Data Analysis in Natural Sciences: An R-Based Approach

From Field to Figures: Essential Methods for Environmental and Life Sciences

Dr. Jimmy Moses (PhD)

2025-03-27

Table of contents

Preface

Welcome to **Data Analysis in Natural Sciences: An R-Based Approach**, a comprehensive guide designed for students, professionals, and researchers across the natural sciences. This book provides practical methods for analyzing and visualizing data using R, with applications spanning forestry, agriculture, ecology, marine biology, environmental science, and more.

About the Author

This book has been developed by **Dr. Jimmy Moses (PhD)** from the School of Forestry, Faculty of Natural Resources, Papua New Guinea University of Technology. With extensive experience in ecological research and data analysis, Dr. Moses has created this resource to support students and researchers in developing essential analytical skills for natural science disciplines.

Target Audience

This book is designed for:

- **Undergraduate and postgraduate students** in natural science disciplines
- **Researchers** seeking to enhance their data analysis capabilities
- **Technicians** working in laboratories and field settings
- **Professionals** in government agencies, NGOs, and private sector
- **Hobbyists** with an interest in analyzing environmental data

The content is relevant to those working in:

- Forestry and agroforestry
- Agriculture and agronomy
- Ecology and conservation
- Environmental science
- Geography and GIS/remote sensing
- Marine biology and fisheries
- Botany and plant sciences
- Entomology and zoology
- Epidemiology and veterinary sciences
- Forest economics and natural resource management

What You Will Learn

This book will guide you through:

- The fundamentals of data analysis with R
- Data preparation and management techniques
- Exploratory data analysis approaches

- Statistical hypothesis testing
- Advanced visualization methods
- Specialized analyses for environmental data
- Reproducible research practices

How to Use This Book

This book is designed to be both a learning resource and a reference guide. You can read it from start to finish to build your skills progressively, or use specific chapters as needed for particular tasks.

Code examples are provided throughout, and you can run them directly in R or RStudio. Each chapter includes practical examples using real datasets from various natural science disciplines.

Prerequisites

To get the most out of this book, you should have:

- Basic computer skills
- R and RStudio installed (instructions provided in Chapter 1)
- A basic understanding of statistics (helpful but not required)

Acknowledgments

I would like to thank all those who contributed to the development of this book, including colleagues, students, and the open-source community that makes tools like R and RStudio possible.

Let's begin our journey into the world of data analysis for natural sciences!

About This Book

About the Author

Dr. Jimmy Moses is a Papua New Guinean entomologist and lecturer at the Papua New Guinea University of Technology's School of Forestry, specializing in ant ecology, biostatistics, and geospatial analysis. He holds a Ph.D. in Entomology from the University of South Bohemia (2021) and has extensive experience in tropical ecology research, particularly focusing on ant communities along elevational gradients.

He currently supervises four master's students and co-supervises a Ph.D. student, Dr. Moses brings significant expertise in both research and education. He has published several peer-reviewed papers, including work in prestigious journals like *Global Ecology and Biogeography* and *Proceedings of the Royal Society B*.

His technical skills span multiple areas:

- Advanced proficiency in R and Python for statistical computing and data science
- Expertise in GIS and Satellite Remote Sensing
- Strong background in biostatistics and experimental design
- Emerging skills in full-stack app development

Dr. Moses has maintained strong international collaborations, having worked with institutions in the Czech Republic, Germany, and Belgium. He has been actively involved with the New Guinea Binatang Research Center, contributing to both research and education initiatives in Papua New Guinea.

His research interests combine ecological field studies with modern analytical approaches, particularly in ant ecology, spatial ecology, macroecology, and crop protection. He spends his free time reading technical, historical, psychological and ecological books and more time tinkering with codes.

Purpose and Scope

This book is designed to serve as both a learning resource and a reference guide for data analysis in the natural sciences, with applications spanning forestry, agriculture, ecology, environmental science, marine biology, and related disciplines. Whether you're a student, researcher, technician, professional, or hobbyist in these fields, this book will help you develop the skills needed to analyze and visualize data effectively using R.

The focus is on practical applications rather than theoretical statistics, with an emphasis on techniques commonly used across natural science disciplines. By working through this book, you will:

- Master the fundamentals of data analysis in R
- Learn to import, clean, and organize various types of scientific data
- Develop skills in exploratory data analysis and visualization
- Apply appropriate statistical tests for different research questions
- Create publication-quality visualizations
- Implement reproducible research workflows
- Interpret and communicate results effectively

Features of This Book

This book includes:

- Step-by-step instructions for R with complete code examples
- Practical examples using real datasets from various natural science disciplines
- Exercises to reinforce learning and build skills
- Tips and best practices from experienced researchers
- Reproducible code that can be adapted for your own research

How to Use the Code Examples

All code examples in this book are written in R and can be executed in RStudio. To use the examples:

1. Make sure you have R and RStudio installed (see Chapter 1 for installation instructions)
2. Install the required packages mentioned at the beginning of each chapter
3. Copy and paste the code into your R console or script editor
4. Modify the code as needed for your own data

The datasets used in the examples are available in the `docs/data` directory of the book's repository and are properly cited throughout the text.

Software Requirements

This book uses:

- R (version 4.0.0 or higher)
- RStudio (latest version recommended)
- Various R packages (installation instructions provided in each chapter)

Feedback and Contributions

Your feedback is valuable for improving future editions of this book. If you find errors, have suggestions, or want to contribute examples, please submit them through the book's repository or contact the author directly.

Acknowledgments

I would like to express my gratitude to colleagues, students, and the broader R community whose insights and feedback have contributed to the development of this book. Special thanks to the creators and maintainers of the R packages used throughout this book, as well as the data providers whose datasets make the examples both practical and relevant.

Part I

Getting Started

Chapter 1

Introduction to Data Analysis

1.1 Overview

Data analysis is a critical skill in modern ecological and forestry research (Wickham & Grolemund, 2016; Zuur et al., 2009). This chapter introduces the fundamental concepts, tools, and approaches that form the foundation of effective data analysis.

1.2 Why Data Analysis Matters in Ecology and Forestry

Data analysis plays a pivotal role in ecological and forestry research for several reasons:

1. **Evidence-Based Decision Making**: Data analysis transforms raw observations into actionable insights, enabling researchers and practitioners to make informed decisions about conservation strategies, forest management practices, and ecological interventions (Bolker et al., 2009).
2. **Pattern Recognition**: Through statistical analysis, researchers can identify patterns, trends, and relationships within ecological systems that might not be apparent from casual observation alone (Zuur et al., 2007).
3. **Hypothesis Testing**: Data analysis provides rigorous methods to test hypotheses about ecological phenomena, allowing researchers to build and refine scientific theories about how ecosystems function (Gotelli & Ellison, 2004).
4. **Prediction and Modeling**: Advanced analytical techniques enable the development of predictive models that can forecast ecological changes, such as species distribution shifts under climate change or forest growth patterns (Elith et al., 2009).

1.3 Tools for Data Analysis

This book focuses on two primary tools for data analysis:

1.3.1 R and RStudio

R is a powerful programming language and environment specifically designed for statistical computing and graphics. RStudio is an integrated development environment (IDE) that makes working with R more accessible and efficient.

Key advantages of R include:

- **Open-source and free**: Available to anyone without cost

- **Extensive package ecosystem**: Thousands of specialized packages for various types of analyses
- **Reproducibility**: Code-based approach ensures analyses can be repeated and verified
- **Flexibility**: Can be adapted to virtually any analytical need
- **Active community**: Large user base provides support and continuous development

```r
# A simple example of R code
data(iris)
head(iris)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

```r
summary(iris$Sepal.Length)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.300   5.100   5.800   5.843   6.400   7.900
```

1.3.2 Jamovi

Jamovi is a free, open-source statistical software that provides a user-friendly graphical interface for common statistical analyses. It's particularly helpful for those who prefer not to write code.

Key advantages of Jamovi include:

- **Intuitive interface**: Point-and-click approach with immediate visual feedback
- **R integration**: Built on R, allowing access to its statistical capabilities
- **Reproducibility**: Automatically records analyses for future reference
- **Modern design**: Clean, contemporary interface with publication-ready outputs
- **Extensibility**: Modules can be added to extend functionality

1.4 Setting Up Your Environment

1.4.1 Installing R and RStudio

To install R and RStudio:

1. Download and install R from CRAN
2. Download and install RStudio from RStudio's website

1.4.2 Installing Jamovi

To install Jamovi:

1. Download and install Jamovi from Jamovi's website

1.4.3 Essential R Packages

For the analyses in this book, you'll need several R packages. You can install them with the following code:

```r
install.packages(c(
  "tidyverse",  # Data manipulation and visualization
  "rstatix",    # Statistical tests
  "ggplot2",    # Advanced plotting
```

```
  "knitr",      # Document generation
  "rmarkdown"   # Document formatting
))
```

1.5 The Data Analysis Workflow

Effective data analysis typically follows a structured workflow:

1. **Define the Question**: Clearly articulate what you want to learn from your data
2. **Collect Data**: Gather the necessary data through fieldwork, experiments, or existing datasets
3. **Clean and Prepare Data**: Handle missing values, correct errors, and format data appropriately
4. **Explore Data**: Conduct exploratory data analysis to understand patterns and distributions
5. **Analyze Data**: Apply appropriate statistical methods to address your research questions
6. **Interpret Results**: Draw conclusions based on your analysis
7. **Communicate Findings**: Present your results through visualizations, reports, or publications

Throughout this book, we'll follow this workflow as we explore various ecological and forestry datasets.

1.6 Types of Data in Ecological Research

Ecological and forestry research involves several types of data:

1.6.1 Categorical Data

Categorical data represent qualitative characteristics, such as: - Species names - Habitat types - Land-use categories - Management regimes

1.6.2 Numerical Data

Numerical data involve measurements or counts: - Continuous measurements (e.g., tree height, temperature, carbon content) - Discrete counts (e.g., number of individuals, species richness)

1.6.3 Spatial Data

Spatial data describe geographical distributions: - Coordinates (latitude/longitude) - Elevation - Topographic features - Land cover maps

1.6.4 Temporal Data

Temporal data track changes over time: - Time series of measurements - Seasonal patterns - Long-term monitoring data

Understanding the type of data you're working with is crucial for selecting appropriate analytical methods.

1.7 Summary

In this chapter, we've introduced the importance of data analysis in ecological and forestry research and the tools we'll be using throughout this book. We've also outlined the typical data analysis workflow and the types of data commonly encountered in ecological studies.

In the next chapter, we'll dive deeper into data basics, learning how to import, clean, and prepare data for analysis.

1.8 Exercises

1. Install R, RStudio, and Jamovi on your computer.
2. Install the required R packages listed in this chapter.
3. Open RStudio and create a new R script. Try running a simple command like `summary(iris)`.
4. Open Jamovi and explore its interface. Try loading the built-in "Tooth Growth" dataset.
5. Think about a research question in ecology or forestry that interests you. What type of data would you need to address this question?

Chapter 2

Data Basics

2.1 Introduction

This chapter covers the fundamental concepts of working with data in R and Jamovi. You'll learn how to import, clean, and prepare data for analysis, which are essential skills for any data analysis project.

2.2 Understanding Data Structures

2.2.1 Data Types in R

R has several basic data types that you'll encounter frequently:

1. **Numeric**: Numbers such as 1, 2.5, -3.14
2. **Character**: Text values like "species", "forest", "treatment"
3. **Logical**: Boolean values TRUE or FALSE
4. **Factor**: Categorical variables with predefined levels
5. **Date/Time**: Specialized formats for dates and times

```
# Examples of different data types
numeric_example <- 42.5
character_example <- "Ecological data"
logical_example <- TRUE
factor_example <- factor(c("Control", "Treatment", "Control"),
                         levels = c("Control", "Treatment"))
date_example <- as.Date("2025-03-27")

# Check the data types
str(numeric_example)
```

```
 num 42.5
```

```
str(character_example)
```

```
 chr "Ecological data"
```

```
str(logical_example)
```

```
 logi TRUE
```

```
str(factor_example)
```

```
 Factor w/ 2 levels "Control","Treatment": 1 2 1
```

```
str(date_example)
```

```
 Date[1:1], format: "2025-03-27"
```

2.2.2 Data Structures in R

R uses several data structures to organize and store data:

1. **Vectors**: One-dimensional arrays of elements of the same type
2. **Matrices**: Two-dimensional arrays of elements of the same type
3. **Data Frames**: Two-dimensional structures that can contain different types of data
4. **Lists**: Collections of objects that can be of different types
5. **Arrays**: Multi-dimensional structures of elements of the same type

```
# Vector example
tree_heights <- c(15.2, 18.7, 12.3, 20.1, 17.5)
print(tree_heights)
```

```
[1] 15.2 18.7 12.3 20.1 17.5
```

```
# Matrix example
soil_samples <- matrix(1:12, nrow = 4, ncol = 3)
colnames(soil_samples) <- c("pH", "Nitrogen", "Phosphorus")
print(soil_samples)
```

```
     pH Nitrogen Phosphorus
[1,]  1        5          9
[2,]  2        6         10
[3,]  3        7         11
[4,]  4        8         12
```

```
# Data frame example
forest_data <- data.frame(
  species = c("Oak", "Pine", "Maple", "Birch"),
  height = c(18.2, 22.5, 15.7, 12.8),
  diameter = c(45, 30, 25, 20),
  age = c(120, 80, 60, 40)
)
print(forest_data)
```

```
  species height diameter age
1     Oak   18.2       45 120
2    Pine   22.5       30  80
3   Maple   15.7       25  60
4   Birch   12.8       20  40
```

```
# List example
ecosystem <- list(
  forest_type = "Temperate",
  dominant_species = c("Oak", "Maple", "Beech"),
  average_rainfall = 1200,
  temperature_range = c(-5, 30)
)
print(ecosystem)
```

```
$forest_type
[1] "Temperate"
```

```
$dominant_species
[1] "Oak"   "Maple" "Beech"

$average_rainfall
[1] 1200

$temperature_range
[1] -5 30
```

2.3 Importing Data

2.3.1 Reading Data from Files

R provides several functions for importing data from different file formats:

```
# CSV files
data_csv <- read.csv("data/forest_inventory.csv")

# Tab-delimited files
data_tab <- read.delim("data/species_counts.txt", header = TRUE)

# Excel files (requires readxl package)
library(readxl)
data_excel <- read_excel("data/climate_data.xlsx", sheet = "Temperature")

# Using the tidyverse approach
library(tidyverse)
data_csv_tidy <- readr::read_csv("data/forest_inventory.csv")
```

2.3.2 Using Our Downloaded Forestry Dataset

Let's explore the forestry dataset we've downloaded:

```
# Basic read.csv function
forest_data <- read.csv("../data/forestry/forest_inventory.csv")
head(forest_data)
```

```
            name height mass  hair_color  skin_color eye_color birth_year
1 Luke Skywalker    172   77       blond        fair      blue       19.0
2          C-3PO    167   75        <NA>        gold    yellow      112.0
3          R2-D2     96   32        <NA> white, blue       red       33.0
4    Darth Vader    202  136        none       white    yellow       41.9
5    Leia Organa    150   49       brown       light     brown       19.0
6      Owen Lars    178  120 brown, grey       light      blue       52.0
     sex    gender homeworld species
1   male masculine  Tatooine   Human
2   none masculine  Tatooine   Droid
3   none masculine     Naboo   Droid
4   male masculine  Tatooine   Human
5 female  feminine  Alderaan   Human
6   male masculine  Tatooine   Human

1                                          A New Hope, The Empire Strikes Back, Return of the Jedi, Rev
2                   A New Hope, The Empire Strikes Back, Return of the Jedi, The Phantom Menace, Attacl
```

```
3 A New Hope, The Empire Strikes Back, Return of the Jedi, The Phantom Menace, Attack of the Clones, Rev
4                                                                    A New Hope, The Empire Strikes Back, Retu
5                                                A New Hope, The Empire Strikes Back, Return of the Jedi, Rev
6                                                                                          A New Hope, Attacl
                           vehicles                starships
1 Snowspeeder, Imperial Speeder Bike X-wing, Imperial shuttle
2
3
4                                          TIE Advanced x1
5              Imperial Speeder Bike
6
```

```
# Using readr package for better performance and control
library(readr)
forest_data_tidy <- read_csv("../data/forestry/forest_inventory.csv")
head(forest_data_tidy)
```

```
# A tibble: 6 x 14
  name      height  mass hair_color skin_color eye_color birth_year sex   gender
  <chr>      <dbl> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
1 Luke Sky~    172    77 blond      fair       blue            19   male  mascu~
2 C-3PO        167    75 <NA>       gold       yellow         112   none  mascu~
3 R2-D2         96    32 <NA>       white, bl~ red             33   none  mascu~
4 Darth Va~    202   136 none       white      yellow          41.9 male  mascu~
5 Leia Org~    150    49 brown      light      brown           19   fema~ femin~
6 Owen Lars    178   120 brown, gr~ light      blue            52   male  mascu~
# i 5 more variables: homeworld <chr>, species <chr>, films <chr>,
#   vehicles <chr>, starships <chr>
```

2.3.3 Importing Data in Jamovi

In Jamovi, you can import data by:

1. Clicking on the "Data" tab
2. Selecting "Import" from the menu
3. Choosing the file format (CSV, Excel, SPSS, etc.)
4. Navigating to and selecting your file
5. Adjusting import options if necessary

2.4 Data Cleaning and Preparation

2.4.1 Handling Missing Values

Missing values are common in ecological datasets and need to be addressed before analysis:

```
# Create a dataset with missing values
tree_data <- data.frame(
  height = c(15.2, NA, 12.3, 20.1, NA),
  diameter = c(45, 30, NA, 20, 35),
  age = c(120, 80, 60, NA, 90)
)

# Check for missing values
sum(is.na(tree_data))
```

```
[1] 4
```

```r
colSums(is.na(tree_data))
```

```
  height diameter      age
       2        1        1
```

```r
# Remove rows with any missing values
tree_data_complete <- na.omit(tree_data)
print(tree_data_complete)
```

```
  height diameter age
1   15.2       45 120
```

```r
# Replace missing values with the mean
tree_data$height[is.na(tree_data$height)] <- mean(tree_data$height, na.rm = TRUE)
tree_data$diameter[is.na(tree_data$diameter)] <- mean(tree_data$diameter, na.rm = TRUE)
tree_data$age[is.na(tree_data$age)] <- mean(tree_data$age, na.rm = TRUE)
print(tree_data)
```

```
    height diameter   age
1 15.20000     45.0 120.0
2 15.86667     30.0  80.0
3 12.30000     32.5  60.0
4 20.10000     20.0  87.5
5 15.86667     35.0  90.0
```

2.4.2 Data Transformation

Often, you'll need to transform your data to meet the assumptions of statistical tests or to make patterns more apparent:

```r
# Load a dataset
data(iris)

# Log transformation
iris$log_sepal_length <- log(iris$Sepal.Length)

# Square root transformation
iris$sqrt_sepal_width <- sqrt(iris$Sepal.Width)

# Standardization (z-score)
iris$z_petal_length <- scale(iris$Petal.Length)

# View the first few rows of the transformed data
head(iris[, c("Sepal.Length", "log_sepal_length",
              "Sepal.Width", "sqrt_sepal_width",
              "Petal.Length", "z_petal_length")])
```

```
  Sepal.Length log_sepal_length Sepal.Width sqrt_sepal_width Petal.Length
1          5.1         1.629241         3.5         1.870829          1.4
2          4.9         1.589235         3.0         1.732051          1.4
3          4.7         1.547563         3.2         1.788854          1.3
4          4.6         1.526056         3.1         1.760682          1.5
5          5.0         1.609438         3.6         1.897367          1.4
6          5.4         1.686399         3.9         1.974842          1.7
  z_petal_length
1      -1.335752
```

```
2      -1.335752
3      -1.392399
4      -1.279104
5      -1.335752
6      -1.165809
```

2.4.3 Creating New Variables

Creating new variables based on existing ones is a common task in data preparation:

```
# Create a dataset
forest_plots <- data.frame(
  plot_id = 1:5,
  length = c(100, 150, 120, 200, 180),
  width = c(80, 100, 90, 120, 110)
)

# Calculate area
forest_plots$area <- forest_plots$length * forest_plots$width

# Create a categorical variable
forest_plots$size_class <- cut(
  forest_plots$area,
  breaks = c(0, 10000, 15000, Inf),
  labels = c("Small", "Medium", "Large")
)

print(forest_plots)
```

```
  plot_id length width  area size_class
1       1    100    80  8000      Small
2       2    150   100 15000     Medium
3       3    120    90 10800     Medium
4       4    200   120 24000      Large
5       5    180   110 19800      Large
```

2.5 Data Manipulation with dplyr

The dplyr package from the tidyverse provides a powerful grammar for data manipulation:

```
library(dplyr)

# Sample dataset
data(starwars)

# Select specific columns
characters <- starwars %>%
  select(name, species, height, mass)

# Filter rows based on conditions
humans <- starwars %>%
  filter(species == "Human")

# Arrange data
tall_characters <- starwars %>%
```

```
  arrange(desc(height))

# Create new variables
bmi_data <- starwars %>%
  filter(!is.na(height), !is.na(mass)) %>%
  mutate(bmi = mass / ((height / 100)^2))

# Summarize data
species_summary <- starwars %>%
  group_by(species) %>%
  summarize(
    count = n(),
    avg_height = mean(height, na.rm = TRUE),
    avg_mass = mean(mass, na.rm = TRUE)
  ) %>%
  filter(count > 1) %>%
  arrange(desc(count))

