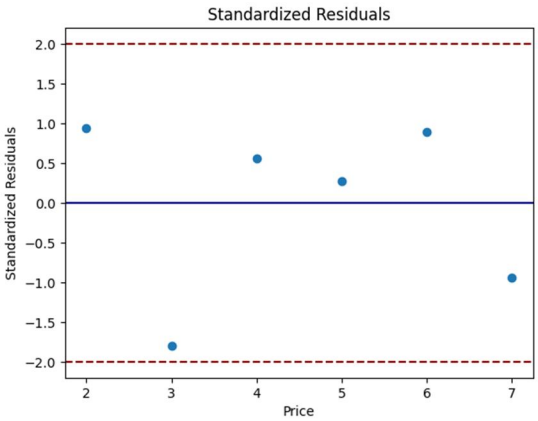


The Python output is below:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 Demand   R-squared:                       0.971
Model:                            OLS   Adj. R-squared:                  0.964
Method:                 Least Squares   F-statistic:                     134.8
Date:                Sat, 11 Apr 2026   Prob (F-statistic):           0.000314
Time:                        01:11:32   Log-Likelihood:                -13.110
No. Observations:                   6   AIC:                             30.22
Df Residuals:                       4   BIC:                             29.80
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        104.9143      3.032     34.606      0.000      96.497     113.332
Price         -7.3143      0.630    -11.612      0.000      -9.063      -5.565
==============================================================================
Omnibus:                        nan   Durbin-Watson:                   2.749
Prob(Omnibus):                  nan   Jarque-Bera (JB):                0.920
Skew:                        -0.875   Prob(JB):                        0.631
Kurtosis:                     2.214   Cond. No.                         14.1
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.


Regression Results:
Intercept: 104.91428571428581
Slope: -7.314285714285729

Standard Errors:
const    3.031670
Price    0.629869
dtype: float64
```

```
R-squared: 0.9711914641375223

ANOVA Table:
           df      sum_sq     mean_sq           F    PR(>F)
Price     1.0  936.228571  936.228571  134.847737  0.000314
Residual  4.0   27.771429    6.942857         NaN       NaN

F-statistic: 134.84773662551487
p-value: 0.00031426242216900824

Interpretation:
The slope indicates how demand changes with price.
A negative slope suggests that higher prices reduce demand.

R-squared Interpretation:
R-squared measures how much variation in demand is explained by price.

/usr/local/lib/python3.12/dist-packages/statsmodels/stats/stattools.py:74:
ValueWarning: omni_normtest is not valid with less than 8 observations; 6 samples were
given.
  warn("omni_normtest is not valid with less than 8 observations; %i "
```

The plot of the standardized residuals is below:



The rest of the Python output is below:

```
Standardized Residuals:
Observation 1: 0.943
Observation 2: -1.795
Observation 3: 0.563
Observation 4: 0.276
Observation 5: 0.891
Observation 6: -0.943

Outlier Check:
No strong evidence of outliers.
```

Note: The regression produced an error because the procedure requires at least 8 observations. Errors are common, especially when code writers update libraries and re-organize and change commands.

Chapter 6 Answers

Problem 1.

The Python program is below:

```
# ------------------------------------------------
# Multiple Regression Example: Housing Prices
# ------------------------------------------------

# Import libraries
import pandas as pd
import statsmodels.api as sm

# Create dataset manually
data = {
    "Size": [50,60,80,100,120,140,160,180,70,90,110,130,150,170,200],
    "Bedrooms": [1,2,3,3,4,4,5,5,2,3,3,4,4,5,6],
    "Age": [20,15,10,5,8,3,2,1,12,9,7,6,4,2,1],
    "Price": [120,150,200,250,280,320,360,400,170,220,260,300,340,380,420]
}

# Convert to DataFrame
df = pd.DataFrame(data)

# Display dataset
print(df)

# Define independent variables
X = df[["Size", "Bedrooms", "Age"]]
X = sm.add_constant(X)

# Define dependent variable
y = df["Price"]

# Estimate regression model
model = sm.OLS(y, X).fit()

# Print results
print(model.summary())
```

The Python output is below:

```
                            OLS Regression Results
==============================================================================
```

```
Dep. Variable:                  Price   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.997
Method:                 Least Squares   F-statistic:                     1779.
Date:                Fri, 10 Apr 2026   Prob (F-statistic):           4.70e-15
Time:                        04:57:41   Log-Likelihood:                -42.401
No. Observations:                  15   AIC:                             92.80
Df Residuals:                      11   BIC:                             95.63
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         98.5352     13.233      7.446      0.000      69.410     127.660
Size           1.6839      0.121     13.913      0.000       1.418       1.950
Bedrooms      -0.3815      4.168     -0.092      0.929      -9.554       8.791
Age           -3.1933      0.618     -5.168      0.000      -4.553      -1.833
==============================================================================
Omnibus:                        4.573   Durbin-Watson:                   2.069
Prob(Omnibus):                  0.102   Jarque-Bera (JB):                2.311
Skew:                          -0.930   Prob(JB):                        0.315
Kurtosis:                       3.488   Cond. No.                     1.39e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.39e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

First, we examine the F-statistic. The p-value is extremely small, Thus, it indicates that we reject the null hypothesis that all slope coefficients are equal to zero. At least one explanatory variable has a statistically significant relationship with housing prices. Overall, the model is statistically significant.

Second, we evaluate the individual t-statistics for each coefficient.

- The intercept, Size, and Age are statistically significant
- The number of Bedrooms is not statistically significant

This means that, after controlling for the other variables:

- Size and Age contribute meaningfully to explaining house prices
- Bedrooms does not add additional explanatory power beyond what is already captured by the other variables

Third, we consider whether the signs and magnitudes of the coefficients make economic sense.

- Size (positive): Larger houses have higher prices, which is consistent with expectations

- Bedrooms (insignificant): This may be due to multicollinearity, since the number of bedrooms is likely highly correlated with size. Once size is included, bedrooms may not provide much additional information
- Age (negative): Older houses tend to have lower prices, which is also consistent with real-world behavior

Overall, the signs of the coefficients are intuitive and consistent with economic reasoning.

Fourth, we examine the R^2 value. The $R^2 = 0.998$, meaning the model explains 99.8% of the variation in housing prices. This indicates an excellent fit. However, students should be cautious: A very high R^2 does not guarantee that the model is correct. It may also reflect a small or highly structured dataset.

Fifth, we consider the Durbin-Watson statistic, which is approximately 2. A value near 2 suggests no autocorrelation. However, this dataset is cross-sectional, not time series. Therefore, the order of observations has no meaningful interpretation, and autocorrelation is generally not a concern in this context.

Problem 2.

Write a Python program to complete the following tasks:

a. Import the dataset Chapter_06-cubic_cost_function.csv into Python and display the first few observations.

The Python program is on the next page:

```
# ------------------------------------------------
# Cubic Cost Function with Noise
# Data Import, Visualization, and Estimation
# ------------------------------------------------

# Step 1: Import libraries
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt

# ------------------------------------------------
# Step 2: Load the dataset
# ------------------------------------------------

# Make sure the CSV file is in your working directory
df = pd.read_csv("Chapter_06-cubic_cost_function.csv")
```

```
# Display first few rows
print("Dataset:")
print(df.head())

# ------------------------------------------------
# Step 3: Plot Total Cost vs Quantity
# ------------------------------------------------

plt.figure()

# Scatter plot
plt.scatter(df["Quantity"], df["Total_Cost"])

# Labels and title
plt.xlabel("Quantity")
plt.ylabel("Total Cost")
plt.title("Total Cost vs Quantity")

plt.show()

# ------------------------------------------------
# Step 4: Create squared and cubic terms
# ------------------------------------------------

df["Quantity_sq"] = df["Quantity"]**2
df["Quantity_cu"] = df["Quantity"]**3

print("\nDataset with Polynomial Terms:")
print(df.head())

# ------------------------------------------------
# Step 5: Correlation Matrix
# ------------------------------------------------

corr_matrix = df[["Quantity", "Quantity_sq", "Quantity_cu", "Total_Cost"]].corr()

print("\nCorrelation Matrix:")
print(corr_matrix)

# ------------------------------------------------
# Step 6: Estimate Cubic Regression Model
# ------------------------------------------------

# Define independent variables
X = df[["Quantity", "Quantity_sq", "Quantity_cu"]]
X = sm.add_constant(X)

# Dependent variable
y = df["Total_Cost"]

# Estimate model
model = sm.OLS(y, X).fit()
```

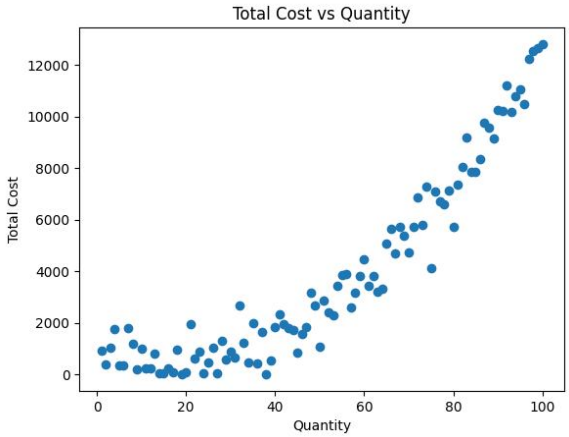
```
# Print regression results
print("\nRegression Results:")
print(model.summary())
```

The Python output is below:

```
Dataset:
   Quantity   Total_Cost
0         1   902.581322
1         2   400.268559
2         3  1035.220830
3         4  1742.263885
4         5   343.927300

Dataset with Polynomial Terms:
   Quantity   Total_Cost  Quantity_sq  Quantity_cu
0         1   902.581322            1            1
1         2   400.268559            4            8
2         3  1035.220830            9           27
3         4  1742.263885           16           64
4         5   343.927300           25          125

Correlation Matrix:
             Quantity  Quantity_sq  Quantity_cu  Total_Cost
Quantity     1.000000     0.968854     0.917552    0.914236
Quantity_sq  0.968854     1.000000     0.986087    0.975385
Quantity_cu  0.917552     0.986087     1.000000    0.981617
Total_Cost   0.914236     0.975385     0.981617    1.000000

Regression Results:
                            OLS Regression Results
==============================================================================
Dep. Variable:             Total_Cost   R-squared:                       0.967
Model:                            OLS   Adj. R-squared:                  0.966
Method:                 Least Squares   F-statistic:                     937.0
Date:                Fri, 10 Apr 2026   Prob (F-statistic):           6.22e-71
Time:                        05:31:03   Log-Likelihood:                -792.57
No. Observations:                 100   AIC:                             1593.
Df Residuals:                      96   BIC:                             1604.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        973.2668    283.975      3.427      0.001     409.580    1536.954
Quantity     -48.9945     24.229     -2.022      0.046     -97.088      -0.901
Quantity_sq    1.4188      0.556      2.552      0.012       0.315       2.522
Quantity_cu    0.0024      0.004      0.677      0.500      -0.005       0.010
==============================================================================
Omnibus:                        1.836   Durbin-Watson:                   2.217
Prob(Omnibus):                  0.399   Jarque-Bera (JB):                1.270
Skew:                          -0.171   Prob(JB):                        0.530
Kurtosis:                       3.434   Cond. No.                     1.60e+06
==============================================================================

Notes:
```

```
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.6e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

b. Create a scatter plot showing the relationship between total cost (TC) and quantity produced (q). Briefly describe the pattern you observe.



The cost function is relatively flat at low levels of output. Beyond approximately 40 units, the function slopes upward.

c. Generate two additional variables:

- q^2 (quantity squared)
- q^3 (quantity cubed)

Then compute and display the correlation matrix for q, q^2, q^3, and TC.

Quantity is highly correlated with both q^2 and q^3, with correlations exceeding 0.7. This indicates multicollinearity. This is common in polynomial regression models because the higher-order terms are functions of the original variable.

Multicollinearity is expected in polynomial models and does not necessarily invalidate the model. However, it can make individual coefficient estimates less stable and harder to interpret.

d. Estimate the following cubic cost function using ordinary least squares (OLS):

$$TC = \beta_0 + \beta_1 \cdot q + \beta_2 \cdot q^2 + \beta_3 \cdot q^3 + \varepsilon$$

e. Interpret the regression results. In your answer, comment on:

- The statistical significance of the coefficients
- The overall fit of the model
- Whether the estimated relationship is consistent with a typical total cost function

First, we examine the F-statistic. The F-statistic is statistically significant. It indicates that we reject the null hypothesis that all slope coefficients are equal to zero. At least one explanatory variable has a significant relationship with total cost, so the overall model is statistically significant.

Second, we evaluate the individual coefficients using t-statistics. The intercept (constant), quantity, and quantity squared are statistically significant. The quantity cubed term is not statistically significant. This suggests that the linear and quadratic terms contribute meaningfully to the model, while the cubic term does not appear to add additional explanatory power based on the t-test alone.

Third, we examine the goodness of fit. The $R^2 = 0.967$, meaning the model explains 96.7% of the variation in total cost. The model fits the data very well and captures most of the variation in total cost.

Fourth, we consider the insignificance of the cubic term. Although the coefficient on q^3 is not statistically significant, this does not necessarily mean the variable should be removed. The explanatory variables q, q^2, and q^3 are highly correlated with each other, leading to multicollinearity. Multicollinearity can inflate standard errors and make it difficult to detect statistical significance, even when a variable is theoretically important.

Overall, the estimated relationship is consistent with a typical total cost function: Costs increase as output increases. The presence of curvature (captured by q^2 and q^3) reflects increasing marginal costs at higher levels of production. Even though the cubic term is not statistically significant, it may still be appropriate to retain it in the model because it helps capture the nonlinear structure implied by economic theory.

Key Insight

In polynomial regression, multicollinearity is expected, and statistical insignificance does not always imply that a variable should be excluded—especially when theory suggests it belongs in the model.

Chapter 7 Answers

Problem 1.

The Python code is below:

```
# ==========================================
# MANOVA Example: Iris Dataset
# ==========================================

# ------------------------------------------
# Step 1: Import required libraries
# ------------------------------------------
import pandas as pd
from statsmodels.multivariate.manova import MANOVA
from sklearn.datasets import load_iris

# ------------------------------------------
# Step 2: Load the dataset
# ------------------------------------------
# Load iris dataset from sklearn
data = load_iris()

# Convert to pandas DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)

# Add species (target variable)
df['species'] = data.target

# Preview the data
print("First five observations:")
print(df.head())

# ------------------------------------------
# Step 3: Clean and rename variables
# ------------------------------------------
# Rename columns for easier use in formulas
df.columns = [
    'sepal_length',
    'sepal_width',
    'petal_length',
    'petal_width',
    'species'
]

# ------------------------------------------
# Step 4: Convert numeric labels to names
# ------------------------------------------
# Replace numeric species codes with labels
df['species'] = df['species'].map({
    0: 'setosa',
    1: 'versicolor',
```

```
    2: 'virginica'
})

# Verify changes
print("\nUnique species categories:")
print(df['species'].unique())

# ------------------------------------------
# Step 5: Perform MANOVA
# ------------------------------------------
# Define MANOVA model:
# Dependent variables: four flower measurements
# Independent variable: species
model = MANOVA.from_formula(
    'sepal_length + sepal_width + petal_length + petal_width ~ species',
    data=df
)

# Fit model and display results
print("\nMANOVA Results:")
print(model.mv_test())
```

The output is below:

```
Unique species categories:
['setosa' 'versicolor' 'virginica']

MANOVA Results:
                  Multivariate linear model
================================================================

----------------------------------------------------------------
       Intercept         Value  Num DF  Den DF   F Value  Pr > F
----------------------------------------------------------------
          Wilks' lambda  0.0170 4.0000 144.0000 2086.7720 0.0000
         Pillai's trace  0.9830 4.0000 144.0000 2086.7720 0.0000
 Hotelling-Lawley trace 57.9659 4.0000 144.0000 2086.7720 0.0000
    Roy's greatest root 57.9659 4.0000 144.0000 2086.7720 0.0000
----------------------------------------------------------------

----------------------------------------------------------------
        species          Value  Num DF  Den DF   F Value  Pr > F
----------------------------------------------------------------
          Wilks' lambda  0.0234 8.0000 288.0000  199.1453 0.0000
         Pillai's trace  1.1919 8.0000 290.0000   53.4665 0.0000
 Hotelling-Lawley trace 32.4773 8.0000 203.4024  582.1970 0.0000
    Roy's greatest root 32.1919 4.0000 145.0000 1166.9574 0.0000
================================================================
```

All multivariate test statistics, e.g., Pillai's Trace, Wilks' Lambda, Hotelling's Trace, and Roy's Largest Root, are statistically significant at the 5% level. This provides strong evidence to reject the null hypothesis that the mean vectors of the dependent variables are equal across species.

In practical terms, this means that the three species of iris flowers differ significantly when considering sepal length, sepal width, petal length, and petal width simultaneously. The differences are not limited to a single characteristic but reflect a combined pattern across all four measurements.

Because MANOVA accounts for the relationships among these variables, the results indicate that the overall profile of flower characteristics varies by species, rather than differences occurring in isolation for each variable.

Problem 2.

The Python code for the teen gambling problem is below:

```
# ============================================
# ANCOVA Example: Teen Gambling Dataset
# ============================================

# --------------------------------------------
# Step 1: Import required libraries
# --------------------------------------------
import pandas as pd
import statsmodels.api as sm
from statsmodels.formula.api import ols