# Display the results
head(characters)
```

```
# A tibble: 6 x 4
  name           species height  mass
  <chr>          <chr>    <int> <dbl>
1 Luke Skywalker Human      172    77
2 C-3PO          Droid      167    75
3 R2-D2          Droid       96    32
4 Darth Vader    Human      202   136
5 Leia Organa    Human      150    49
6 Owen Lars      Human      178   120
```

```
head(humans)
```

```
# A tibble: 6 x 14
  name      height  mass hair_color skin_color eye_color birth_year sex   gender
  <chr>      <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
1 Luke Sky~    172    77 blond      fair       blue            19   male  mascu~
2 Darth Va~    202   136 none       white      yellow          41.9 male  mascu~
3 Leia Org~    150    49 brown      light      brown           19   fema~ femin~
4 Owen Lars    178   120 brown, gr~ light      blue            52   male  mascu~
5 Beru Whi~    165    75 brown      light      blue            47   fema~ femin~
6 Biggs Da~    183    84 black      light      brown           24   male  mascu~
# i 5 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

```
head(tall_characters)
```

```
# A tibble: 6 x 14
  name      height  mass hair_color skin_color eye_color birth_year sex   gender
  <chr>      <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
1 Yarael P~    264    NA none       white      yellow            NA male  mascu~
2 Tarfful      234   136 brown      brown      blue              NA male  mascu~
3 Lama Su      229    88 none       grey       black             NA male  mascu~
4 Chewbacca    228   112 brown      unknown    blue             200 male  mascu~
5 Roos Tar~    224    82 none       grey       orange            NA male  mascu~
```

```
6 Grievous     216   159 none       brown, wh~ green, y~         NA male  mascu~
# i 5 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

```
head(bmi_data)
```

```
# A tibble: 6 x 15
  name      height  mass hair_color skin_color eye_color birth_year sex   gender
  <chr>      <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
1 Luke Sky~    172    77 blond      fair       blue            19   male  mascu~
2 C-3PO        167    75 <NA>       gold       yellow         112   none  mascu~
3 R2-D2         96    32 <NA>       white, bl~ red             33   none  mascu~
4 Darth Va~    202   136 none       white      yellow          41.9 male  mascu~
5 Leia Org~    150    49 brown      light      brown           19   fema~ femin~
6 Owen Lars    178   120 brown, gr~ light      blue            52   male  mascu~
# i 6 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>, bmi <dbl>
```

```
head(species_summary)
```

```
# A tibble: 6 x 4
  species  count avg_height avg_mass
  <chr>    <int>      <dbl>    <dbl>
1 Human       35       178      81.3
2 Droid        6       131.     69.8
3 <NA>         4       175      81
4 Gungan       3       209.     74
5 Kaminoan     2       221      88
6 Mirialan     2       168      53.1
```

2.5.1 Using Our Downloaded Epidemiology Dataset

Now let's explore our epidemiology dataset (storm data) using dplyr:

```
# Let's use our epidemiology dataset
disease_data <- read_csv("../data/epidemiology/disease_data.csv")
head(disease_data)
```

```
# A tibble: 6 x 13
  name   year month   day  hour   lat  long status       category  wind pressure
  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>           <dbl> <dbl>    <dbl>
1 Amy    1975     6    27     0  27.5 -79   tropical de~       NA    25     1013
2 Amy    1975     6    27     6  28.5 -79   tropical de~       NA    25     1013
3 Amy    1975     6    27    12  29.5 -79   tropical de~       NA    25     1013
4 Amy    1975     6    27    18  30.5 -79   tropical de~       NA    25     1013
5 Amy    1975     6    28     0  31.5 -78.8 tropical de~       NA    25     1012
6 Amy    1975     6    28     6  32.4 -78.7 tropical de~       NA    25     1012
# i 2 more variables: tropicalstorm_force_diameter <dbl>,
#   hurricane_force_diameter <dbl>
```

```
# Filter rows
storm_data <- disease_data %>%
  filter(year >= 2000)

# Select columns
storm_subset <- disease_data %>%
  select(name, year, month, wind, pressure)
```

```
# Create new variables
storm_analysis <- disease_data %>%
  mutate(wind_kph = wind * 1.852) # Convert knots to kph

# Summarize data
storm_summary <- disease_data %>%
  group_by(name) %>%
  summarize(
    avg_wind = mean(wind, na.rm = TRUE),
    max_wind = max(wind, na.rm = TRUE),
    count = n()
  ) %>%
  arrange(desc(max_wind))

head(storm_summary)
```

```
# A tibble: 6 x 4
  name    avg_wind max_wind count
  <chr>      <dbl>    <dbl> <int>
1 Allen      100.       165    44
2 Dorian      62.0      160   121
3 Gilbert     77.3      160    49
4 Wilma       88.5      160    48
5 Irma        95.3      155    72
6 Mitch       63.1      155    78
```

2.6 Working with Data in Jamovi

Jamovi provides a user-friendly interface for data manipulation:

1. **Data View**: The main spreadsheet-like interface for viewing and editing data
2. **Variables View**: For defining variable properties like name, type, and measurement level
3. **Compute**: For creating new variables based on formulas
4. **Transform**: For recoding variables or creating categorical variables from continuous ones
5. **Filters**: For selecting subsets of data based on conditions

2.7 Exploratory Data Analysis

Before diving into formal statistical tests, it's important to explore your data:

```
# Load a dataset
data(mtcars)

# Basic summary statistics
summary(mtcars)
```

```
      mpg             cyl             disp             hp       
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
```

```
      drat             wt             qsec             vs
 Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000
 1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000
 Median :3.695   Median :3.325   Median :17.71   Median :0.0000
 Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375
 3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000
 Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000
       am              gear            carb
 Min.   :0.0000   Min.   :3.000   Min.   :1.000
 1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
 Median :0.0000   Median :4.000   Median :2.000
 Mean   :0.4062   Mean   :3.688   Mean   :2.812
 3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000
 Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

```
# Structure of the data
str(mtcars)
```

```
'data.frame':   32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

```
# First few rows
head(mtcars)
```

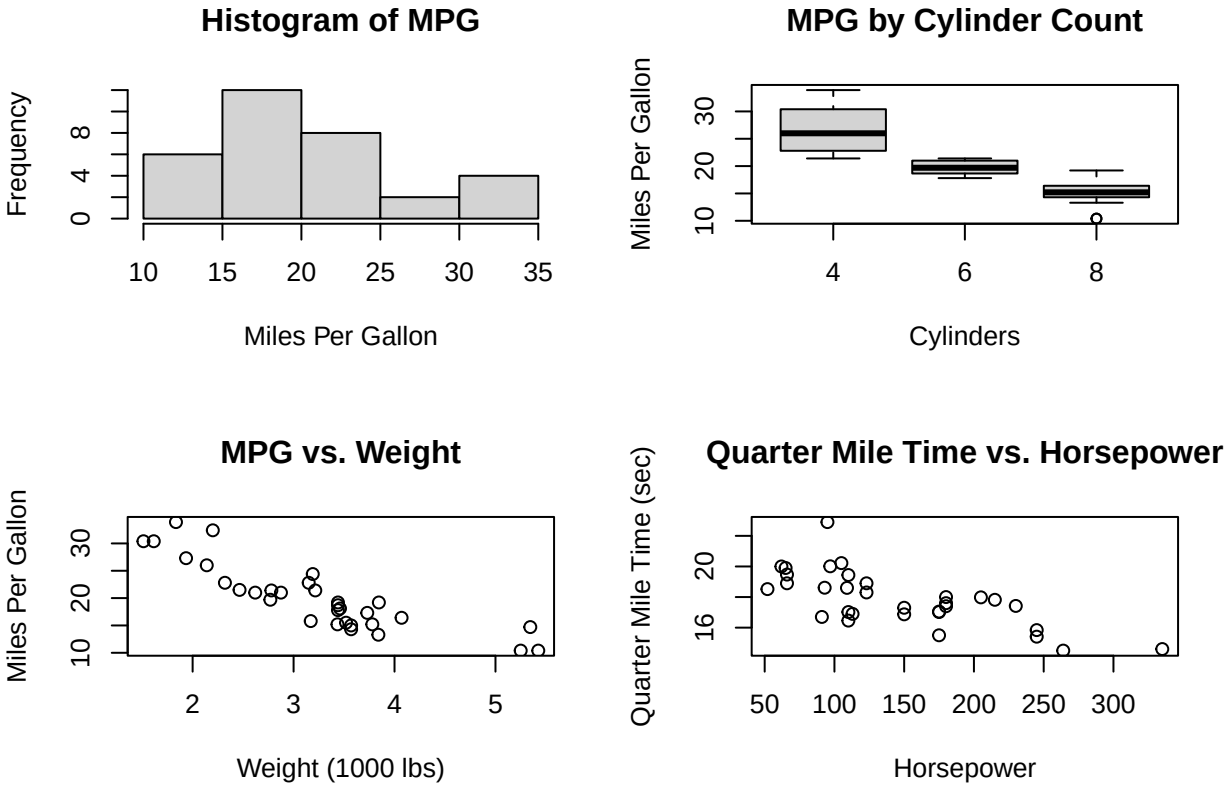
```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
# Correlation matrix
cor_matrix <- cor(mtcars)
round(cor_matrix, 2)
```

```
       mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
mpg   1.00 -0.85 -0.85 -0.78  0.68 -0.87  0.42  0.66  0.60  0.48 -0.55
cyl  -0.85  1.00  0.90  0.83 -0.70  0.78 -0.59 -0.81 -0.52 -0.49  0.53
disp -0.85  0.90  1.00  0.79 -0.71  0.89 -0.43 -0.71 -0.59 -0.56  0.39
hp   -0.78  0.83  0.79  1.00 -0.45  0.66 -0.71 -0.72 -0.24 -0.13  0.75
drat  0.68 -0.70 -0.71 -0.45  1.00 -0.71  0.09  0.44  0.71  0.70 -0.09
wt   -0.87  0.78  0.89  0.66 -0.71  1.00 -0.17 -0.55 -0.69 -0.58  0.43
qsec  0.42 -0.59 -0.43 -0.71  0.09 -0.17  1.00  0.74 -0.23 -0.21 -0.66
vs    0.66 -0.81 -0.71 -0.72  0.44 -0.55  0.74  1.00  0.17  0.21 -0.57
am    0.60 -0.52 -0.59 -0.24  0.71 -0.69 -0.23  0.17  1.00  0.79  0.06
gear  0.48 -0.49 -0.56 -0.13  0.70 -0.58 -0.21  0.21  0.79  1.00  0.27
```

```
carb -0.55  0.53  0.39  0.75 -0.09  0.43 -0.66 -0.57  0.06  0.27  1.00
```

```
# Basic visualizations
par(mfrow = c(2, 2))
hist(mtcars$mpg, main = "Histogram of MPG", xlab = "Miles Per Gallon")
boxplot(mpg ~ cyl, data = mtcars, main = "MPG by Cylinder Count",
        xlab = "Cylinders", ylab = "Miles Per Gallon")
plot(mtcars$wt, mtcars$mpg, main = "MPG vs. Weight",
     xlab = "Weight (1000 lbs)", ylab = "Miles Per Gallon")
plot(mtcars$hp, mtcars$qsec, main = "Quarter Mile Time vs. Horsepower",
     xlab = "Horsepower", ylab = "Quarter Mile Time (sec)")
```



2.8 Summary

In this chapter, we've covered the basics of working with data in R and Jamovi:

- Understanding different data types and structures
- Importing data from various file formats
- Cleaning data by handling missing values
- Transforming data to meet analysis requirements
- Creating new variables
- Using dplyr for powerful data manipulation
- Conducting initial exploratory data analysis

These skills form the foundation for all the analyses we'll perform in the subsequent chapters. By mastering these basics, you'll be well-prepared to tackle more complex analytical challenges in ecological and forestry research.

2.9 Exercises

1. Import a CSV file of your choice into R and examine its structure.
2. Create a new variable in the dataset based on existing variables.
3. Filter the dataset to include only observations that meet specific criteria.
4. Calculate summary statistics for different groups within your dataset.
5. Create basic visualizations to explore relationships between variables.
6. Repeat exercises 1-5 using Jamovi instead of R.

Part II

Data Analysis Fundamentals

Chapter 3

Exploratory Data Analysis

3.1 Introduction

Exploratory Data Analysis (EDA) is a critical first step in any data analysis project. In this chapter, you'll learn how to systematically explore your data to understand its structure, identify patterns, detect anomalies, and generate hypotheses for further investigation.

3.2 The Purpose of Exploratory Data Analysis

Exploratory Data Analysis serves several important purposes in natural sciences research:

1. **Understanding Data Structure**: Gain insights into the basic properties of your dataset
2. **Checking Data Quality**: Identify missing values, outliers, and potential errors
3. **Discovering Patterns**: Detect relationships, trends, and distributions
4. **Generating Hypotheses**: Develop questions and hypotheses for formal testing
5. **Informing Analysis Choices**: Guide decisions about appropriate statistical methods

3.3 Summarizing Data

3.3.1 Descriptive Statistics

Descriptive statistics provide a concise summary of your data's central tendency, dispersion, and shape:

```
# Load necessary libraries
library(tidyverse)

# Load the crop yield dataset
crop_yields <- read_csv("../data/agriculture/crop_yields.csv")

# View the first few rows
head(crop_yields)
```

```
# A tibble: 6 x 14
  Entity      Code   Year `Wheat (tonnes per hectare)` Rice (tonnes per hectar~1
  <chr>       <chr> <dbl>                        <dbl>                      <dbl>
1 Afghanistan AFG    1961                         1.02                       1.52
2 Afghanistan AFG    1962                         0.974                      1.52
3 Afghanistan AFG    1963                         0.832                      1.52
4 Afghanistan AFG    1964                         0.951                      1.73
```

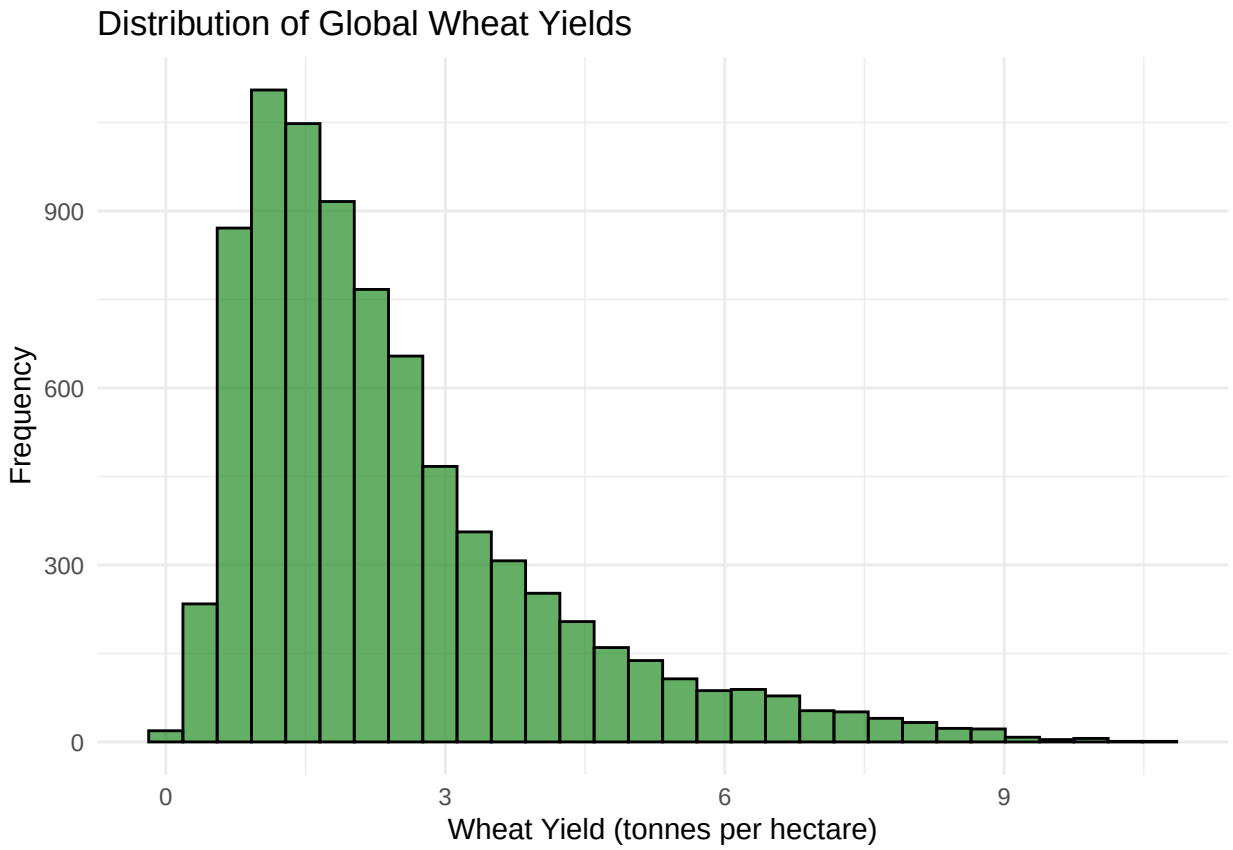
```
5 Afghanistan AFG    1965                         0.972                         1.73
6 Afghanistan AFG    1966                         0.867                         1.52
# i abbreviated name: 1: `Rice (tonnes per hectare)`
# i 9 more variables: `Maize (tonnes per hectare)` <dbl>,
#   `Soybeans (tonnes per hectare)` <dbl>,
#   `Potatoes (tonnes per hectare)` <dbl>, `Beans (tonnes per hectare)` <dbl>,
#   `Peas (tonnes per hectare)` <dbl>, `Cassava (tonnes per hectare)` <dbl>,
#   `Barley (tonnes per hectare)` <dbl>,
#   `Cocoa beans (tonnes per hectare)` <dbl>, ...
```

```r
# Get summary statistics for wheat yields
wheat_summary <- crop_yields %>%
  filter(!is.na(`Wheat (tonnes per hectare)`)) %>%
  summarize(
    Mean = mean(`Wheat (tonnes per hectare)`, na.rm = TRUE),
    Median = median(`Wheat (tonnes per hectare)`, na.rm = TRUE),
    StdDev = sd(`Wheat (tonnes per hectare)`, na.rm = TRUE),
    Min = min(`Wheat (tonnes per hectare)`, na.rm = TRUE),
    Max = max(`Wheat (tonnes per hectare)`, na.rm = TRUE),
    Q1 = quantile(`Wheat (tonnes per hectare)`, 0.25, na.rm = TRUE),
    Q3 = quantile(`Wheat (tonnes per hectare)`, 0.75, na.rm = TRUE)
  )

# Display the summary statistics
knitr::kable(wheat_summary, caption = "Summary Statistics for Global Wheat Yields")
```

Table 3.1: Summary Statistics for Global Wheat Yields

Mean	Median	StdDev	Min	Max	Q1	Q3
2.434914	1.99	1.687949	0	10.6677	1.228	3.1245

```r
# Visualize the distribution of wheat yields
ggplot(crop_yields, aes(x = `Wheat (tonnes per hectare)`)) +
  geom_histogram(bins = 30, fill = "forestgreen", color = "black", alpha = 0.7) +
  labs(title = "Distribution of Global Wheat Yields",
       x = "Wheat Yield (tonnes per hectare)",
       y = "Frequency") +
  theme_minimal()
```




```
# Identify top wheat-producing countries (by average yield)
top_wheat_countries <- crop_yields %>%
  filter(!is.na(`Wheat (tonnes per hectare)`)) %>%
  group_by(Entity) %>%
  summarize(Avg_Yield = mean(`Wheat (tonnes per hectare)`, na.rm = TRUE)) %>%
  arrange(desc(Avg_Yield)) %>%
  head(10)

# Display the top countries
knitr::kable(top_wheat_countries, caption = "Top 10 Countries by Average Wheat Yield")
```

Table 3.2: Top 10 Countries by Average Wheat Yield

Entity	Avg_Yield
Belgium	8.544200
Netherlands	7.030172
Ireland	6.829840
United Kingdom	6.366400
Denmark	6.175285
Luxembourg	5.977411
Germany	5.893978
Europe, Western	5.723267
France	5.645341
Northern Europe	5.589988

3.3.2 Frequency Tables

Frequency tables are useful for understanding the distribution of categorical variables:

```
# Let's create a categorical variable based on wheat yield levels
crop_yields_with_categories <- crop_yields %>%
  filter(!is.na(`Wheat (tonnes per hectare)`)) %>%
  mutate(yield_category = case_when(
    `Wheat (tonnes per hectare)` < 2 ~ "Low",
    `Wheat (tonnes per hectare)` >= 2 & `Wheat (tonnes per hectare)` < 4 ~ "Medium",
    `Wheat (tonnes per hectare)` >= 4 ~ "High"
  ))

# Frequency table for yield categories
table(crop_yields_with_categories$yield_category)
```

```
  High    Low Medium
  1279   4081   2741
```

```
# Proportions
prop.table(table(crop_yields_with_categories$yield_category))
```

```
     High       Low    Medium
0.1578817 0.5037650 0.3383533
```

```
# Create a decade variable for temporal analysis
crop_yields_with_categories <- crop_yields_with_categories %>%
  mutate(decade = floor(Year / 10) * 10)

# Two-way frequency table: yield category by decade
yield_decade_table <- table(crop_yields_with_categories$yield_category,
                            crop_yields_with_categories$decade)
yield_decade_table
```

```
        1960 1970 1980 1990 2000 2010
  High    34  102  200  261  326  356
  Low    838  833  760  681  563  406
  Medium 239  335  344  550  656  617
```