# --------------------------------------------
# Step 2: Load the dataset
# --------------------------------------------
teengamb = pd.read_csv('Chapter_07-teen_gambling.csv')

print("First five observations:")
print(teengamb.head())

# --------------------------------------------
# Step 3: Estimate regression model (OLS)
# --------------------------------------------
# This step helps us understand relationships
# before performing ANCOVA

model = ols(
    'gamble ~ income + sex + status + verbal',
    data=teengamb
).fit()

print("\nOLS Regression Results:")
print(model.summary())

# --------------------------------------------
# Step 4: Perform ANCOVA
```

```
# ----------------------------------------
# dv = dependent variable (gamble)
# covar = continuous control variables
# between = categorical factor (sex)

from pingouin import ancova

ancova_results = ancova(
    data=teengamb,
    dv='gamble',
    covar=['income', 'status', 'verbal'],
    between='sex'
)


print("\nANCOVA Results:")
print(ancova_results)
```

The Python output is below:

```
OLS Regression Results:
                            OLS Regression Results
==============================================================================
Dep. Variable:                 gamble   R-squared:                       0.527
Model:                            OLS   Adj. R-squared:                  0.482
Method:                 Least Squares   F-statistic:                     11.69
Date:                Mon, 06 Apr 2026   Prob (F-statistic):           1.81e-06
Time:                        11:34:21   Log-Likelihood:                -210.78
No. Observations:                  47   AIC:                             431.6
Df Residuals:                      42   BIC:                             440.8
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     22.5557     17.197      1.312      0.197     -12.149      57.260
income         4.9620      1.025      4.839      0.000       2.893       7.031
sex          -22.1183      8.211     -2.694      0.010     -38.689      -5.548
status         0.0522      0.281      0.186      0.853      -0.515       0.620
verbal        -2.9595      2.172     -1.362      0.180      -7.343       1.424
==============================================================================
Omnibus:                       31.143   Durbin-Watson:                   2.214
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              101.046
Skew:                           1.604   Prob(JB):                     1.14e-22
Kurtosis:                       9.427   Cond. No.                         264.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.

ANCOVA Results:
     Source            SS  DF          F     p_unc       np2
0       sex   3735.790512   1   7.256053  0.010112  0.147313
1    income  12056.238564   1  23.416920  0.000018  0.357964
```

```
2    status     17.775781   1   0.034526  0.853487  0.000821
3    verbal    955.734110   1   1.856329  0.180311  0.042328
4  Residual  21623.767055  42        NaN       NaN       NaN
```

We begin with the standard OLS regression results. The estimates indicate that sex and income are statistically significant predictors of gambling behavior.

The variable for sex is coded such that female = 1. The negative coefficient implies that females spend less on gambling compared to males, holding other variables constant. In contrast, income has a positive and statistically significant effect. It suggests that individuals with higher income tend to spend more on gambling.

The purpose of this analysis is to examine whether gambling behavior differs between genders while controlling for other factors. The ANCOVA results confirm that sex remains statistically significant. It indicates that there is a meaningful difference in gambling behavior between males and females even after accounting for income, status, and verbal ability.

Among the covariates, income is the only variable that is statistically significant, reinforcing the idea that financial resources play an important role in gambling behavior. The other variables, status and verbal ability, do not have a statistically significant independent effect in this model.

Problem 3.

The Python code is below:

```
# ============================================
# Discriminant Analysis: Loan Default Example
# ============================================

# --------------------------------------------
# Step 1: Import required libraries
# --------------------------------------------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.metrics import accuracy_score, confusion_matrix

# --------------------------------------------
# Step 2: Load the dataset
# --------------------------------------------
```

```
df = pd.read_csv('Chapter_07-GermanCredit.csv')

# ------------------------------------------
# Step 3: Data preparation
# ------------------------------------------
# Select relevant variables
df_model = df[[
    'duration',
    'amount',
    'age',
    'installment_rate',
    'number_credits',
    'credit_risk'
]].copy()

# Drop missing values
df_model = df_model.dropna()

# Define features (X) and target (y)
X = df_model[['duration', 'amount', 'age', 'installment_rate', 'number_credits']]
y = df_model['credit_risk']

# ------------------------------------------
# Step 4: Exploratory analysis
# ------------------------------------------
print("\nSummary statistics by credit risk:")
print(df_model.groupby('credit_risk').mean())

# Visualize distribution
plt.figure()
sns.histplot(data=df_model, x='amount', hue='credit_risk', kde=True)
plt.title("Loan Amount Distribution by Credit Risk")
plt.show()

# ------------------------------------------
# Step 5: Split into training and testing sets
# ------------------------------------------
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y   # ensures balanced classes in both sets
)

# ------------------------------------------
# Step 6: Apply Linear Discriminant Analysis (LDA)
# ------------------------------------------
lda = LinearDiscriminantAnalysis()

lda.fit(X_train, y_train)

# Predict on test data
y_pred = lda.predict(X_test)
```

```
# ------------------------------------------
# Step 7: Evaluate model performance
# ------------------------------------------
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print(f"\nAccuracy: {accuracy:.2f}")

print("\nConfusion Matrix:")
print(conf_matrix)

# ------------------------------------------
# Step 8: Visualize confusion matrix
# ------------------------------------------
plt.figure()
sns.heatmap(
    conf_matrix,
    annot=True,
    fmt="d",
    cbar=False,
    square=True,
    cmap="Blues"
)

plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

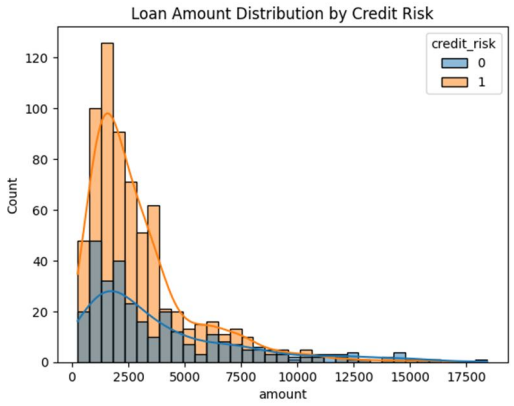
a. Compute summary statistics for the explanatory variables. Compare borrowers who default with those who do not.