```
# Convert to proportions (by row)
prop.table(yield_decade_table, margin = 1)
```

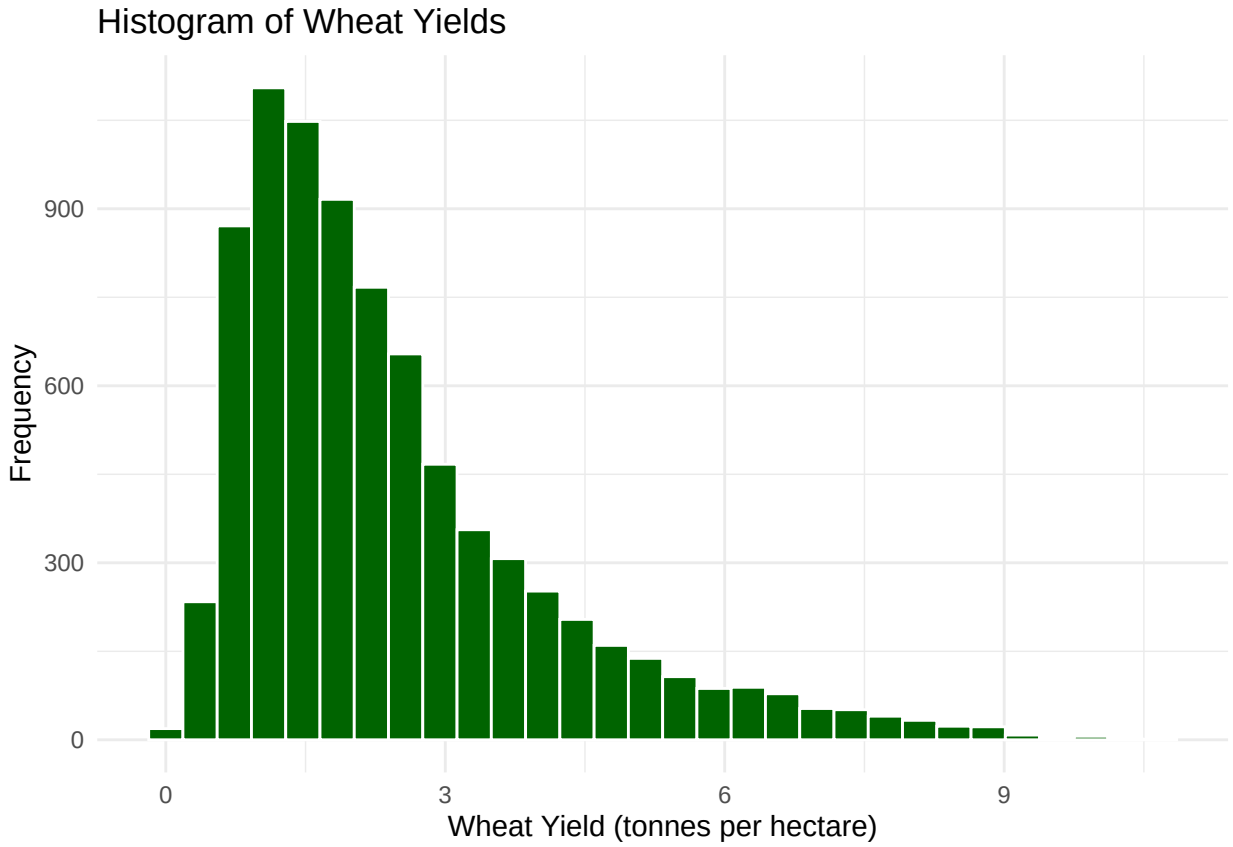
```
              1960       1970       1980       1990       2000       2010
  High   0.02658327 0.07974980 0.15637217 0.20406568 0.25488663 0.27834246
  Low    0.20534183 0.20411664 0.18622887 0.16687086 0.13795638 0.09948542
  Medium 0.08719445 0.12221817 0.12550164 0.20065669 0.23932871 0.22510033
```

3.4 Visualizing Distributions

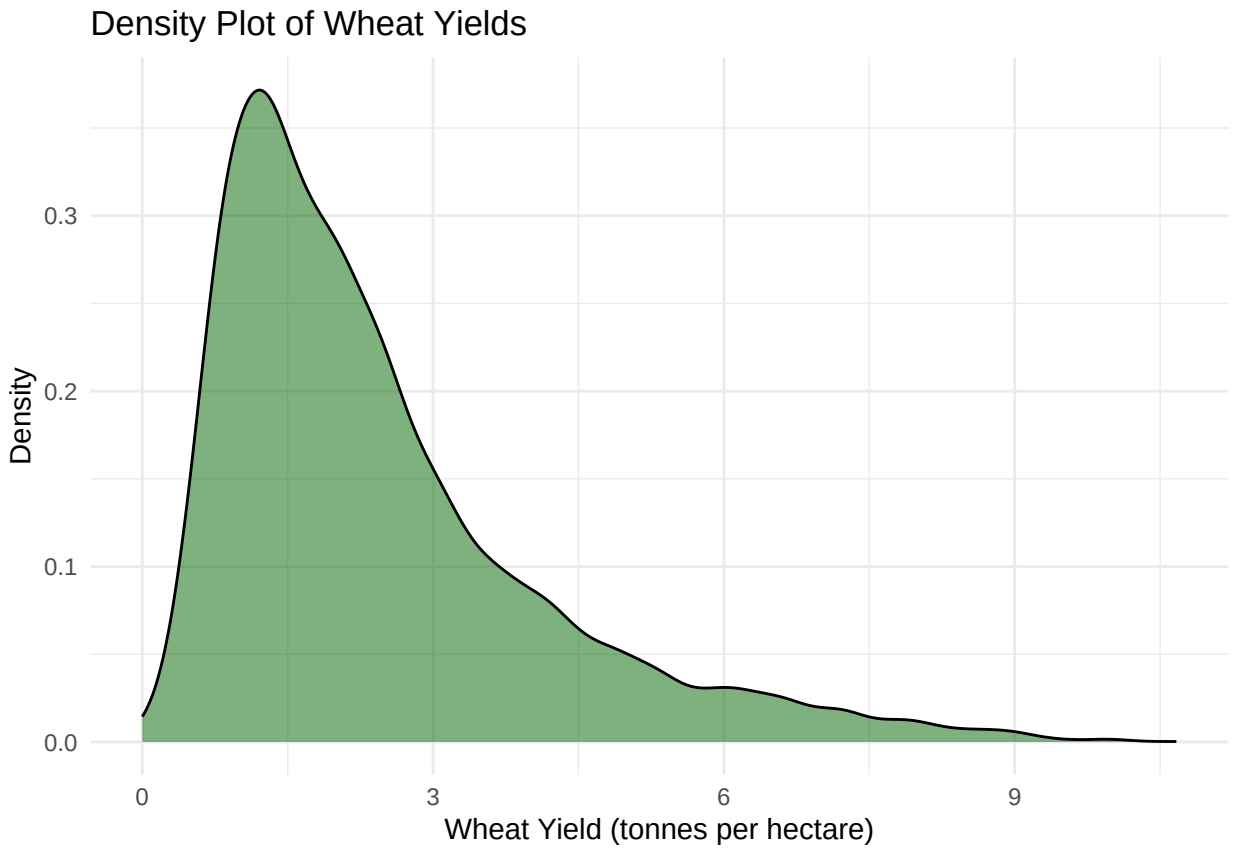
3.4.1 Histograms and Density Plots

Histograms and density plots help visualize the distribution of continuous variables:

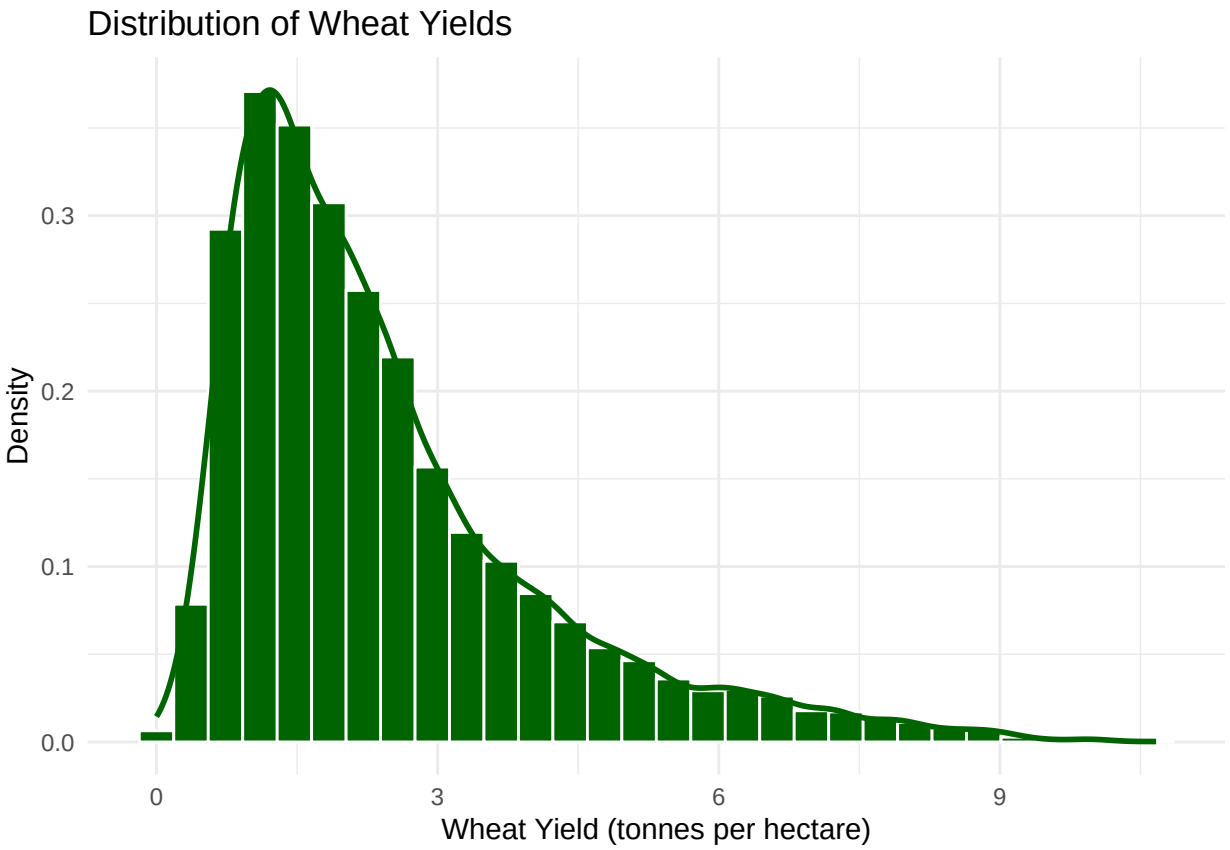
```
# Histogram of wheat yields
ggplot(crop_yields, aes(x = `Wheat (tonnes per hectare)`)) +
  geom_histogram(bins = 30, fill = "darkgreen", color = "white", na.rm = TRUE) +
  labs(title = "Histogram of Wheat Yields",
       x = "Wheat Yield (tonnes per hectare)",
       y = "Frequency") +
  theme_minimal()
```




```
# Density plot
ggplot(crop_yields, aes(x = `Wheat (tonnes per hectare)`)) +
  geom_density(fill = "darkgreen", alpha = 0.5, na.rm = TRUE) +
  labs(title = "Density Plot of Wheat Yields",
       x = "Wheat Yield (tonnes per hectare)",
       y = "Density") +
  theme_minimal()
```



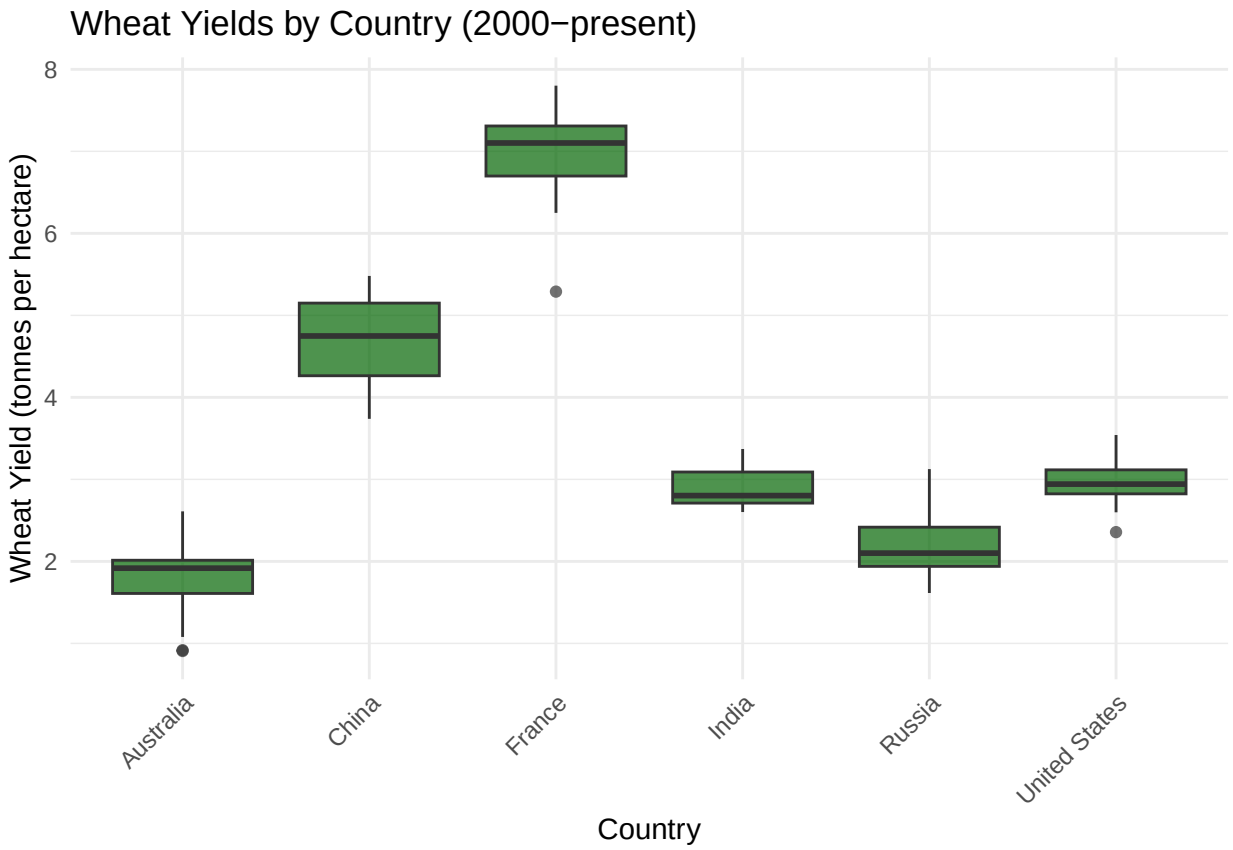

```
# Histogram with density overlay
ggplot(crop_yields, aes(x = `Wheat (tonnes per hectare)`)) +
  geom_histogram(aes(y = ..density..), bins = 30, fill = "darkgreen", color = "white", na.rm = TRUE) +
  geom_density(color = "darkgreen", linewidth = 1, na.rm = TRUE) +
  labs(title = "Distribution of Wheat Yields",
       x = "Wheat Yield (tonnes per hectare)",
       y = "Density") +
  theme_minimal()
```



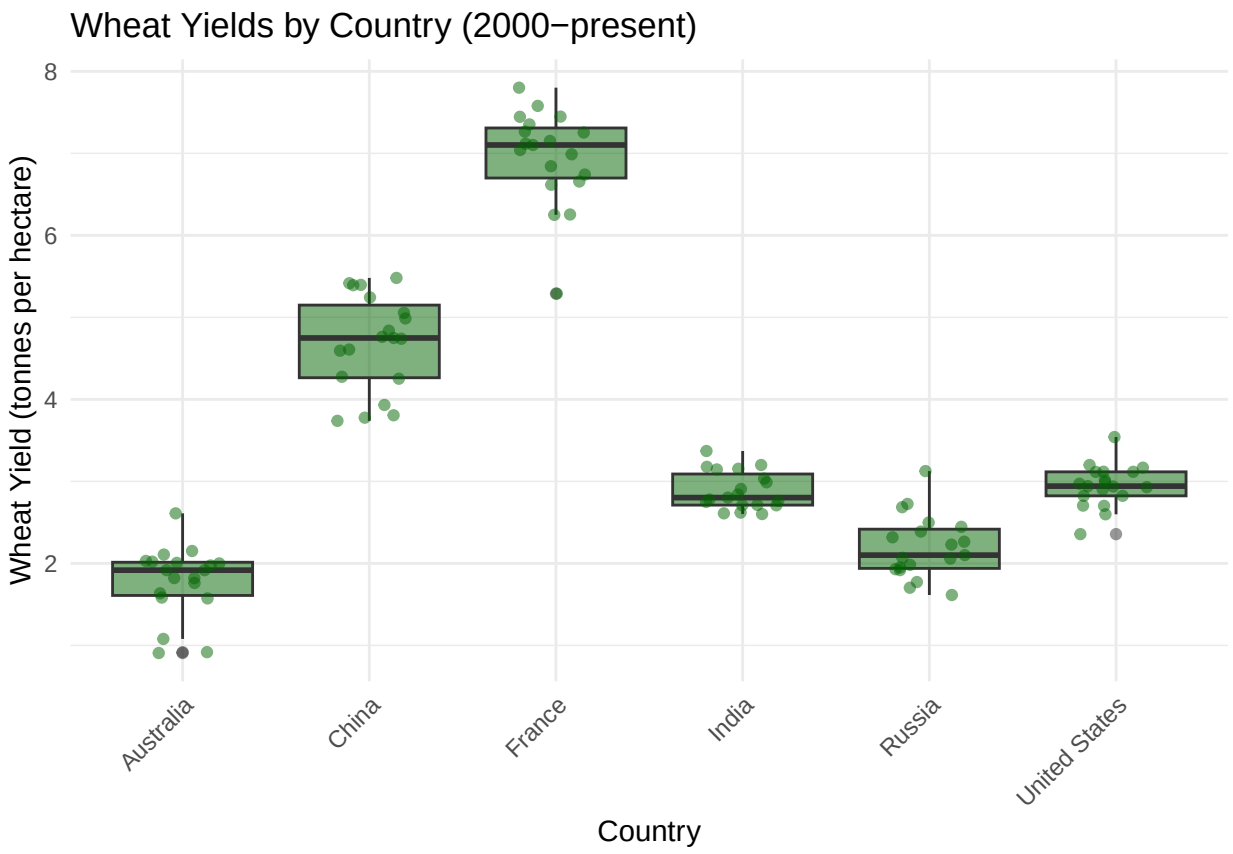
3.4.2 Box Plots

Box plots are excellent for comparing distributions across groups:

```
# Select a few major countries for comparison
major_wheat_producers <- c("United States", "China", "India", "Russia", "France", "Australia")

# Filter data for these countries and recent years
recent_wheat_data <- crop_yields %>%
  filter(Entity %in% major_wheat_producers,
         Year >= 2000,
         !is.na(`Wheat (tonnes per hectare)`))

# Box plot of wheat yields by country
ggplot(recent_wheat_data, aes(x = Entity, y = `Wheat (tonnes per hectare)`)) +
  geom_boxplot(fill = "darkgreen", alpha = 0.7) +
  labs(title = "Wheat Yields by Country (2000-present)",
       x = "Country",
       y = "Wheat Yield (tonnes per hectare)") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



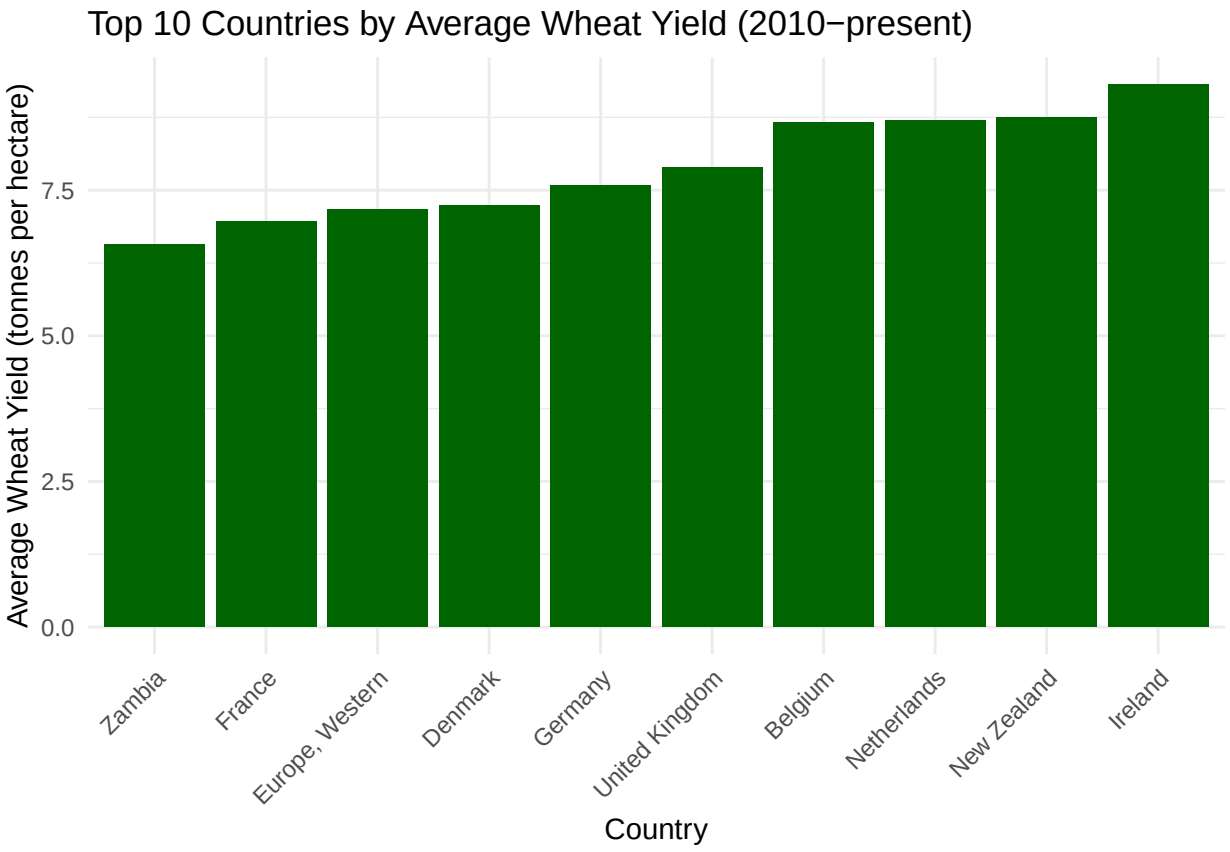

```
# Enhanced box plot with jittered points
ggplot(recent_wheat_data, aes(x = Entity, y = `Wheat (tonnes per hectare)`)) +
  geom_boxplot(fill = "darkgreen", alpha = 0.5) +
  geom_jitter(width = 0.2, alpha = 0.5, color = "darkgreen") +
  labs(title = "Wheat Yields by Country (2000-present)",
       x = "Country",
       y = "Wheat Yield (tonnes per hectare)") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



3.4.3 Bar Charts

Bar charts are useful for visualizing categorical data:

```
# Calculate average wheat yield by country for the last decade
recent_avg_yields <- crop_yields %>%
  filter(Year >= 2010, !is.na(`Wheat (tonnes per hectare)`)) %>%
  group_by(Entity) %>%
  summarize(avg_wheat_yield = mean(`Wheat (tonnes per hectare)`, na.rm = TRUE)) %>%
  arrange(desc(avg_wheat_yield)) %>%
  head(10)  # Top 10 countries

# Bar chart of average wheat yields
ggplot(recent_avg_yields, aes(x = reorder(Entity, avg_wheat_yield), y = avg_wheat_yield)) +
  geom_bar(stat = "identity", fill = "darkgreen") +
  labs(title = "Top 10 Countries by Average Wheat Yield (2010-present)",
       x = "Country",
       y = "Average Wheat Yield (tonnes per hectare)") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

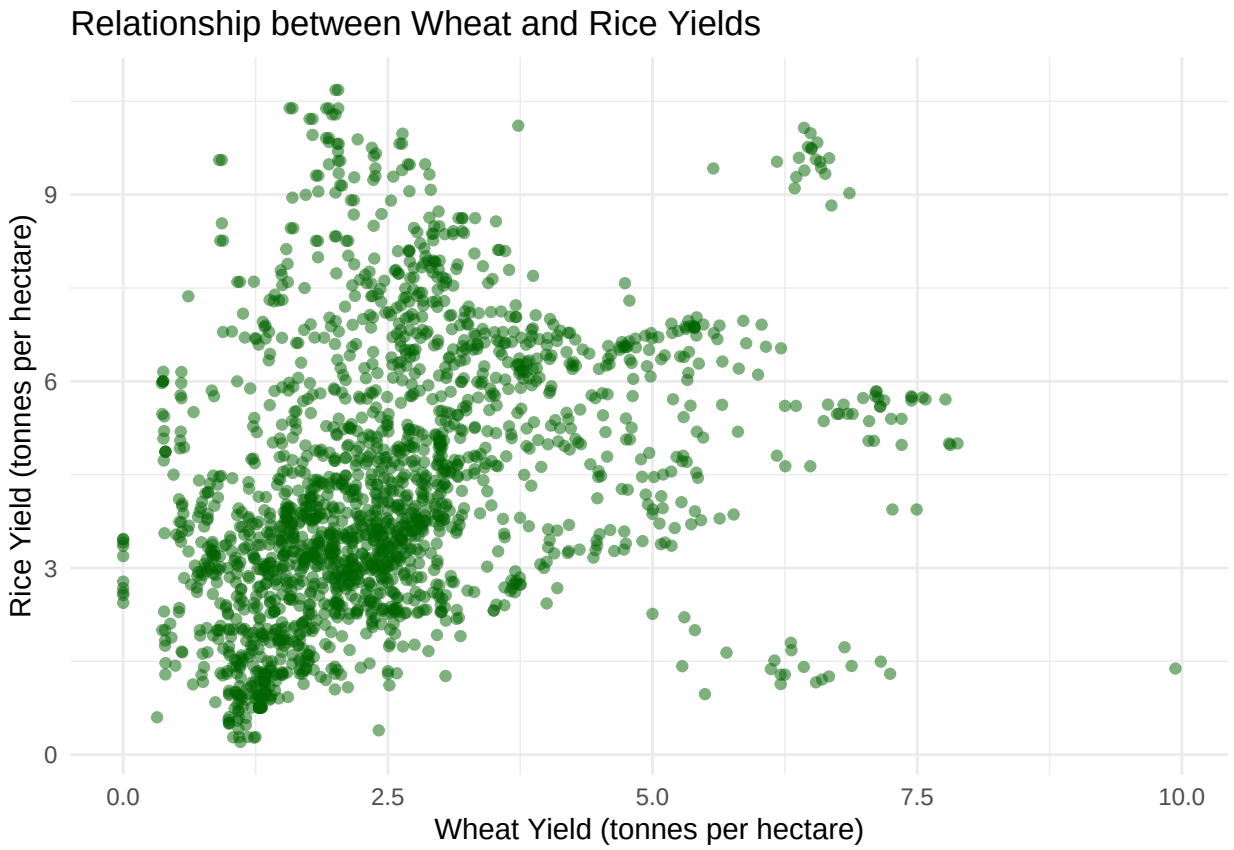


3.5 Exploring Relationships

3.5.1 Scatter Plots

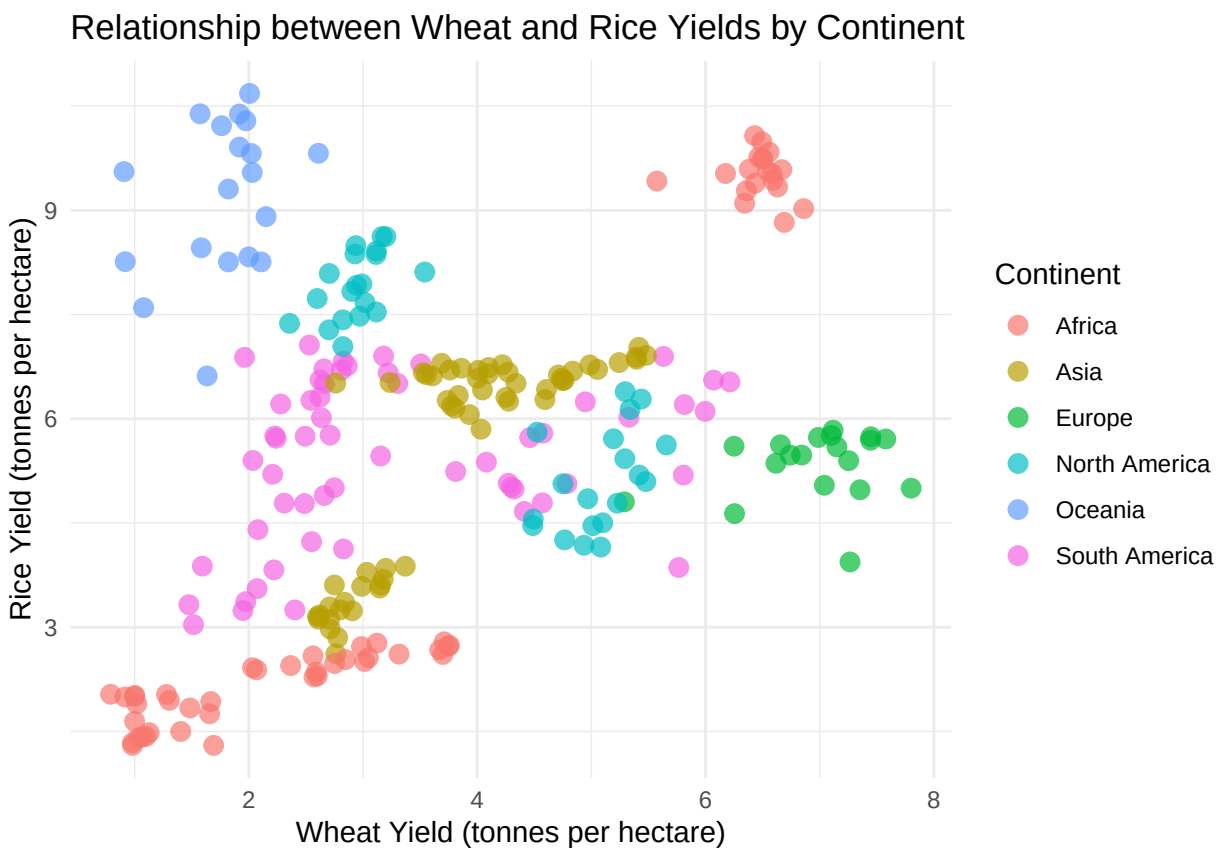
Scatter plots help visualize relationships between two continuous variables:

```
# Let's compare wheat and rice yields
crop_yields_filtered <- crop_yields %>%
  filter(!is.na(`Wheat (tonnes per hectare)`), !is.na(`Rice (tonnes per hectare)`)) %>%
  filter(Year >= 2000)

# Basic scatter plot
ggplot(crop_yields_filtered, aes(x = `Wheat (tonnes per hectare)`, y = `Rice (tonnes per hectare)`)) +
  geom_point(alpha = 0.5, color = "darkgreen") +
  labs(title = "Relationship between Wheat and Rice Yields",
       x = "Wheat Yield (tonnes per hectare)",
       y = "Rice Yield (tonnes per hectare)") +
  theme_minimal()
```




```
# Scatter plot with color by continent (we'll need to add continent information)
# For demonstration, let's create a simple mapping for a few countries
continent_mapping <- tibble(
  Entity = c("United States", "Canada", "Mexico",
             "China", "India", "Japan",
             "Germany", "France", "United Kingdom",
             "Brazil", "Argentina", "Chile",
             "Egypt", "Nigeria", "South Africa",
             "Australia", "New Zealand"),
  Continent = c(rep("North America", 3),
                rep("Asia", 3),
                rep("Europe", 3),
                rep("South America", 3),
                rep("Africa", 3),
                rep("Oceania", 2))
)

# Join with our dataset
crop_yields_with_continent <- crop_yields_filtered %>%
  inner_join(continent_mapping, by = "Entity")

# Scatter plot with color by continent
ggplot(crop_yields_with_continent, aes(x = `Wheat (tonnes per hectare)`, y = `Rice (tonnes per hectare)
  geom_point(size = 3, alpha = 0.7) +
  labs(title = "Relationship between Wheat and Rice Yields by Continent",
       x = "Wheat Yield (tonnes per hectare)",
       y = "Rice Yield (tonnes per hectare)") +
```

```
  theme_minimal()
```



3.5.2 Correlation Analysis

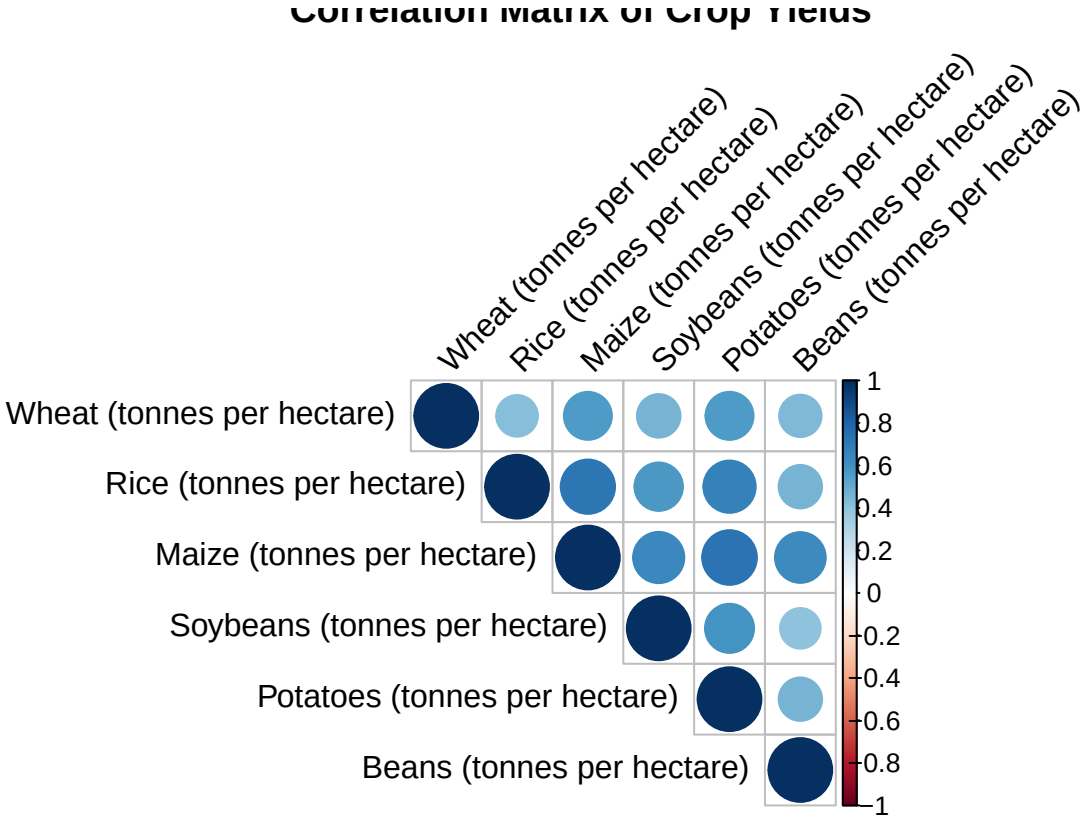
Correlation analysis quantifies the strength and direction of relationships between variables:

```
# Select numeric columns for correlation analysis
crop_numeric <- crop_yields %>%
  select(`Wheat (tonnes per hectare)`, `Rice (tonnes per hectare)`, `Maize (tonnes per hectare)`, `Soyb
  na.omit()

# Correlation matrix
cor_matrix <- cor(crop_numeric)
round(cor_matrix, 2)
```

```
                            Wheat (tonnes per hectare)
Wheat (tonnes per hectare)                        1.00
Rice (tonnes per hectare)                         0.43
Maize (tonnes per hectare)                        0.57
Soybeans (tonnes per hectare)                     0.47
Potatoes (tonnes per hectare)                     0.57
Beans (tonnes per hectare)                        0.44
                            Rice (tonnes per hectare)
Wheat (tonnes per hectare)                       0.43
Rice (tonnes per hectare)                        1.00
Maize (tonnes per hectare)                       0.73
Soybeans (tonnes per hectare)                    0.58
```

```
Potatoes (tonnes per hectare)                      0.67
Beans (tonnes per hectare)                         0.46
                              Maize (tonnes per hectare)
Wheat (tonnes per hectare)                          0.57
Rice (tonnes per hectare)                           0.73
Maize (tonnes per hectare)                          1.00
Soybeans (tonnes per hectare)                       0.65
Potatoes (tonnes per hectare)                       0.74
Beans (tonnes per hectare)                          0.63
                              Soybeans (tonnes per hectare)
Wheat (tonnes per hectare)                             0.47
Rice (tonnes per hectare)                              0.58
Maize (tonnes per hectare)                             0.65
Soybeans (tonnes per hectare)                          1.00
Potatoes (tonnes per hectare)                          0.59
Beans (tonnes per hectare)                             0.41
                              Potatoes (tonnes per hectare)
Wheat (tonnes per hectare)                             0.57
Rice (tonnes per hectare)                              0.67
Maize (tonnes per hectare)                             0.74
Soybeans (tonnes per hectare)                          0.59
Potatoes (tonnes per hectare)                          1.00
Beans (tonnes per hectare)                             0.46
                              Beans (tonnes per hectare)
Wheat (tonnes per hectare)                          0.44
Rice (tonnes per hectare)                           0.46
Maize (tonnes per hectare)                          0.63
Soybeans (tonnes per hectare)                       0.41
Potatoes (tonnes per hectare)                       0.46
Beans (tonnes per hectare)                          1.00
```

```r
# Visualize correlation matrix
library(corrplot)
corrplot(cor_matrix, method = "circle", type = "upper",
         tl.col = "black", tl.srt = 45,
         title = "Correlation Matrix of Crop Yields")
```

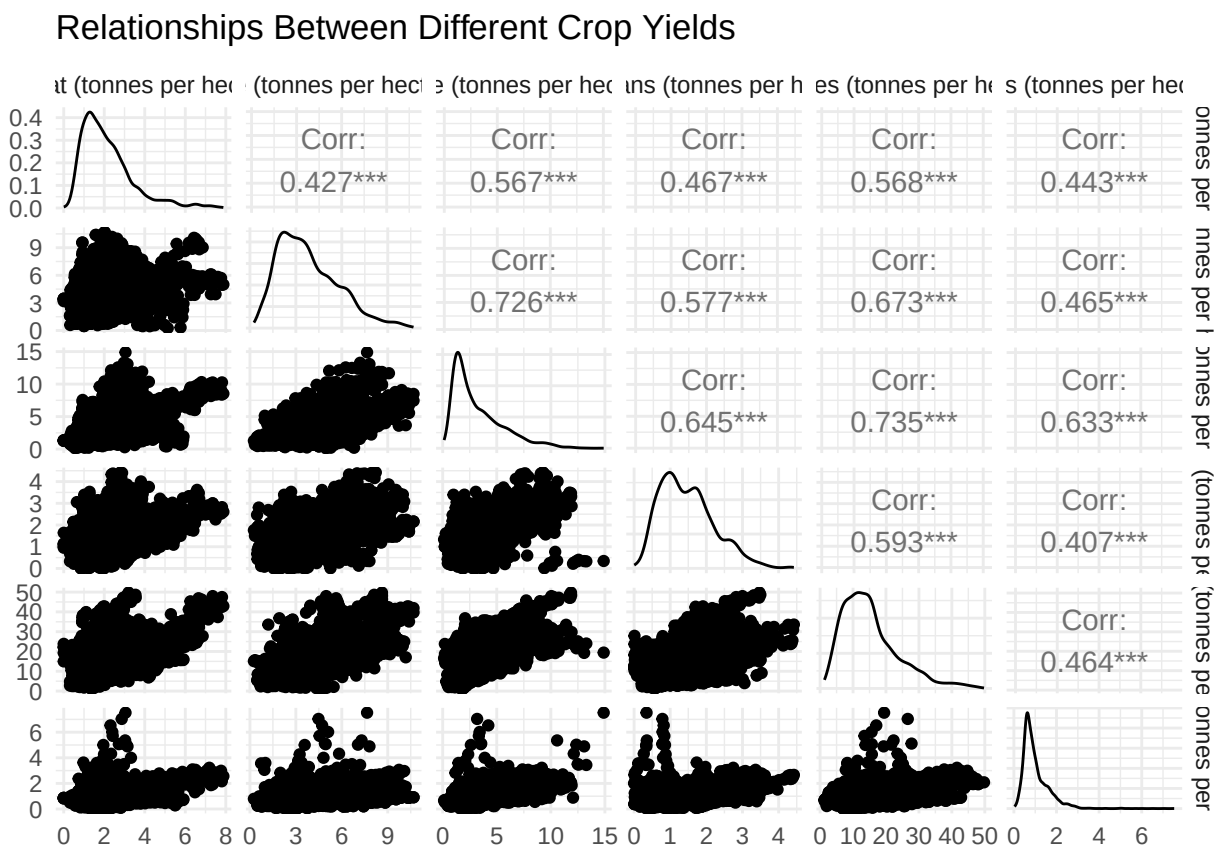


3.5.3 Pair Plots

Pair plots provide a comprehensive view of relationships between multiple variables:

```
# Basic pair plot
pairs(crop_numeric, pch = 19, col = "darkgreen")
```


```
# Enhanced pair plot with GGally
library(GGally)
ggpairs(crop_numeric) +
  theme_minimal() +
  labs(title = "Relationships Between Different Crop Yields")
```