The Python output is below:

```
Summary statistics by credit risk:
              duration       amount        age  installment_rate  \
credit_risk
0            24.860000  3938.126667  33.963333          3.096667
1            19.207143  2985.457143  36.224286          2.920000

              number_credits
credit_risk
0                   1.366667
1                   1.424286
```

> Borrowers who default tend to have shorter loan durations, smaller loan amounts, and are, on average, older than those who do not default.

b. Select one key variable and visualize its distribution using a histogram. Compare the distributions across the two groups.

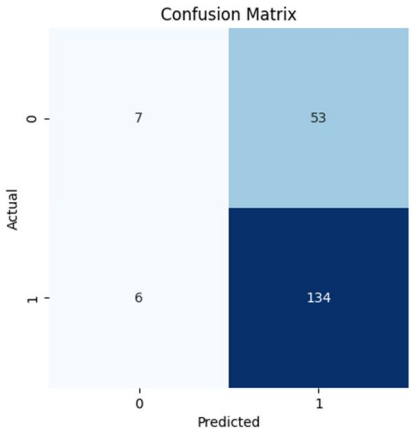


For all loan amounts, there are more borrowers who default than borrowers who do not default.

c. Estimate a discriminant model using Linear Discriminant Analysis (LDA).

- Split the data into training (80%) and testing (20%) sets
- Use the model to classify borrowers
- Construct and interpret a confusion matrix

The confusion matrix is below:



The LDA model performs better at identifying borrowers who default than those who do not. Specifically, 134 borrowers who default were correctly classified, while only 6 were misclassified.

In contrast, the model performs poorly for borrowers who do not default. Only 7 non-defaulting borrowers were correctly identified, while 53 were incorrectly classified as defaulting.

Problem 4.

The Python code is below:

```
# ============================================
# Cluster Analysis: Customer Segmentation
# ============================================

# --------------------------------------------
# Step 1: Import required libraries
# --------------------------------------------
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.cluster import KMeans

# --------------------------------------------
# Step 2: Load the dataset
# --------------------------------------------

df = pd.read_csv('Chapter_07-mall_customers.csv')

# --------------------------------------------
# Step 3: Explore the data
# --------------------------------------------
print("First five observations:")
print(df.head())

print("\nSummary statistics:")
print(df.describe())

# --------------------------------------------
# Step 4: Visualize the data
# --------------------------------------------
plt.figure()
sns.scatterplot(
    data=df,
    x='Annual Income (k$)',
    y='Spending Score (1-100)'
)

plt.title('Customer Distribution')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score')
plt.show()

# --------------------------------------------
# Step 5: Determine number of clusters (Elbow Method)
# --------------------------------------------
X = df[['Annual Income (k$)', 'Spending Score (1-100)']]
```

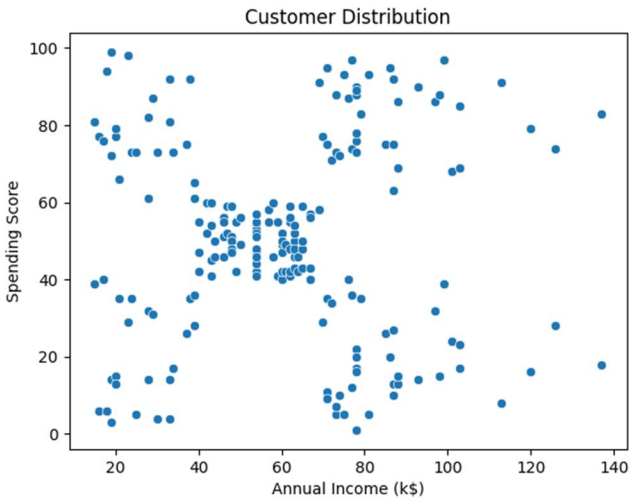
```
inertia = []

for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X)
    inertia.append(kmeans.inertia_)

# Plot Elbow curve
plt.figure()
plt.plot(range(1, 11), inertia, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.show()

# -----------------------------------------
# Step 6: Apply K-means using chosen k
# -----------------------------------------
# Based on the elbow plot, students select k (typically k = 5)
optimal_k = 5

kmeans = KMeans(n_clusters=optimal_k, random_state=42)
df['Cluster'] = kmeans.fit_predict(X)

# -----------------------------------------
# Step 7: Visualize clusters
# -----------------------------------------
plt.figure()
sns.scatterplot(
    data=df,
    x='Annual Income (k$)',
    y='Spending Score (1-100)',
    hue='Cluster',
    palette='Set2'
)

plt.title('Customer Segments')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score')
plt.legend(title='Cluster')
plt.show()

# -----------------------------------------
# Step 8: Examine cluster characteristics
# -----------------------------------------
print("\nCluster Summary:")
print(df.groupby('Cluster')[['Age', 'Annual Income (k$)', 'Spending Score (1-
100)']].mean())
```

a. We explore the data, display the first five observations, and compute summary statistics.

The Python output is below:

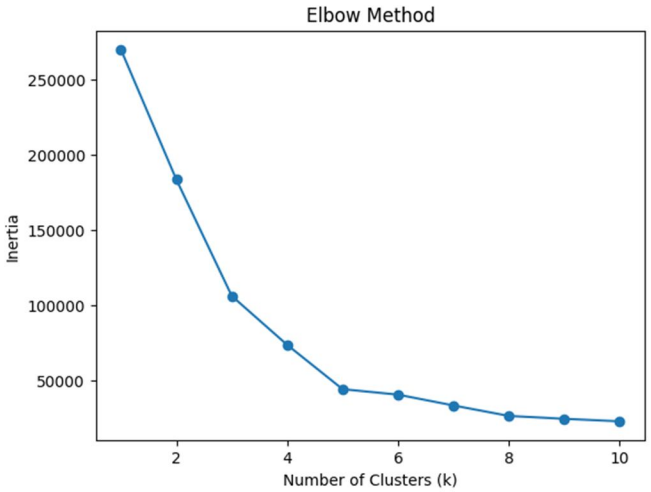
```
First five observations:
   CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
0           1    Male   19                  15                      39
1           2    Male   21                  15                      81
2           3  Female   20                  16                       6
3           4  Female   23                  16                      77
4           5  Female   31                  17                      40

Summary statistics:
       CustomerID         Age  Annual Income (k$)  Spending Score (1-100)
count  200.000000  200.000000          200.000000              200.000000
mean   100.500000   38.850000           60.560000               50.200000
std     57.879185   13.969007           26.264721               25.823522
min      1.000000   18.000000           15.000000                1.000000
25%     50.750000   28.750000           41.500000               34.750000
50%    100.500000   36.000000           61.500000               50.000000
75%    150.250000   49.000000           78.000000               73.000000
max    200.000000   70.000000          137.000000               99.000000
```

b. We create a scatter plot of annual income vs spending score. What patterns do we observe?



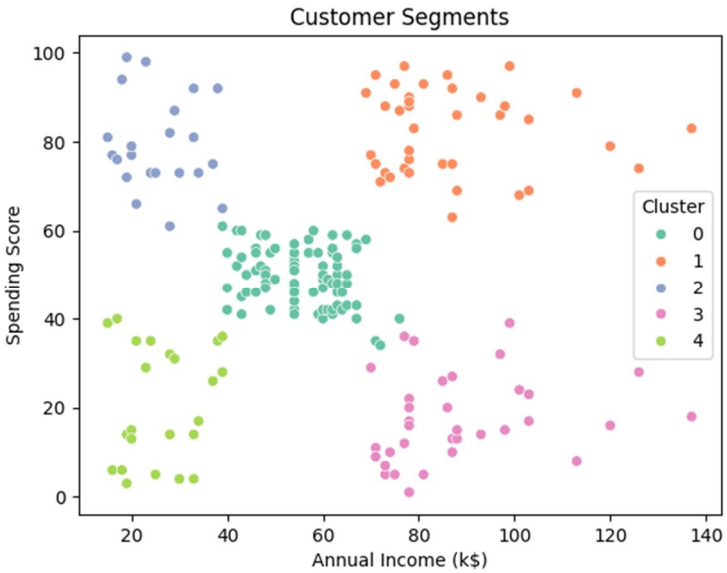
It appears that our customers form five clusters.

c. We determine the number of clusters. We use the Elbow Method to identify the optimal number of clusters.



The elbow bends at 5 clusters.

d. We plot the clusters using a scatter plot. We use different colors for each cluster.



e. We describe each cluster in plain language. For example: “High income, high spending.” How could the company use these clusters for marketing?

We print out the average for each cluster.

```
Cluster Summary:
               Age  Annual Income (k$)  Spending Score (1-100)
```

```
Cluster
0        42.716049          55.296296          49.518519
1        32.692308          86.538462          82.128205
2        25.272727          25.727273          79.363636
3        41.114286          88.200000          17.114286
4        45.217391          26.304348          20.913043
```

The results show that customers naturally divide into distinct groups based on income and spending behavior. Among customers with high annual income, there are two clear segments: high spenders and low spenders. Similarly, customers with low annual income also split into high spenders and low spenders.

In contrast, customers with middle income levels tend to exhibit moderate spending behavior, falling between the high- and low-spending groups.

From a marketing perspective, the company should focus on at least two key segments:

- High-income, high-spending customers, who represent the most valuable group and should be targeted with premium products and loyalty programs
- Low-income, high-spending customers, who may be highly engaged and responsive to promotions, making them an important group for targeted marketing campaigns

By tailoring strategies to these segments, the company can improve customer engagement and maximize revenue.

Problem 5.

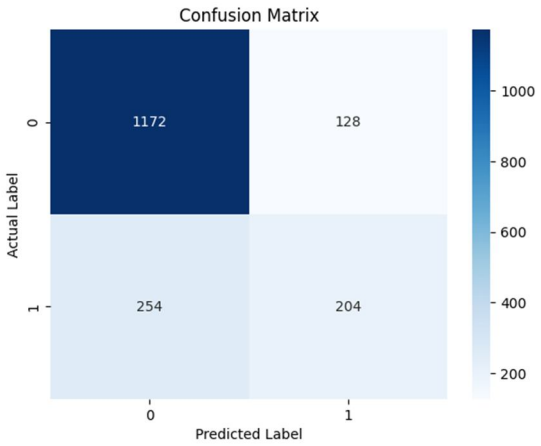
The Python code is below:

```
# ==========================================
# Logistic Regression: Customer Churn Example
# ==========================================

# ------------------------------------------
# Step 1: Import required libraries
# ------------------------------------------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import classification_report

import statsmodels.api as sm
```

```
# ------------------------------------------
# Step 2: Load the dataset
# ------------------------------------------

df = pd.read_csv('Chapter_07-churn_data.csv')

print("First five observations:")
print(df.head())

# ------------------------------------------
# Step 3: Data cleaning
# ------------------------------------------
# Convert TotalCharges to numeric (it may contain spaces)
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')

# Drop missing values
df = df.dropna()

# Convert Churn to binary (Yes=1, No=0)
df['Churn'] = df['Churn'].map({'Yes': 1, 'No': 0})

# ------------------------------------------
# Step 4: Select variables
# ------------------------------------------
# Keep a simple set of variables for teaching clarity
df_model = df[[
    'tenure',
    'MonthlyCharges',
    'TotalCharges',
    'SeniorCitizen',
    'Churn'
]]

# Define X and y
X = df_model[['tenure', 'MonthlyCharges', 'TotalCharges', 'SeniorCitizen']]
y = df_model['Churn']

# ------------------------------------------
# Step 5: Split data into training and testing
# ------------------------------------------
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42
)

# ------------------------------------------
# Step 6: Estimate logistic regression (sklearn)
# ------------------------------------------
model = LogisticRegression(max_iter=1000)

model.fit(X_train, y_train)

# Predictions
```

```
y_pred = model.predict(X_test)

# ------------------------------------------
# Step 7: Evaluate model performance
# ------------------------------------------
conf_matrix = metrics.confusion_matrix(y_test, y_pred)

print("\nConfusion Matrix:")
print(conf_matrix)

# ------------------------------------------
# Step 8: Visualize confusion matrix
# ------------------------------------------
plt.figure()
sns.heatmap(
    pd.DataFrame(conf_matrix),
    annot=True,
    fmt='g',
    cmap='Blues'
)

plt.title('Confusion Matrix')
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
plt.show()

# ------------------------------------------
# Step 9: Classification report
# ------------------------------------------
target_names = ['No Churn', 'Churn']

print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=target_names))

# ------------------------------------------
# Step 10: Estimate Logit model (statsmodels)
# ------------------------------------------
# Add constant for intercept
X_train_sm = sm.add_constant(X_train)

# Fit Logit model
logit_model = sm.Logit(
    y_train.astype(float),
    X_train_sm.astype(float)
).fit()

print("\nLogit Model Summary:")
print(logit_model.summary())
```