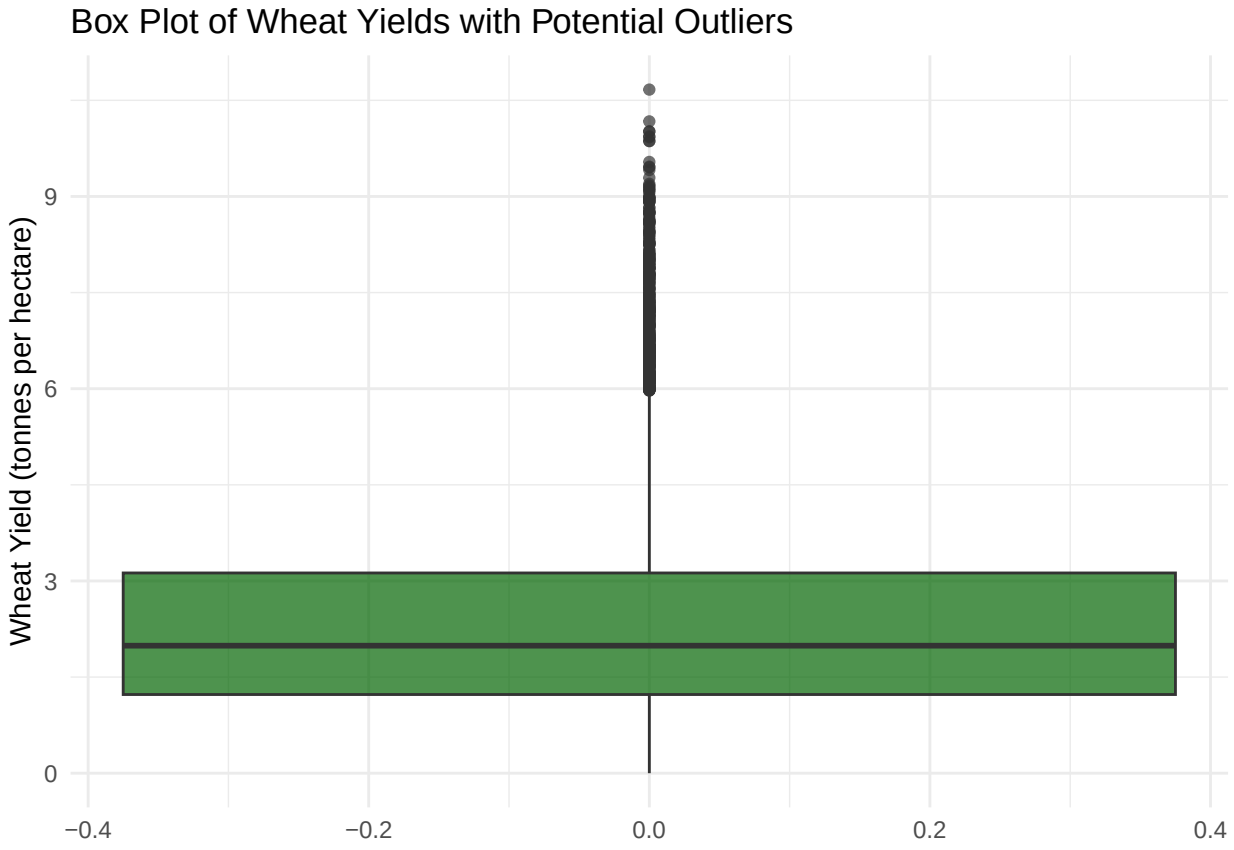


3.6 Identifying Outliers and Anomalies

3.6.1 Box Plots for Outlier Detection

Box plots can help identify potential outliers:

```
# Box plot to identify outliers in wheat yield
ggplot(crop_yields, aes(y = `Wheat (tonnes per hectare)`)) +
  geom_boxplot(fill = "darkgreen", alpha = 0.7, na.rm = TRUE) +
  labs(title = "Box Plot of Wheat Yields with Potential Outliers",
       y = "Wheat Yield (tonnes per hectare)") +
  theme_minimal()
```




```
# Identify potential outliers
wheat_outliers <- crop_yields %>%
  filter(!is.na(`Wheat (tonnes per hectare)`)) %>%
  mutate(
    q1 = quantile(`Wheat (tonnes per hectare)`, 0.25),
    q3 = quantile(`Wheat (tonnes per hectare)`, 0.75),
    iqr = q3 - q1,
    lower_bound = q1 - 1.5 * iqr,
    upper_bound = q3 + 1.5 * iqr,
    is_outlier = `Wheat (tonnes per hectare)` < lower_bound | `Wheat (tonnes per hectare)` > upper_boun
  ) %>%
  filter(is_outlier) %>%
  select(Entity, Year, `Wheat (tonnes per hectare)`)

# Display the outliers
head(wheat_outliers, 10)
```

```
# A tibble: 10 x 3
   Entity   Year `Wheat (tonnes per hectare)`
   <chr>   <dbl>                        <dbl>
 1 Austria  2016                         6.25
 2 Belgium  2000                         7.92
 3 Belgium  2001                         8.05
 4 Belgium  2002                         8.28
 5 Belgium  2003                         8.58
 6 Belgium  2004                         8.98
 7 Belgium  2005                         8.27
```

```
 8 Belgium  2006                          8.25
 9 Belgium  2007                          7.89
10 Belgium  2008                          8.76
```

3.6.2 Z-Scores for Outlier Detection

Z-scores can also help identify outliers:

```
# Calculate z-scores for wheat yields
wheat_z_scores <- crop_yields %>%
  filter(!is.na(`Wheat (tonnes per hectare)`)) %>%
  mutate(
    wheat_mean = mean(`Wheat (tonnes per hectare)`),
    wheat_sd = sd(`Wheat (tonnes per hectare)`),
    z_score = (`Wheat (tonnes per hectare)` - wheat_mean) / wheat_sd,
    is_extreme = abs(z_score) > 3
  )

# Display extreme values (z-score > 3 or < -3)
wheat_extremes <- wheat_z_scores %>%
  filter(is_extreme) %>%
  select(Entity, Year, `Wheat (tonnes per hectare)`, z_score) %>%
  arrange(desc(abs(z_score)))

head(wheat_extremes, 10)
```

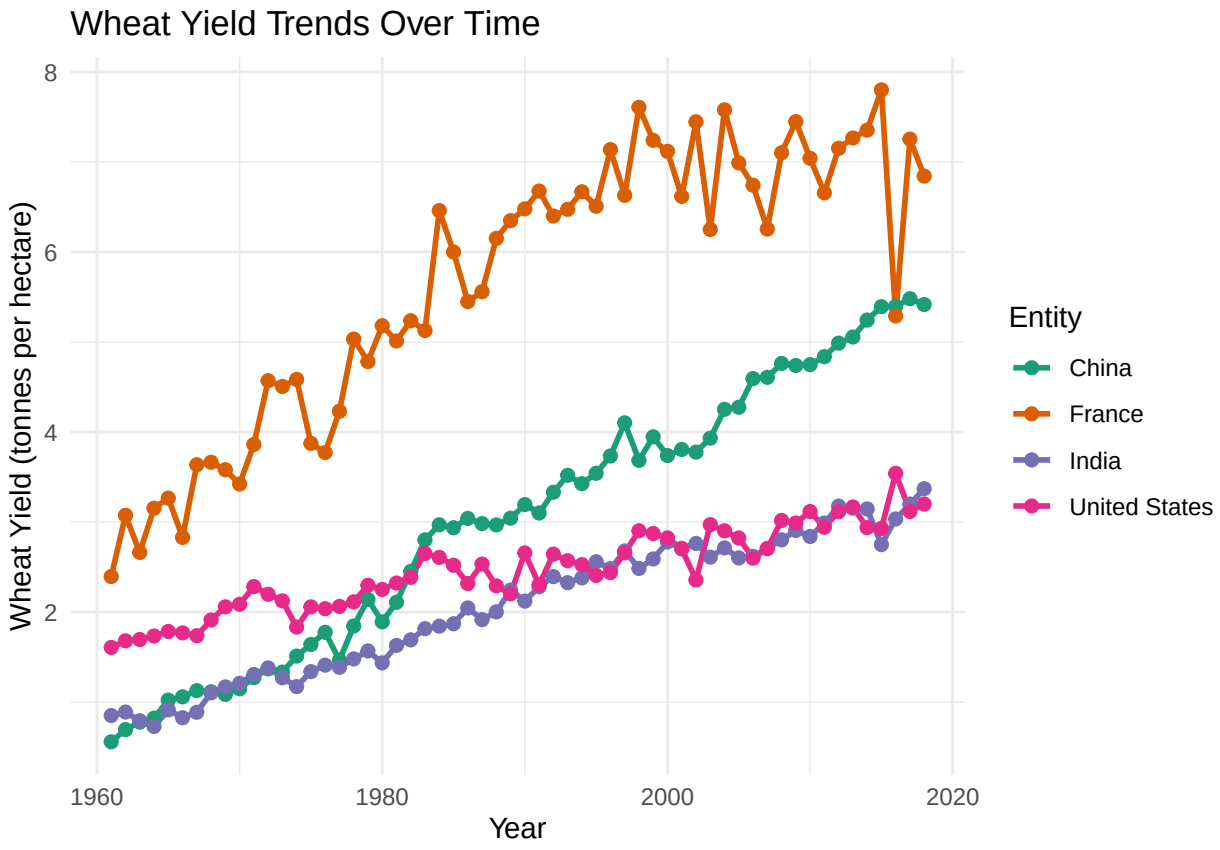
```
# A tibble: 10 x 4
   Entity       Year `Wheat (tonnes per hectare)` z_score
   <chr>       <dbl>                        <dbl>   <dbl>
 1 Ireland      2015                        10.7     4.88
 2 Ireland      2017                        10.2     4.58
 3 Belgium      2015                        10.0     4.49
 4 Ireland      2014                        10.0     4.49
 5 Zambia       2008                         9.94    4.45
 6 Ireland      2004                         9.92    4.44
 7 New Zealand  2017                         9.86    4.40
 8 Ireland      2011                         9.86    4.40
 9 Ireland      2016                         9.54    4.21
10 Belgium      2009                         9.47    4.16
```

3.7 Time Series Exploration

Agricultural data often contains important temporal patterns:

```
# Select a few countries for time series analysis
countries_for_ts <- c("United States", "China", "India", "France")

# Filter data for these countries
wheat_ts_data <- crop_yields %>%
  filter(Entity %in% countries_for_ts, !is.na(`Wheat (tonnes per hectare)`)) %>%
  filter(Year >= 1960)

# Time series plot
ggplot(wheat_ts_data, aes(x = Year, y = `Wheat (tonnes per hectare)`, color = Entity)) +
  geom_line(linewidth = 1) +
```

```
  geom_point(size = 2) +
  labs(title = "Wheat Yield Trends Over Time",
       x = "Year",
       y = "Wheat Yield (tonnes per hectare)") +
  theme_minimal() +
  scale_color_brewer(palette = "Dark2")
```



3.8 Missing Data Analysis

Understanding patterns of missing data is crucial:

```
# Check for missing values in each column
colSums(is.na(crop_yields))
```

```
                          Entity                             Code
                               0                             1919
                            Year       Wheat (tonnes per hectare)
                               0                             4974
       Rice (tonnes per hectare)       Maize (tonnes per hectare)
                            4604                             2301
   Soybeans (tonnes per hectare)    Potatoes (tonnes per hectare)
                            7114                             3059
      Beans (tonnes per hectare)        Peas (tonnes per hectare)
                            5066                             6840
    Cassava (tonnes per hectare)     Barley (tonnes per hectare)
                            5887                             6342
Cocoa beans (tonnes per hectare)    Bananas (tonnes per hectare)
```

8466 4166

```
# Visualize missing data patterns
if(requireNamespace("naniar", quietly = TRUE)) {
  library(naniar)

  # Create a visualization of missing data
  gg_miss_var(crop_yields)

  # Create a matrix showing missing data patterns
  vis_miss(crop_yields[, c("Entity", "Year", "Wheat (tonnes per hectare)", "Rice (tonnes per hectare)",
} else {
  message("The 'naniar' package is not installed. Install it with install.packages('naniar') to visuali

  # Alternative: simple summary of missing data
  missing_summary <- sapply(crop_yields, function(x) sum(is.na(x)))
  missing_df <- data.frame(
    Variable = names(missing_summary),
    Missing_Count = missing_summary,
    Missing_Percent = round(missing_summary / nrow(crop_yields) * 100, 2)
  )

  # Display the summary
  missing_df <- missing_df[order(-missing_df$Missing_Count), ]
  head(missing_df, 10)
}
```

```
                                                 Variable Missing_Count
Cocoa beans (tonnes per hectare) Cocoa beans (tonnes per hectare)          8466
Soybeans (tonnes per hectare)       Soybeans (tonnes per hectare)          7114
Peas (tonnes per hectare)              Peas (tonnes per hectare)          6840
Barley (tonnes per hectare)           Barley (tonnes per hectare)          6342
Cassava (tonnes per hectare)         Cassava (tonnes per hectare)          5887
Beans (tonnes per hectare)             Beans (tonnes per hectare)          5066
Wheat (tonnes per hectare)             Wheat (tonnes per hectare)          4974
Rice (tonnes per hectare)               Rice (tonnes per hectare)          4604
Bananas (tonnes per hectare)         Bananas (tonnes per hectare)          4166
Potatoes (tonnes per hectare)       Potatoes (tonnes per hectare)          3059
                                Missing_Percent
Cocoa beans (tonnes per hectare)           64.75
Soybeans (tonnes per hectare)              54.41
Peas (tonnes per hectare)                  52.31
Barley (tonnes per hectare)                48.50
Cassava (tonnes per hectare)               45.02
Beans (tonnes per hectare)                 38.75
Wheat (tonnes per hectare)                 38.04
Rice (tonnes per hectare)                  35.21
Bananas (tonnes per hectare)               31.86
Potatoes (tonnes per hectare)              23.40
```

3.9 Summary

This chapter has demonstrated various techniques for exploratory data analysis using a real agricultural dataset. We've covered:

1. Computing and interpreting descriptive statistics
2. Creating and analyzing frequency tables
3. Visualizing distributions with histograms, density plots, and box plots
4. Exploring relationships with scatter plots and correlation analysis
5. Identifying outliers and anomalies
6. Analyzing time series patterns
7. Examining missing data

These techniques provide a foundation for understanding your data before proceeding to more advanced analyses. By thoroughly exploring your data, you can make informed decisions about appropriate statistical methods and generate meaningful hypotheses for testing.

3.10 Exercises

1. Load the plant biodiversity dataset from `docs/data/ecology/biodiversity.csv` and perform a comprehensive exploratory analysis.
2. Create a histogram and density plot for another crop in the dataset. How does its distribution compare to wheat?
3. Investigate the relationship between potato yields and latitude (you'll need to find or create a dataset with latitude information).
4. Identify countries with the most significant improvement in crop yields over time.
5. Create a time series plot showing the ratio of wheat to rice yields over time for major producing countries.
6. Perform the same exploratory analyses in R for the spatial dataset in `docs/data/geography/spatial.csv`.

Chapter 4

Hypothesis Testing

4.1 Introduction

Hypothesis testing is a fundamental statistical approach used to make inferences about populations based on sample data. In ecological and forestry research, hypothesis testing helps researchers determine whether observed patterns or differences are statistically significant or merely due to random chance.

4.2 The Logic of Hypothesis Testing

4.2.1 Null and Alternative Hypotheses

The foundation of hypothesis testing involves two competing hypotheses:

1. **Null Hypothesis (H)**: This is the default position that assumes no effect, no difference, or no relationship exists. For example, "There is no difference in tree height between two forest types."
2. **Alternative Hypothesis (H or H)**: This is the hypothesis that the researcher typically wants to provide evidence for. For example, "There is a significant difference in tree height between two forest types."

4.2.2 Example in Ecological Research

Let's consider a specific example from forestry research:

- **Research Question**: Is there a difference in the average height of oak trees between Site A and Site B?
- **Null Hypothesis (H)**: There is no difference in the average height of oak trees between Site A and Site B.
- **Alternative Hypothesis (H)**: There is a significant difference in the average height of oak trees between Site A and Site B.

4.3 Understanding P-values and Significance Levels

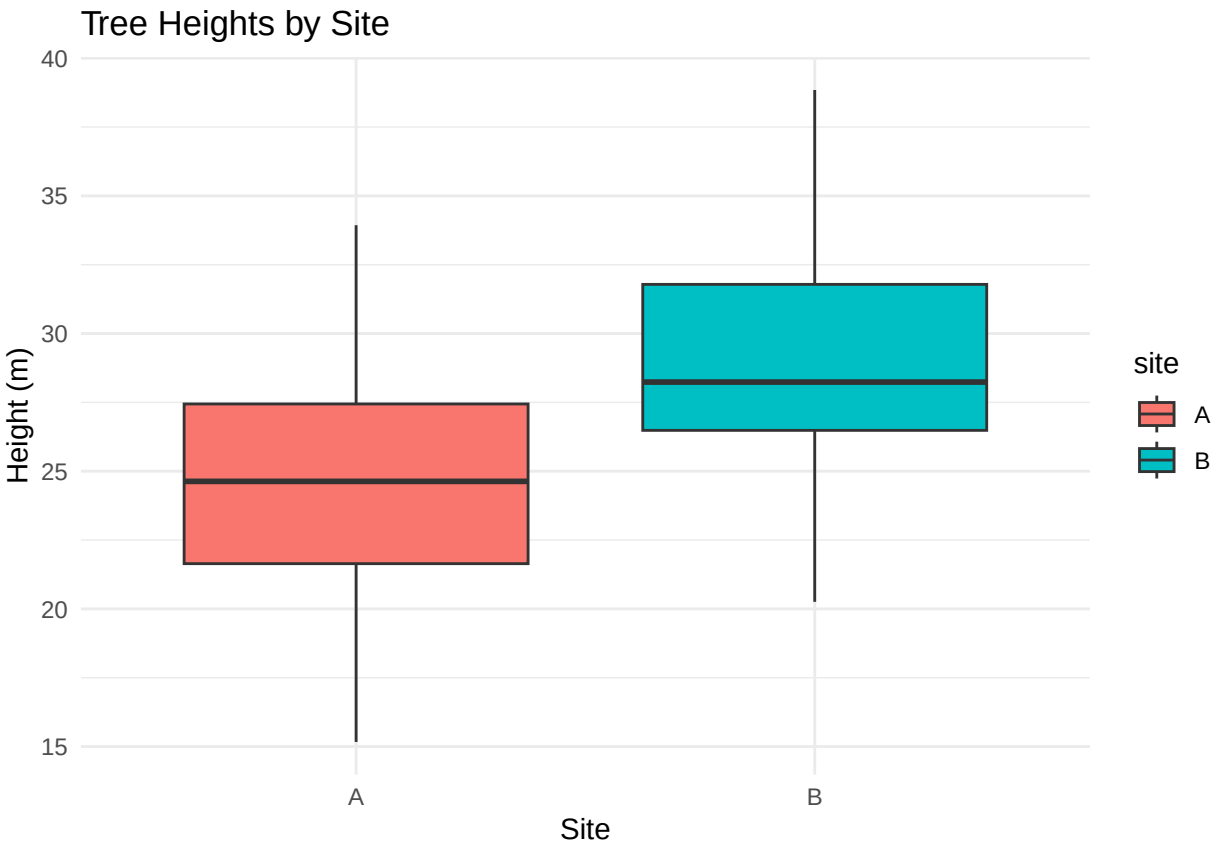
4.3.1 The P-value

The p-value is the probability of obtaining results at least as extreme as those observed, assuming the null hypothesis is true. A smaller p-value indicates stronger evidence against the null hypothesis.

4.3.2 Significance Level (Alpha)

The significance level, often denoted as (alpha), represents the threshold for statistical significance. In most research, it is set at 0.05 (5%). This value signifies the maximum acceptable probability of making a Type I error — wrongly rejecting the null hypothesis when it is true.

```
# Simulate tree height data for two sites
set.seed(123)
site_A <- rnorm(30, mean = 25, sd = 5)  # 30 trees with mean height 25m
site_B <- rnorm(30, mean = 28, sd = 5)  # 30 trees with mean height 28m

# Create a data frame
tree_data <- data.frame(
  height = c(site_A, site_B),
  site = factor(rep(c("A", "B"), each = 30))
)

# Visualize the data
library(ggplot2)
ggplot(tree_data, aes(x = site, y = height, fill = site)) +
  geom_boxplot() +
  labs(title = "Tree Heights by Site",
       x = "Site",
       y = "Height (m)") +
  theme_minimal()
```




```r
# Perform a t-test
t_test_result <- t.test(height ~ site, data = tree_data)
print(t_test_result)
```

```
    Welch Two Sample t-test

data:  height by site
t = -3.5092, df = 56.559, p-value = 0.0008892
alternative hypothesis: true difference in means between group A and group B is not equal to 0
95 percent confidence interval:
 -6.482713 -1.771708
sample estimates:
mean in group A mean in group B
       24.76448        28.89169
```

```r
# Interpret the result
alpha <- 0.05
if (t_test_result$p.value < alpha) {
  cat("With a p-value of", round(t_test_result$p.value, 4),
      "we reject the null hypothesis.\n",
      "There is a statistically significant difference in tree heights between sites.")
} else {
  cat("With a p-value of", round(t_test_result$p.value, 4),
      "we fail to reject the null hypothesis.\n",
      "There is not enough evidence to conclude a significant difference in tree heights.")
}
```

```
With a p-value of 9e-04 we reject the null hypothesis.
 There is a statistically significant difference in tree heights between sites.
```

4.5 Example: Using Marine Dataset for Two-Sample t-test

Let's apply the t-test to analyze real data. We'll use our marine dataset to compare fishing yields between different regions:

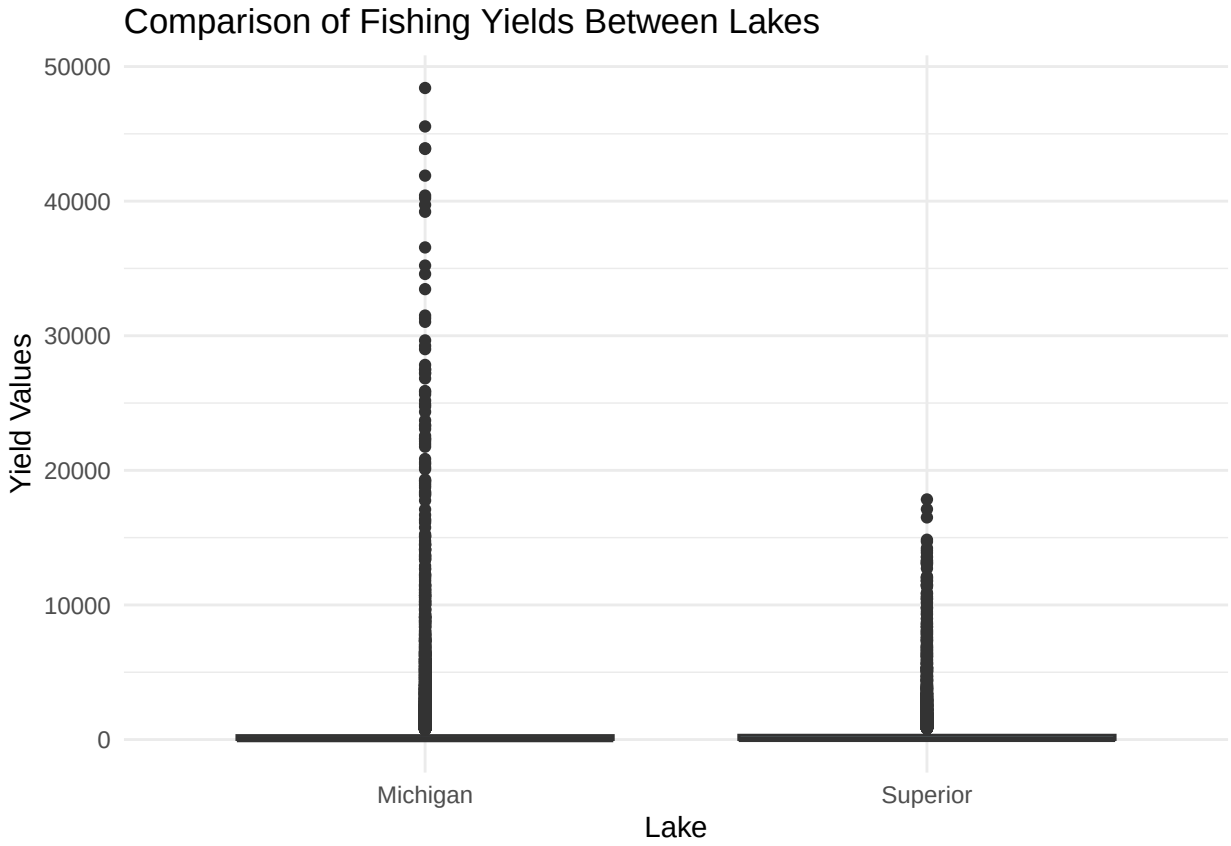
```
# Load necessary packages
library(tidyverse)

# Load the marine dataset
marine_data <- read_csv("../data/marine/ocean_data.csv")

# View the structure of the dataset
str(marine_data)
```

```
spc_tbl_ [65,706 x 7] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ year       : num [1:65706] 1991 1991 1991 1991 1991 ...
 $ lake       : chr [1:65706] "Erie" "Erie" "Erie" "Erie" ...
 $ species    : chr [1:65706] "American Eel" "American Eel" "American Eel" "American Eel" ...
 $ grand_total: num [1:65706] 1 1 1 1 1 1 0 0 0 0 ...
 $ comments   : chr [1:65706] NA NA NA NA ...
 $ region     : chr [1:65706] "Michigan (MI)" "New York (NY)" "Ohio (OH)" "Pennsylvania (PA)" ...
 $ values     : num [1:65706] 0 0 0 0 0 1 0 0 0 0 ...
 - attr(*, "spec")=
  .. cols(
  ..   year = col_double(),
  ..   lake = col_character(),
  ..   species = col_character(),
  ..   grand_total = col_double(),
  ..   comments = col_character(),
  ..   region = col_character(),
  ..   values = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

```
# Let's compare fishing yields between two lakes
if("lake" %in% colnames(marine_data) & "values" %in% colnames(marine_data)) {
  # Select two lakes for comparison
  lake_comparison <- marine_data %>%
    filter(lake %in% c("Michigan", "Superior")) %>%
    select(lake, values)

  # Perform t-test
  t_test_result <- t.test(values ~ lake, data = lake_comparison)

  # Display the results
  print(t_test_result)

  # Visualize the comparison
  ggplot(lake_comparison, aes(x = lake, y = values)) +
    geom_boxplot(fill = "lightblue") +
    labs(title = "Comparison of Fishing Yields Between Lakes",
         x = "Lake", y = "Yield Values") +
```

```
    theme_minimal()
} else {
  # If the columns don't match exactly, adapt to the actual structure
  # This is a fallback to ensure the code runs with the actual data
  print("Column names don't match expected structure. Adapting...")

  # Identify numeric columns for analysis
  numeric_cols <- sapply(marine_data, is.numeric)
  if(sum(numeric_cols) > 0) {
    numeric_col <- names(marine_data)[numeric_cols][1]

    # Identify a categorical column for grouping
    cat_cols <- sapply(marine_data, function(x) is.character(x) || is.factor(x))
    if(sum(cat_cols) > 0) {
      cat_col <- names(marine_data)[cat_cols][1]

      # Get the two most frequent categories
      top_categories <- names(sort(table(marine_data[[cat_col]]), decreasing = TRUE)[1:2])

      # Filter data for these categories
      comparison_data <- marine_data %>%
        filter(!!sym(cat_col) %in% top_categories) %>%
        select(!!sym(cat_col), !!sym(numeric_col))

      # Rename columns for easier formula creation
      names(comparison_data) <- c("category", "value")

      # Perform t-test
      t_test_result <- t.test(value ~ category, data = comparison_data)

      # Display the results
      print(t_test_result)

      # Visualize the comparison
      ggplot(comparison_data, aes(x = category, y = value)) +
        geom_boxplot(fill = "lightblue") +
        labs(title = paste("Comparison of", numeric_col, "Between Groups"),
             x = cat_col, y = numeric_col) +
        theme_minimal()
    }
  }
}
```

```
    Welch Two Sample t-test

data:  values by lake
t = 7.0924, df = 16555, p-value = 1.371e-12
alternative hypothesis: true difference in means between group Michigan and group Superior is not equal
95 percent confidence interval:
 164.1330 289.5019
sample estimates:
mean in group Michigan mean in group Superior
              759.5080               532.6905
```



4.6 Types of Errors in Hypothesis Testing

4.6.1 Type I and Type II Errors

In hypothesis testing, two types of errors can occur:

1. **Type I Error**: Rejecting a true null hypothesis (false positive).
   - Probability = (significance level)
   - Example: Concluding there's a difference in tree heights when there actually isn't.
2. **Type II Error**: Failing to reject a false null hypothesis (false negative).
   - Probability =
   - Example: Failing to detect a real difference in tree heights.

4.6.2 Statistical Power

Statistical power is the probability of correctly rejecting a false null hypothesis (1 -). Factors affecting power include:

1. Sample size
2. Effect size
3. Significance level ( )
4. Variability in the data

```
# Demonstrate power calculation for a t-test
library(pwr)

# Calculate power for our example
effect_size <- (28 - 25) / 5  # (mean difference) / standard deviation
```

```
power_result <- pwr.t.test(
  n = 30,                    # Sample size per group
  d = effect_size,           # Cohen's d effect size
  sig.level = 0.05,          # Significance level
  type = "two.sample",       # Two-sample t-test
  alternative = "two.sided"  # Two-sided alternative
)

print(power_result)
```

```
     Two-sample t test power calculation

              n = 30
              d = 0.6
      sig.level = 0.05
          power = 0.6275046
    alternative = two.sided

NOTE: n is number in *each* group
```

```
# Calculate required sample size for 80% power
sample_size_result <- pwr.t.test(
  d = effect_size,           # Cohen's d effect size
  sig.level = 0.05,          # Significance level
  power = 0.8,               # Desired power
  type = "two.sample",       # Two-sample t-test
  alternative = "two.sided"  # Two-sided alternative
)

print(sample_size_result)
```

```
     Two-sample t test power calculation

              n = 44.58577
              d = 0.6
      sig.level = 0.05
          power = 0.8
    alternative = two.sided

NOTE: n is number in *each* group
```

4.7 One-Sample Tests

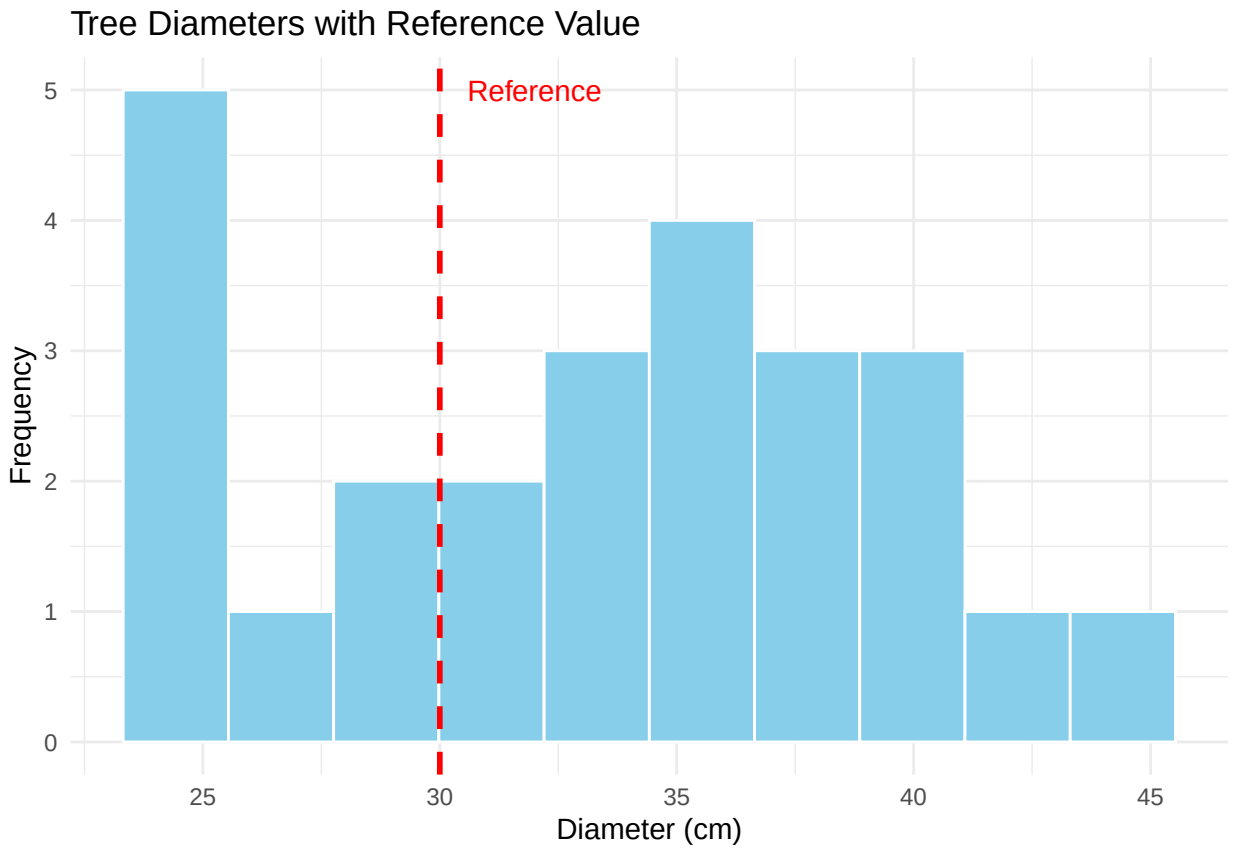
One-sample tests compare a sample mean to a known or hypothesized population value.

4.7.1 One-Sample t-Test

The one-sample t-test is used when: - The sample is approximately normally distributed - The population standard deviation is unknown

```
# Simulate tree diameter data
set.seed(456)
```

```r
tree_diameters <- rnorm(25, mean = 32, sd = 5)  # 25 trees with mean diameter 32cm

# Known reference value (e.g., from previous studies)
reference_diameter <- 30  # cm

# Visualize the data
ggplot(data.frame(diameter = tree_diameters), aes(x = diameter)) +
  geom_histogram(bins = 10, fill = "skyblue", color = "white") +
  geom_vline(xintercept = reference_diameter, color = "red", linetype = "dashed", size = 1) +
  labs(title = "Tree Diameters with Reference Value",
       x = "Diameter (cm)",
       y = "Frequency") +
  annotate("text", x = reference_diameter + 2, y = 5,
           label = "Reference", color = "red") +
  theme_minimal()
```




```r
# Perform a one-sample t-test
one_sample_result <- t.test(tree_diameters, mu = reference_diameter)
print(one_sample_result)
```

```
    One Sample t-test

data:  tree_diameters
t = 2.7309, df = 24, p-value = 0.01165
alternative hypothesis: true mean is not equal to 30
95 percent confidence interval:
```

```
 30.79206 35.69358
sample estimates:
mean of x
 33.24282
```

4.8 Two-Sample Tests

Two-sample tests compare means between two independent groups.

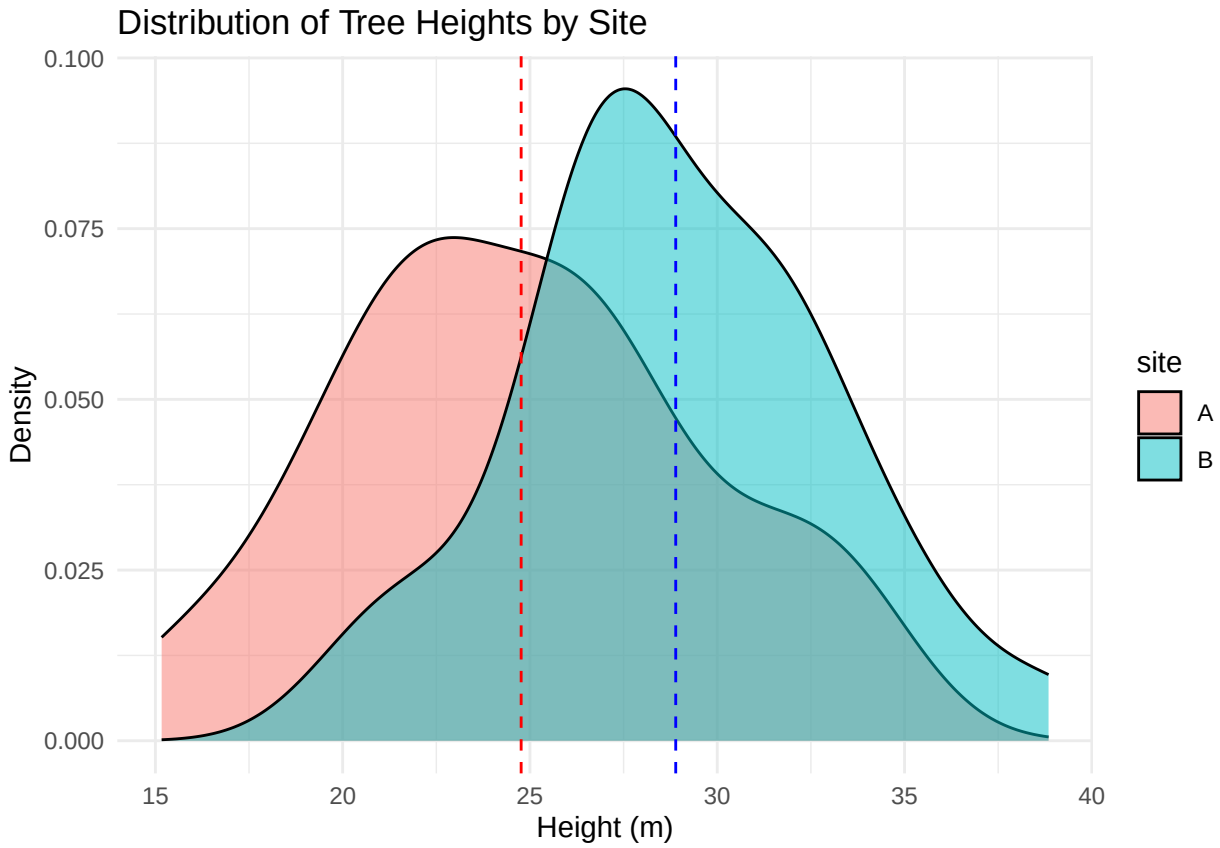
4.8.1 Independent Samples t-Test

The independent samples t-test is used when: - Both samples are approximately normally distributed - The two samples are independent

```
# We already performed this test in our initial example
# Let's visualize it differently

# Create density plots
ggplot(tree_data, aes(x = height, fill = site)) +
  geom_density(alpha = 0.5) +
  labs(title = "Distribution of Tree Heights by Site",
       x = "Height (m)",
       y = "Density") +
  theme_minimal()
```


```
# Add mean lines
ggplot(tree_data, aes(x = height, fill = site)) +
```

```
  geom_density(alpha = 0.5) +
  geom_vline(xintercept = mean(site_A), color = "red", linetype = "dashed") +
  geom_vline(xintercept = mean(site_B), color = "blue", linetype = "dashed") +
  labs(title = "Distribution of Tree Heights by Site",
       x = "Height (m)",
       y = "Density") +
  theme_minimal()
```

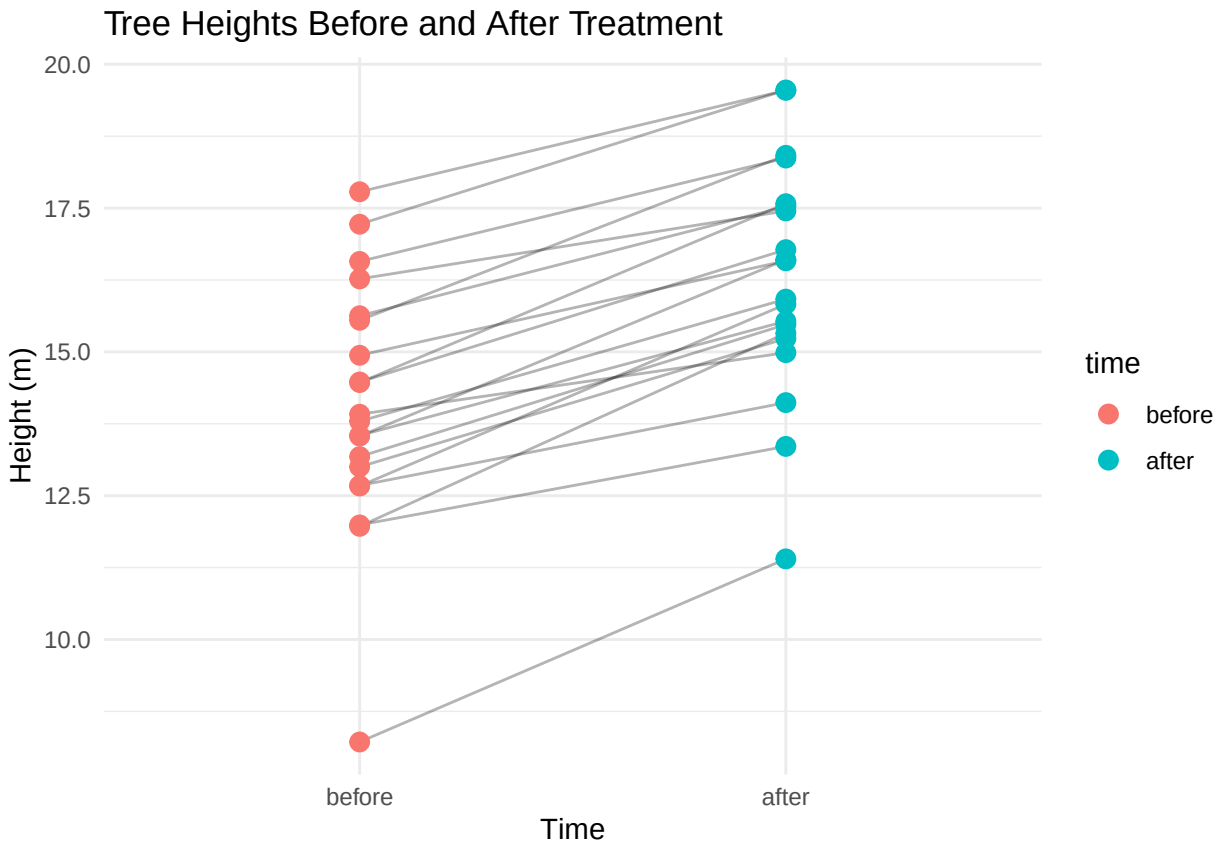


4.8.2 Paired Samples t-Test

The paired samples t-test is used when: - Measurements are taken from the same subjects under different conditions - The differences between pairs are approximately normally distributed

```
# Simulate paired data (e.g., tree growth before and after treatment)
set.seed(789)
before_treatment <- rnorm(20, mean = 15, sd = 3)
after_treatment <- before_treatment + rnorm(20, mean = 2.5, sd = 1)  # Growth effect

# Create a data frame
growth_data <- data.frame(
  tree_id = 1:20,
  before = before_treatment,
  after = after_treatment,
  difference = after_treatment - before_treatment
)

# Visualize paired data
```

```
growth_long <- reshape2::melt(growth_data[, c("tree_id", "before", "after")],
                              id.vars = "tree_id",
                              variable.name = "time",
                              value.name = "height")

ggplot(growth_long, aes(x = time, y = height, group = tree_id)) +
  geom_line(alpha = 0.3) +
  geom_point(aes(color = time), size = 3) +
  labs(title = "Tree Heights Before and After Treatment",
       x = "Time",
       y = "Height (m)") +
  theme_minimal()
```




```
# Perform a paired t-test
paired_result <- t.test(growth_data$after, growth_data$before, paired = TRUE)
print(paired_result)
```

```
    Paired t-test

data:  growth_data$after and growth_data$before
t = 13.843, df = 19, p-value = 2.239e-11
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 1.874955 2.542925
sample estimates:
mean difference
```