The confusion matrix is on the next page.



We observe that customers who remain with the company (No Churn) are classified quite accurately. Specifically, 1,172 customers were correctly classified as staying, while 128 were incorrectly classified.

However, predicting customers who leave (Churn) is more difficult. Although 204 customers were correctly identified as leaving, 254 customers who actually left were incorrectly classified as staying. This indicates that the model has more difficulty detecting churn behavior.

The classification report is below:

```
Classification Report:
              precision    recall  f1-score   support

    No Churn       0.82      0.90      0.86      1300
       Churn       0.61      0.45      0.52       458

    accuracy                           0.78      1758
   macro avg       0.72      0.67      0.69      1758
weighted avg       0.77      0.78      0.77      1758
```

Overall, the model achieves an accuracy of 78%. It means that 78% of all customers are correctly classified.

- Customers who stay are correctly classified about 82% of the time
- Customers who leave are correctly classified only 61% of the time (precision) and identified 45% of the time (recall)

This suggests that the model is better at predicting customer retention than customer churn. In practice, this is a common issue because churn cases are often less frequent and harder to predict.

The logistic regression is below:

```
Optimization terminated successfully.
         Current function value: 0.445201
         Iterations 7

Logit Model Summary:
                           Logit Regression Results
==============================================================================
Dep. Variable:                  Churn   No. Observations:                 5274
Model:                          Logit   Df Residuals:                     5269
Method:                           MLE   Df Model:                            4
Date:                Mon, 06 Apr 2026   Pseudo R-squ.:                 0.2335
Time:                        06:29:28   Log-Likelihood:                -2348.0
converged:                       True   LL-Null:                       -3063.1
Covariance Type:            nonrobust   LLR p-value:                1.907e-308
==============================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const             -1.5674      0.135    -11.588      0.000      -1.833      -1.302
tenure            -0.0731      0.007    -11.157      0.000      -0.086      -0.060
MonthlyCharges     0.0284      0.002     14.220      0.000       0.024       0.032
TotalCharges       0.0002   7.29e-05      2.800      0.005    6.13e-05       0.000
SeniorCitizen      0.6783      0.092      7.392      0.000       0.498       0.858
==============================================================================
```

All variables are statistically significant at the 5% level. It indicates that each variable contributes meaningfully to explaining customer churn.

- Tenure has a negative coefficient. Customers who have been with the company longer are less likely to leave.
- Monthly Charges has a positive coefficient. Higher monthly costs increase the likelihood of churn.
- Total Charges has a small positive coefficient. Customers who pay more overall are slightly more likely to leave.
- Senior Citizen has a positive coefficient. Older customers are more likely to leave

Chapter 8 Answers

Problem 1.

This is a standard feedforward neural network:

- Input layer → all features
- Hidden layer 1 → 16 neurons (ReLU)
- Hidden layer 2 → 8 neurons (ReLU)

- Output layer → 1 neuron (Sigmoid for probability)

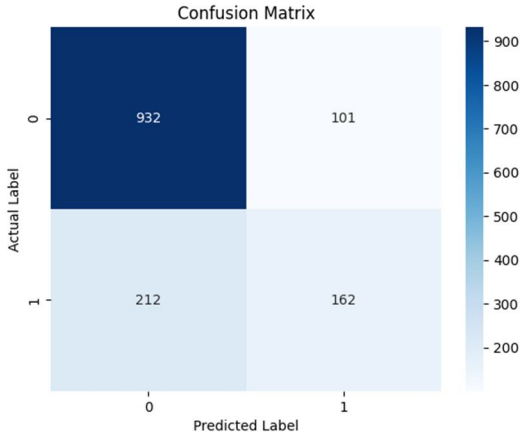
The Python code is below:

```
# ================================
# 1. Import libraries
# ================================
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

# ================================
# 2. Load dataset
# ================================
df = pd.read_csv("Chapter_07-churn_data.csv")

# ================================
# 3. Drop missing values (as requested)
# ================================

# Convert to numeric → blanks become NaN
df["TotalCharges"] = pd.to_numeric(df["TotalCharges"], errors="coerce")

# Drop missing values
df = df.dropna()

# ================================
# 4. Encode target variable
# ================================
df["Churn"] = df["Churn"].map({"Yes": 1, "No": 0})

# ================================
# 5. Encode categorical variables
# ================================
categorical_cols = [
    "Contract",
    "InternetService",
    "PaymentMethod"
]

df = pd.get_dummies(df, columns=categorical_cols, drop_first=True)

# ================================
```

```
# 6. Features and target
# ================================

X = df[['tenure', 'MonthlyCharges', 'TotalCharges', 'SeniorCitizen']]
y = df['Churn']

# ================================
# 7. Train-test split
# ================================
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# ================================
# 8. Scale features
# ================================
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# ================================
# 9. Build neural network
# ================================

model = keras.Sequential([
    keras.Input(shape=(X_train.shape[1],)),  # explicit input layer
    layers.Dense(16, activation='relu'),
    layers.Dense(8, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])

# ================================
# 10. Compile model
# ================================
model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
)

# ================================
# 11. Early stopping callback
# ================================
early_stop = keras.callbacks.EarlyStopping(
    monitor='val_loss',     # watch validation loss
    patience=5,             # stop after 5 epochs of no improvement
    restore_best_weights=True
)

# ================================
# 12. Train model
# ================================
history = model.fit(
```

```
    X_train, y_train,
    epochs=100,                  # set high, early stopping will control it
    batch_size=32,
    validation_split=0.2,
    callbacks=[early_stop],
    verbose=1
)

# ================================
# 13. Evaluate model
# ================================
loss, accuracy = model.evaluate(X_test, y_test)
print(f"\nTest Accuracy: {accuracy:.4f}")

# ================================
# 14. Predictions
# ================================
y_pred_prob = model.predict(X_test)
y_pred = (y_pred_prob > 0.5).astype(int)

# ------------------------------------------
# 15. Visualize confusion matrix
# ------------------------------------------
plt.figure()
sns.heatmap(
    pd.DataFrame(confusion_matrix(y_test, y_pred)),
    annot=True,
    fmt='g',
    cmap='Blues'
)

plt.title('Confusion Matrix')
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
plt.show()

print()
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

The confusion matrix is on the page:



The classification report is below:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.90      0.86      1033
           1       0.62      0.43      0.51       374

    accuracy                           0.78      1407
   macro avg       0.72      0.67      0.68      1407
weighted avg       0.76      0.78      0.76      1407
```