```
2.20894
```

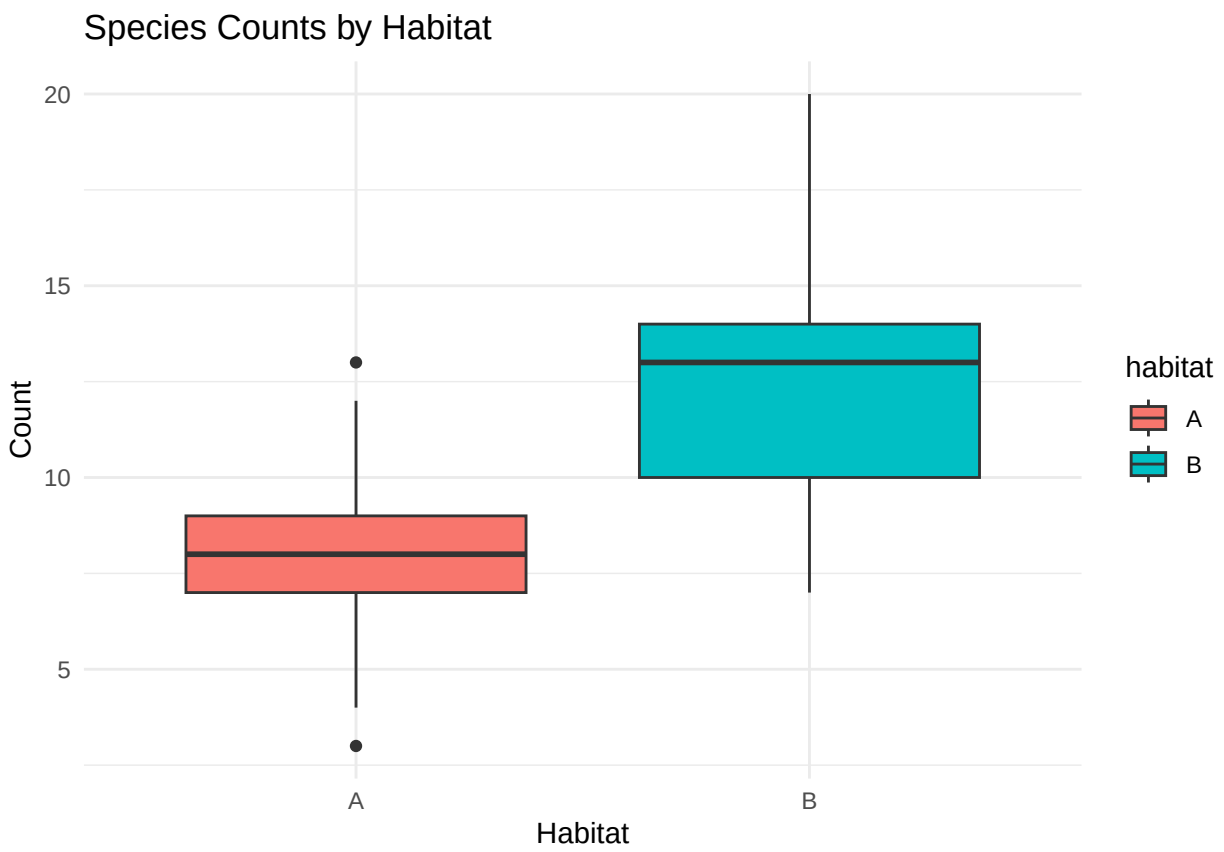
4.9 Non-Parametric Tests

Non-parametric tests are used when the assumptions of parametric tests (like normality) are violated.

4.9.1 Mann-Whitney U Test (Wilcoxon Rank-Sum Test)

This is a non-parametric alternative to the independent samples t-test.

```
# Simulate non-normal data (e.g., species counts in two habitats)
set.seed(101)
habitat_A <- rpois(25, lambda = 8)  # Poisson distribution for count data
habitat_B <- rpois(25, lambda = 12)

# Create a data frame
species_data <- data.frame(
  count = c(habitat_A, habitat_B),
  habitat = factor(rep(c("A", "B"), each = 25))
)

# Visualize the data
ggplot(species_data, aes(x = habitat, y = count, fill = habitat)) +
  geom_boxplot() +
  labs(title = "Species Counts by Habitat",
       x = "Habitat",
       y = "Count") +
  theme_minimal()
```




```
# Check for normality
shapiro.test(habitat_A)
```

```
    Shapiro-Wilk normality test

data:  habitat_A
W = 0.97173, p-value = 0.6892
```

```
shapiro.test(habitat_B)
```

```
    Shapiro-Wilk normality test

data:  habitat_B
W = 0.97562, p-value = 0.7869
```

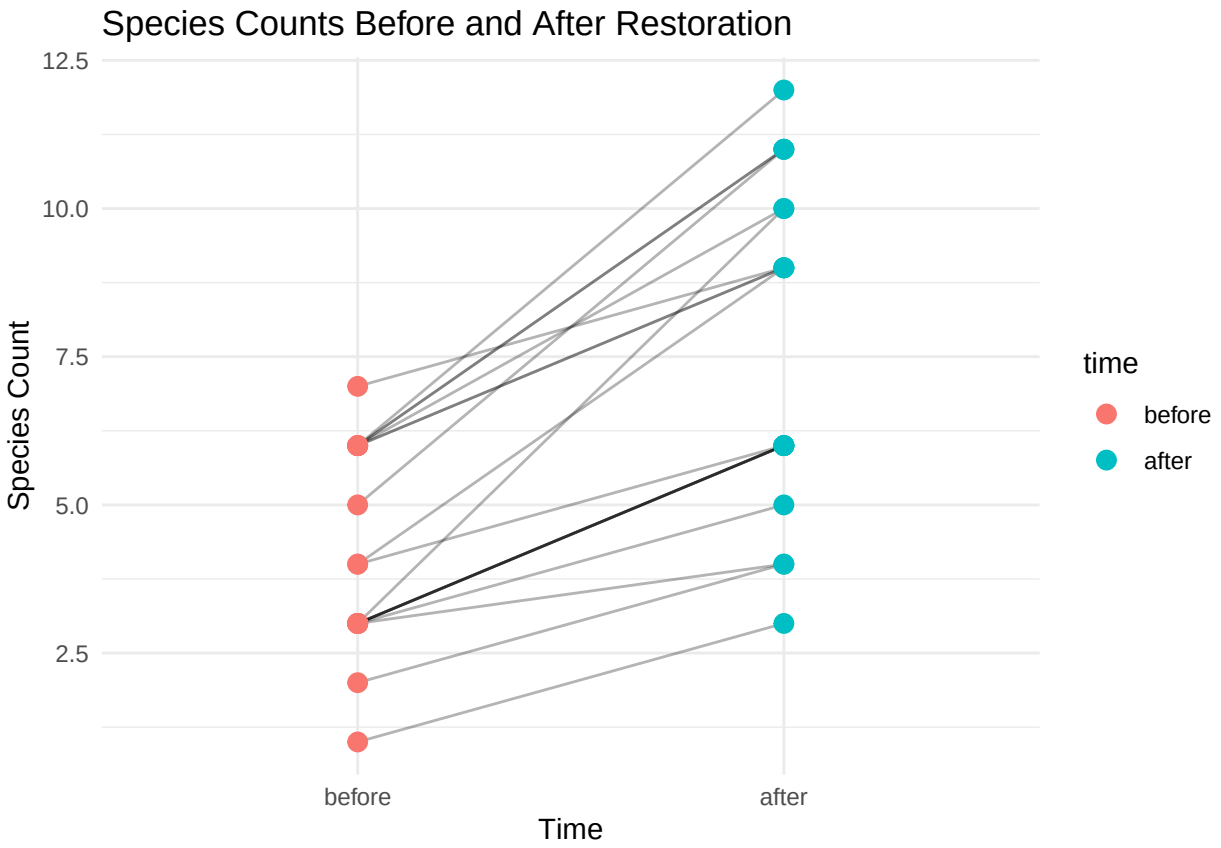
```
# Perform Mann-Whitney U test
wilcox_result <- wilcox.test(count ~ habitat, data = species_data)
print(wilcox_result)
```

```
    Wilcoxon rank sum test with continuity correction

data:  count by habitat
W = 88.5, p-value = 1.308e-05
alternative hypothesis: true location shift is not equal to 0
```

4.9.2 Wilcoxon Signed-Rank Test

This is a non-parametric alternative to the paired samples t-test.

```
# Simulate non-normal paired data
set.seed(202)
before_restoration <- rpois(20, lambda = 5)
after_restoration <- before_restoration + rpois(20, lambda = 3)

# Create a data frame
restoration_data <- data.frame(
  site_id = 1:20,
  before = before_restoration,
  after = after_restoration,
  difference = after_restoration - before_restoration
)

# Visualize paired data
restoration_long <- reshape2::melt(restoration_data[, c("site_id", "before", "after")],
                                   id.vars = "site_id",
                                   variable.name = "time",
                                   value.name = "species_count")

ggplot(restoration_long, aes(x = time, y = species_count, group = site_id)) +
  geom_line(alpha = 0.3) +
  geom_point(aes(color = time), size = 3) +
  labs(title = "Species Counts Before and After Restoration",
       x = "Time",
       y = "Species Count") +
  theme_minimal()
```




```
# Perform Wilcoxon signed-rank test
wilcox_paired_result <- wilcox.test(restoration_data$after, restoration_data$before, paired = TRUE)
print(wilcox_paired_result)
```

```
    Wilcoxon signed rank test with continuity correction

data:  restoration_data$after and restoration_data$before
V = 210, p-value = 8.527e-05
alternative hypothesis: true location shift is not equal to 0
```

4.10 Confidence Intervals

Confidence intervals provide a range of plausible values for a population parameter.

```
# Calculate 95% confidence interval for mean tree height in Site A
ci_result <- t.test(site_A)
print(ci_result$conf.int)
```

```
[1] 22.93287 26.59610
attr(,"conf.level")
[1] 0.95
```

```
# Visualize confidence interval
mean_height <- mean(site_A)
ci_lower <- ci_result$conf.int[1]
ci_upper <- ci_result$conf.int[2]
```

```
ggplot(data.frame(height = site_A), aes(x = height)) +
  geom_histogram(bins = 10, fill = "skyblue", color = "white") +
  geom_vline(xintercept = mean_height, color = "red", size = 1) +
  geom_vline(xintercept = ci_lower, color = "blue", linetype = "dashed") +
  geom_vline(xintercept = ci_upper, color = "blue", linetype = "dashed") +
  annotate("rect", xmin = ci_lower, xmax = ci_upper, ymin = 0, ymax = Inf,
           fill = "blue", alpha = 0.1) +
  labs(title = "Tree Heights in Site A with 95% Confidence Interval",
       x = "Height (m)",
       y = "Frequency") +
  annotate("text", x = mean_height, y = 8, label = "Mean", color = "red") +
  annotate("text", x = ci_lower - 1, y = 6, label = "Lower CI", color = "blue") +
  annotate("text", x = ci_upper + 1, y = 6, label = "Upper CI", color = "blue") +
  theme_minimal()
```

4.11 Hypothesis Testing in Jamovi

Jamovi provides a user-friendly interface for conducting hypothesis tests:

1. **T-Tests**: For one-sample, independent samples, and paired samples t-tests
2. **Non-Parametric Tests**: For Mann-Whitney U and Wilcoxon signed-rank tests
3. **Descriptives**: For calculating confidence intervals

4.12 Summary

In this chapter, we've covered the fundamental concepts and techniques of hypothesis testing in ecological and forestry research:

- Formulating null and alternative hypotheses
- Understanding p-values and significance levels
- Recognizing Type I and Type II errors
- Calculating and interpreting statistical power
- Conducting one-sample, two-sample, and paired tests
- Using non-parametric alternatives when necessary
- Calculating and interpreting confidence intervals

These statistical tools allow researchers to make informed inferences about populations based on sample data, helping to advance knowledge in ecology and forestry.

4.13 Exercises

1. Formulate null and alternative hypotheses for an ecological research question of your choice.
2. Simulate data for two groups and perform an independent samples t-test.
3. Calculate the statistical power for your t-test and determine the sample size needed for 80% power.
4. Create a dataset where the normality assumption is violated and compare the results of a t-test and a Mann-Whitney U test.
5. Calculate and visualize a 95% confidence interval for a sample mean.
6. Perform the same hypothesis tests in Jamovi and compare the results with those obtained in R.

4.14 Statistical Power

Statistical power is the probability of correctly rejecting the null hypothesis when it is false. It is influenced by:

1. Sample size
2. Effect size
3. Significance level ( )
4. Variability in the data

Chapter 5

Demonstrate power calculation for a t-test

```
library(pwr)
```

Chapter 6

Calculate power for our example

effect_size <- (28 - 25) / 5 # (mean difference) / standard deviation power_result <- pwr.t.test(n = 30, # Sample size per group d = effect_size, # Cohen's d effect size sig.level = 0.05, # Significance level type = “two.sample”, # Two-sample t-test alternative = “two.sided” # Two-sided alternative)

print(power_result)

Chapter 7

Calculate required sample size for 80% power

sample_size_result <- pwr.t.test(d = effect_size, # Cohen's d effect size sig.level = 0.05, # Significance level power = 0.8, # Desired power type = "two.sample", # Two-sample t-test alternative = "two.sided" # Two-sided alternative)

print(sample_size_result)

Chapter 8

Common Statistical Tests

8.1 Introduction

This chapter explores common statistical tests used in natural sciences research. Building on the hypothesis testing framework introduced in the previous chapter, we'll examine specific tests for different research scenarios and data types.

8.2 Choosing the Right Statistical Test

Selecting the appropriate statistical test depends on several factors:

1. **Research Question**: What you're trying to determine
2. **Data Type**: Categorical, continuous, or ordinal
3. **Number of Groups**: One, two, or multiple groups
4. **Data Distribution**: Normal or non-normal
5. **Independence**: Whether observations are independent or related

8.2.1 Decision Tree for Common Tests

Decision Tree for Selecting Statistical Tests:

1. **For One Variable:**
   - **One Sample:**
     - Normal, Continuous → One-Sample t-Test
     - Non-normal, Continuous → Wilcoxon Signed-Rank Test
     - Categorical → Binomial Test
   - **Two Samples:**
     - Normal, Continuous, Independent → Independent t-Test
     - Normal, Continuous, Related → Paired t-Test
     - Non-normal, Continuous, Independent → Mann-Whitney U Test
     - Non-normal, Continuous, Related → Wilcoxon Signed-Rank Test
     - Categorical, Independent → Chi-Square Test
     - Categorical, Related → McNemar Test
   - **Multiple Samples:**
     - Normal, Continuous, Independent → ANOVA
     - Normal, Continuous, Related → Repeated Measures ANOVA
     - Non-normal, Continuous, Independent → Kruskal-Wallis Test
     - Non-normal, Continuous, Related → Friedman Test
     - Categorical → Chi-Square Test

2. **For Two Variables:**
   - Normal, Continuous → Pearson Correlation
   - Non-normal or Ordinal → Spearman Correlation
   - Continuous Predictor & Outcome → Linear Regression
   - Continuous Predictor, Binary Outcome → Logistic Regression
3. **For Multiple Variables:**
   - Multiple Continuous Outcomes → MANOVA
   - Dimension Reduction → Principal Component Analysis
   - Grouping → Cluster Analysis

8.3 Parametric vs. Non-Parametric Tests

8.3.1 Parametric Tests

Parametric tests make assumptions about the underlying population distribution, typically that the data follows a normal distribution. Common parametric tests include:

- t-tests
- ANOVA
- Pearson correlation
- Linear regression

8.3.2 Non-Parametric Tests

Non-parametric tests make fewer assumptions about the population distribution and are useful when data doesn't meet the assumptions of parametric tests. Common non-parametric tests include:

- Mann-Whitney U test
- Wilcoxon signed-rank test
- Kruskal-Wallis test
- Spearman correlation

8.3.3 Checking Assumptions

Before applying a parametric test, it's essential to check if your data meets the necessary assumptions. Let's use our crop yield dataset to demonstrate:

```
# Load necessary libraries
library(tidyverse)

# Load the crop yield dataset
crop_yields <- read_csv("../data/agriculture/crop_yields.csv")

# View column names to see how R has formatted them
names(crop_yields)
```

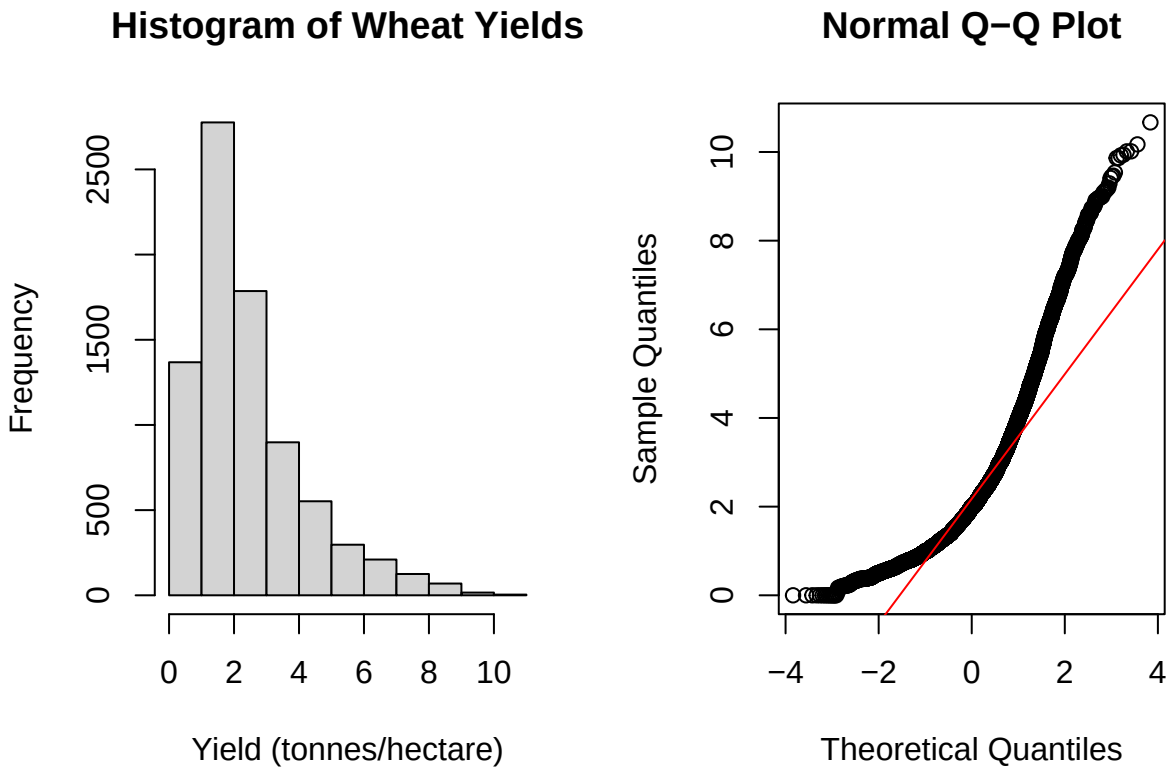
```
 [1] "Entity"                           "Code"
 [3] "Year"                             "Wheat (tonnes per hectare)"
 [5] "Rice (tonnes per hectare)"        "Maize (tonnes per hectare)"
 [7] "Soybeans (tonnes per hectare)"    "Potatoes (tonnes per hectare)"
 [9] "Beans (tonnes per hectare)"       "Peas (tonnes per hectare)"
[11] "Cassava (tonnes per hectare)"     "Barley (tonnes per hectare)"
[13] "Cocoa beans (tonnes per hectare)" "Bananas (tonnes per hectare)"
```

```
# Extract wheat yields for analysis
wheat_yields <- crop_yields %>%
```

```
  filter(!is.na(`Wheat (tonnes per hectare)`)) %>%
  select(Entity, Year, `Wheat (tonnes per hectare)`)

# View the first few rows
head(wheat_yields)
```

```
# A tibble: 6 x 3
  Entity       Year `Wheat (tonnes per hectare)`
  <chr>       <dbl>                        <dbl>
1 Afghanistan  1961                        1.02
2 Afghanistan  1962                        0.974
3 Afghanistan  1963                        0.832
4 Afghanistan  1964                        0.951
5 Afghanistan  1965                        0.972
6 Afghanistan  1966                        0.867
```

```
# Check for normality
# Visual methods
par(mfrow = c(1, 2))
hist(wheat_yields$`Wheat (tonnes per hectare)`, main = "Histogram of Wheat Yields", xlab = "Yield (tonn
qqnorm(wheat_yields$`Wheat (tonnes per hectare)`); qqline(wheat_yields$`Wheat (tonnes per hectare)`, co
```




```
# Statistical test for normality
shapiro.test(sample(wheat_yields$`Wheat (tonnes per hectare)`, min(5000, length(wheat_yields$`Wheat (to
```