We get similar results as the logistic regression in Chapter 7.

Problem 2.

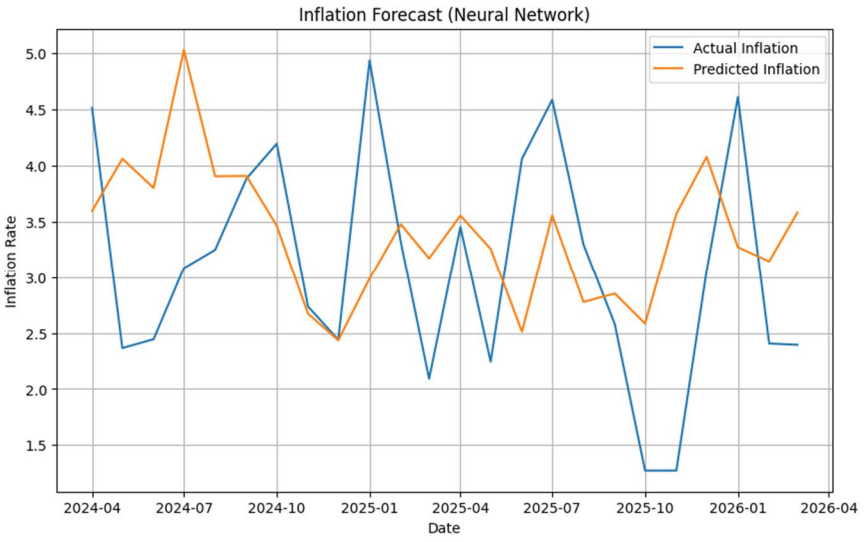
The Python code is below:

```
# ================================
# 1. Import libraries
# ================================
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

# ================================
# 2. Load data
```

```
# ================================
df = pd.read_csv("Chapter_08-inflation.csv")

# Convert date
df["Date"] = pd.to_datetime(df["Date"])
df = df.sort_values("Date")

# ================================
# 3. Create lagged features
# ================================
def create_lags(data, lags=12):
    df_lag = data.copy()
    for i in range(1, lags+1):
        df_lag[f"lag_{i}"] = df_lag["Inflation"].shift(i)
    return df_lag

df_lag = create_lags(df, lags=12)

# Drop missing values from lags
df_lag = df_lag.dropna()

# ================================
# 4. Train-test split (last 2 years = test)
# ================================
# 24 months test set
train = df_lag.iloc[:-24]
test = df_lag.iloc[-24:]

X_train = train.drop(["Date", "Inflation"], axis=1)
y_train = train["Inflation"]

X_test = test.drop(["Date", "Inflation"], axis=1)
y_test = test["Inflation"]

# ================================
# 5. Scale features
# ================================
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# ================================
# 6. Build neural network
# ================================
model = keras.Sequential([
    keras.Input(shape=(X_train_scaled.shape[1],)),
    layers.Dense(32, activation='relu'),
    layers.Dense(16, activation='relu'),
    layers.Dense(1)
])

# Compile
model.compile(
```

```
    optimizer='adam',
    loss='mse'
)

# Early stopping
early_stop = keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=5,
    restore_best_weights=True
)

# ================================
# 7. Train model
# ================================
history = model.fit(
    X_train_scaled, y_train,
    epochs=100,
    batch_size=16,
    validation_split=0.2,
    callbacks=[early_stop],
    verbose=1
)

# ================================
# 8. Forecast (out-of-sample)
# ================================
y_pred = model.predict(X_test_scaled).flatten()

# ================================
# 9. Evaluation
# ================================
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
mae = mean_absolute_error(y_test, y_pred)

print(f"RMSE: {rmse:.4f}")
print(f"MAE:  {mae:.4f}")

# ================================
# 10. Plot results
# ================================
plt.figure(figsize=(10,6))

plt.plot(test["Date"], y_test, label="Actual Inflation")
plt.plot(test["Date"], y_pred, label="Predicted Inflation")

plt.title("Inflation Forecast (Neural Network)")
plt.xlabel("Date")
plt.ylabel("Inflation Rate")
plt.legend()
plt.grid()

plt.show()
```

The forecast is below:



The neuronal network captures some fluctuations and misses others.