```
    Shapiro-Wilk normality test

data:  sample(wheat_yields$`Wheat (tonnes per hectare)`, min(5000, length(wheat_yields$`Wheat (tonnes p
W = 0.86694, p-value < 2.2e-16
```

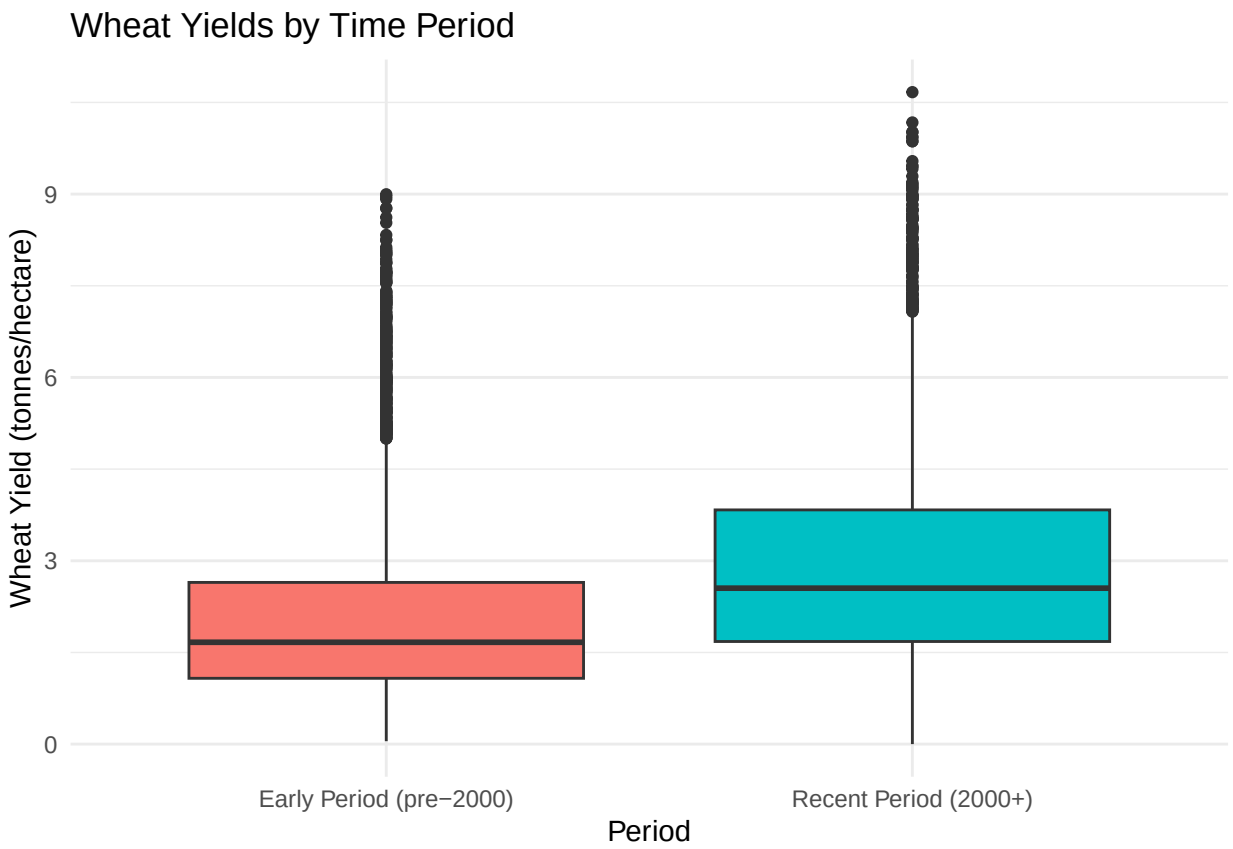
8.4 Tests for Comparing Groups

8.4.1 t-Tests

8.4.1.1 Independent Samples t-Test

Used to compare means between two independent groups. Let's compare wheat yields between two time periods:

```
# Create two groups: early period (before 2000) and recent period (2000 onwards)
crop_yields_grouped <- crop_yields %>%
  filter(!is.na(`Wheat (tonnes per hectare)`) & Year >= 1960) %>%
  mutate(period = ifelse(Year < 2000, "Early Period (pre-2000)", "Recent Period (2000+)"))

# Visualize the data
ggplot(crop_yields_grouped, aes(x = period, y = `Wheat (tonnes per hectare)`, fill = period)) +
  geom_boxplot() +
  labs(title = "Wheat Yields by Time Period",
       x = "Period",
       y = "Wheat Yield (tonnes/hectare)") +
  theme_minimal() +
  theme(legend.position = "none")
```




```
# Perform independent samples t-test using formula interface with backticks
t_test_result <- t.test(`Wheat (tonnes per hectare)` ~ period, data = crop_yields_grouped)
t_test_result
```

```
    Welch Two Sample t-test
```

```
data:  Wheat (tonnes per hectare) by period
t = -22.335, df = 4970.2, p-value < 2.2e-16
alternative hypothesis: true difference in means between group Early Period (pre-2000) and group Recent
95 percent confidence interval:
 -0.9798533 -0.8217198
sample estimates:
mean in group Early Period (pre-2000)   mean in group Recent Period (2000+)
                             2.109781                              3.010567
```

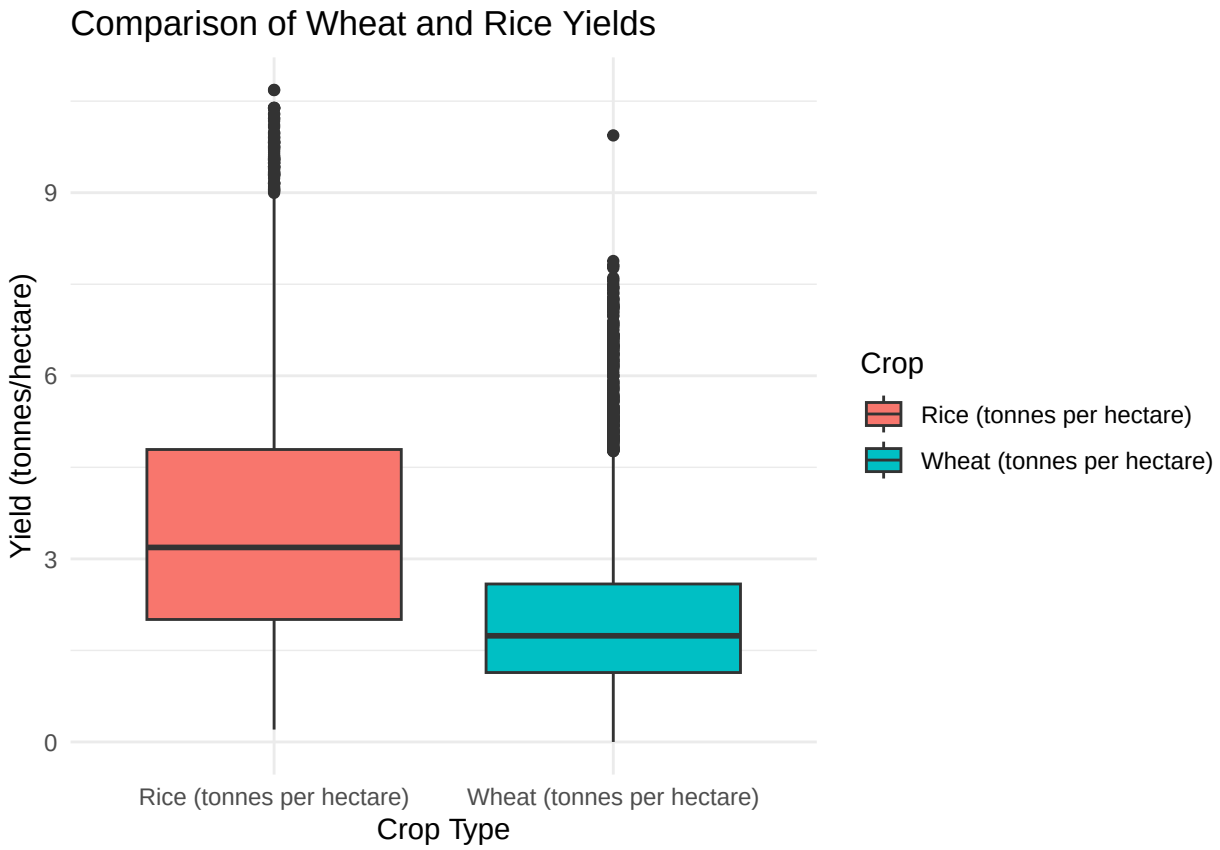
8.4.1.2 Paired Samples t-Test

Used to compare means between two related groups. Let's compare wheat and rice yields for the same countries and years:

```
# Prepare data for paired t-test
paired_data <- crop_yields %>%
  filter(!is.na(`Wheat (tonnes per hectare)`) & !is.na(`Rice (tonnes per hectare)`)) %>%
  select(Entity, Year, `Wheat (tonnes per hectare)`, `Rice (tonnes per hectare)`)

# View the first few rows
head(paired_data)
```

```
# A tibble: 6 x 4
  Entity       Year `Wheat (tonnes per hectare)` `Rice (tonnes per hectare)`
  <chr>       <dbl>                        <dbl>                       <dbl>
1 Afghanistan  1961                        1.02                         1.52
2 Afghanistan  1962                        0.974                        1.52
3 Afghanistan  1963                        0.832                        1.52
4 Afghanistan  1964                        0.951                        1.73
5 Afghanistan  1965                        0.972                        1.73
6 Afghanistan  1966                        0.867                        1.52
```

```
# Visualize the paired data
paired_data_long <- paired_data %>%
  pivot_longer(cols = c(`Wheat (tonnes per hectare)`, `Rice (tonnes per hectare)`), names_to = "Crop", v

ggplot(paired_data_long, aes(x = Crop, y = Yield, fill = Crop)) +
  geom_boxplot() +
  labs(title = "Comparison of Wheat and Rice Yields",
       x = "Crop Type",
       y = "Yield (tonnes/hectare)") +
  theme_minimal()
```




```
# Perform paired t-test using vectors directly
paired_t_test <- t.test(
  paired_data$`Wheat (tonnes per hectare)`,
  paired_data$`Rice (tonnes per hectare)`,
  paired = TRUE
)
paired_t_test
```

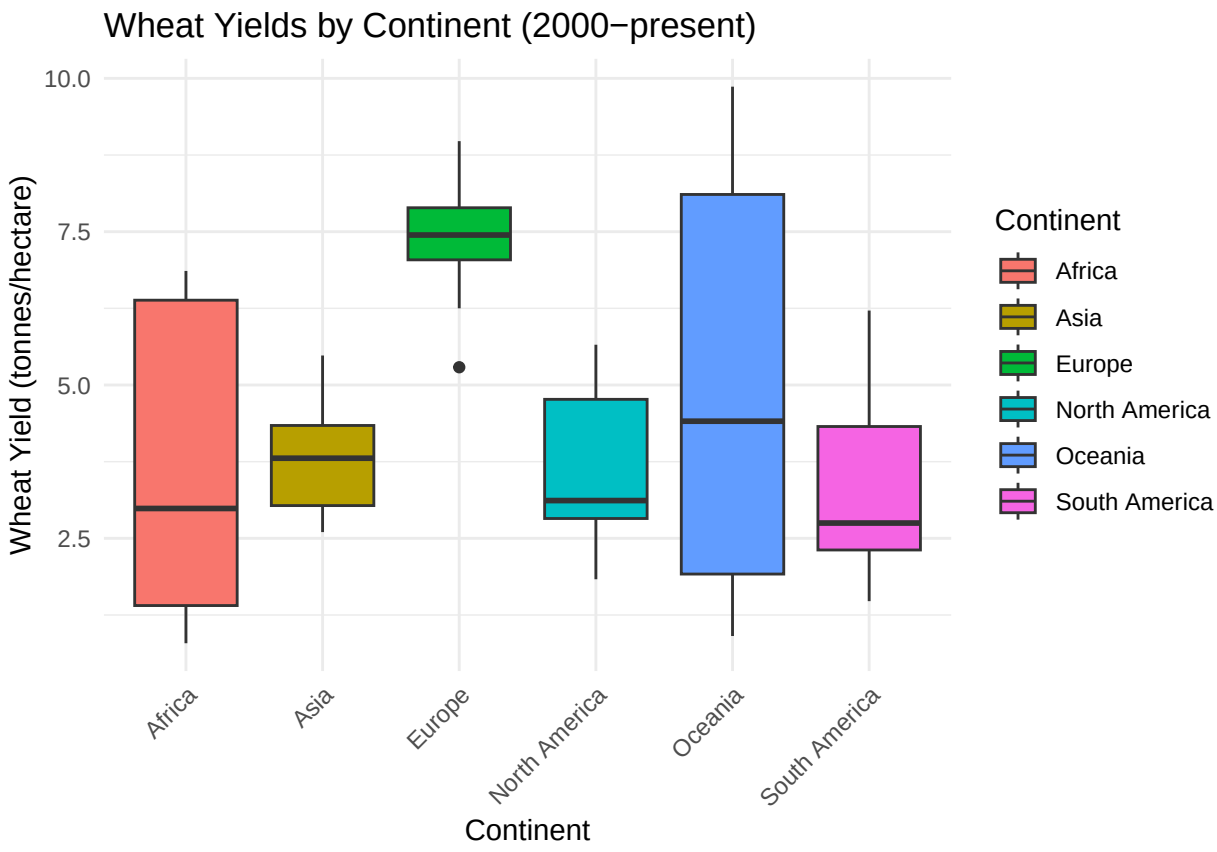
```
    Paired t-test

data:  paired_data$`Wheat (tonnes per hectare)` and paired_data$`Rice (tonnes per hectare)`
t = -61.854, df = 5725, p-value < 2.2e-16
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 -1.565063 -1.468905
sample estimates:
mean difference
      -1.516984
```

8.4.2 Analysis of Variance (ANOVA)

ANOVA is used to compare means among three or more independent groups. Let's compare crop yields across different continents:

```
# Create a mapping of countries to continents (simplified for demonstration)
continent_mapping <- tibble(
  Entity = c("United States", "Canada", "Mexico",
```

```
              "China", "India", "Japan",
              "Germany", "France", "United Kingdom",
              "Brazil", "Argentina", "Chile",
              "Egypt", "Nigeria", "South Africa",
              "Australia", "New Zealand"),
  Continent = c(rep("North America", 3),
                rep("Asia", 3),
                rep("Europe", 3),
                rep("South America", 3),
                rep("Africa", 3),
                rep("Oceania", 2))
)

# Join with crop yields data
continental_yields <- crop_yields %>%
  inner_join(continent_mapping, by = "Entity") %>%
  filter(!is.na(`Wheat (tonnes per hectare)`) & Year >= 2000)

# Visualize wheat yields by continent
ggplot(continental_yields, aes(x = Continent, y = `Wheat (tonnes per hectare)`, fill = Continent)) +
  geom_boxplot() +
  labs(title = "Wheat Yields by Continent (2000-present)",
       x = "Continent",
       y = "Wheat Yield (tonnes/hectare)") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



```
# Perform ANOVA
anova_result <- aov(`Wheat (tonnes per hectare)` ~ Continent, data = continental_yields)
summary(anova_result)
```

```
             Df Sum Sq Mean Sq F value Pr(>F)
Continent     5  698.6  139.73   48.64 <2e-16 ***
Residuals   317  910.7    2.87
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Post-hoc test to identify which groups differ
TukeyHSD(anova_result)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = `Wheat (tonnes per hectare)` ~ Continent, data = continental_yields)

$Continent
                                  diff         lwr        upr     p adj
Asia-Africa                  0.27759825 -0.63272412  1.1879206 0.9523916
Europe-Africa                3.89405789  2.98373553  4.8043803 0.0000000
North America-Africa         0.06069825 -0.84962412  0.9710206 0.9999646
Oceania-Africa               1.36844649  0.35067515  2.3862178 0.0019309
South America-Africa        -0.21777193 -1.12809429  0.6925504 0.9834279
Europe-Asia                  3.61645965  2.70613729  4.5267820 0.0000000
North America-Asia          -0.21690000 -1.12722236  0.6934224 0.9837237
Oceania-Asia                 1.09084825  0.07307691  2.1086196 0.0276496
South America-Asia          -0.49537018 -1.40569254  0.4149522 0.6253648
North America-Europe        -3.83335965 -4.74368201 -2.9230373 0.0000000
Oceania-Europe              -2.52561140 -3.54338274 -1.5078401 0.0000000
South America-Europe        -4.11182982 -5.02215219 -3.2015075 0.0000000
Oceania-North America        1.30774825  0.28997691  2.3255196 0.0036420
South America-North America -0.27847018 -1.18879254  0.6318522 0.9517629
South America-Oceania       -1.58621842 -2.60398976 -0.5684471 0.0001587
```

8.4.3 Non-Parametric Alternatives

8.4.3.1 Mann-Whitney U Test

The Mann-Whitney U test (also called Wilcoxon rank-sum test) is a non-parametric alternative to the independent samples t-test:

```
# Using the same time period groups as before
wilcox_test <- wilcox.test(`Wheat (tonnes per hectare)` ~ period, data = crop_yields_grouped)
wilcox_test
```

```
    Wilcoxon rank sum test with continuity correction

data:  Wheat (tonnes per hectare) by period
W = 5031268, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0
```

8.4.3.2 Kruskal-Wallis Test

The Kruskal-Wallis test is a non-parametric alternative to ANOVA:

```
# Using the same continental data as before
kruskal_result <- kruskal.test(`Wheat (tonnes per hectare)` ~ Continent, data = continental_yields)
kruskal_result
```

```
    Kruskal-Wallis rank sum test

data:  Wheat (tonnes per hectare) by Continent
Kruskal-Wallis chi-squared = 120.17, df = 5, p-value < 2.2e-16
```

```
# Post-hoc test for Kruskal-Wallis
if(requireNamespace("dunn.test", quietly = TRUE)) {
  library(dunn.test)
  dunn.test(continental_yields$`Wheat (tonnes per hectare)`, continental_yields$Continent, method = "bo
} else {
  # Alternative: pairwise Wilcoxon tests
  pairwise.wilcox.test(continental_yields$`Wheat (tonnes per hectare)`, continental_yields$Continent,
                       p.adjust.method = "bonferroni")
}
```

```
  Kruskal-Wallis rank sum test

data: x and group
Kruskal-Wallis chi-squared = 120.1715, df = 5, p-value = 0

                           Comparison of x by group
                                 (Bonferroni)
Col Mean-|
Row Mean |     Africa       Asia     Europe   North Am    Oceania
---------+-----------------------------------------------------
    Asia |  -1.814776
         |     0.5217
         |
  Europe |  -9.079400  -7.264623
         |    0.0000*    0.0000*
         |
North Am |  -0.948758   0.866018   8.130641
         |     1.0000     1.0000    0.0000*
         |
 Oceania |  -2.181366  -0.558180   5.939496  -1.332770
         |     0.2187     1.0000    0.0000*     1.0000
         |
South Am |   0.300874   2.115651   9.380275   1.249633   2.450476
         |     1.0000     0.2578    0.0000*     1.0000     0.1070

alpha = 0.05
Reject Ho if p <= alpha/2
```

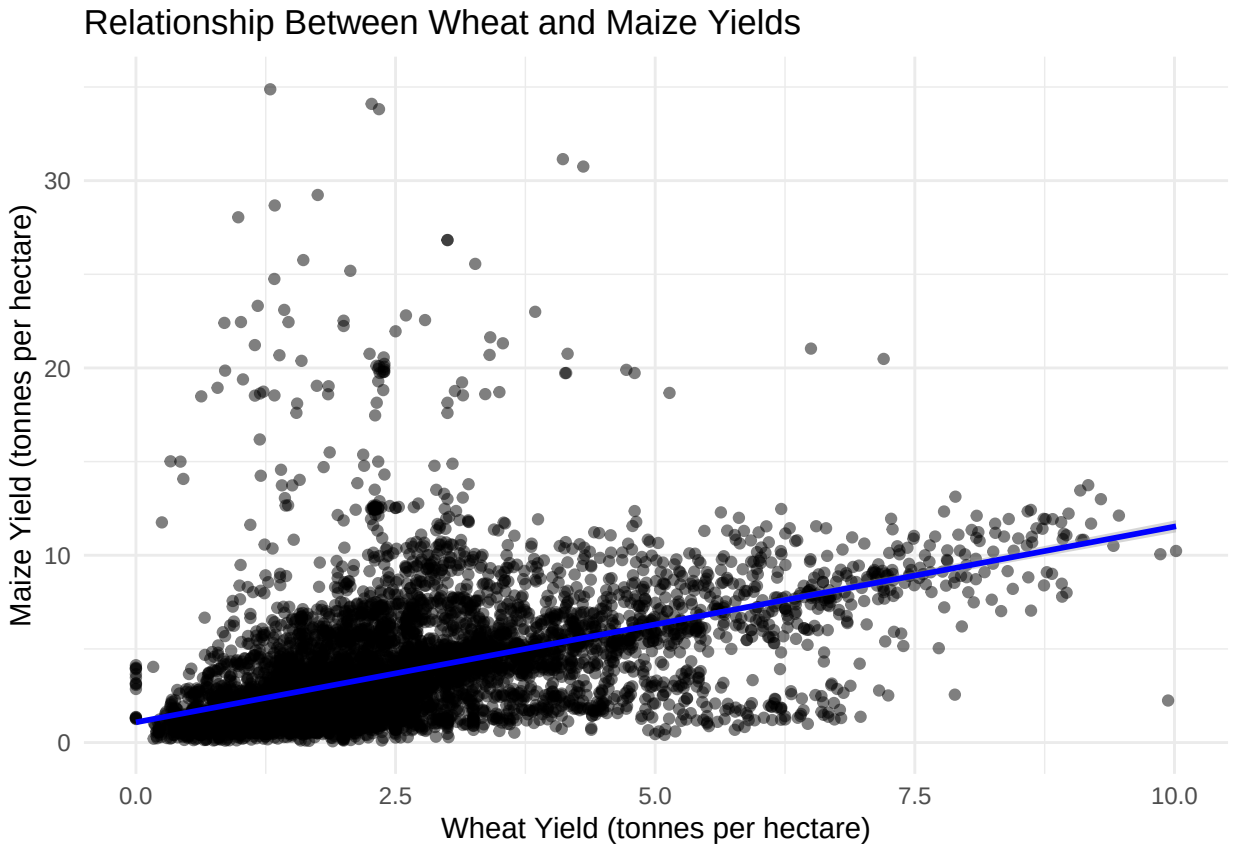
8.5 Tests for Relationships

8.5.1 Correlation Analysis

8.5.1.1 Pearson Correlation

Pearson correlation measures the linear relationship between two continuous variables:

```
# Examine correlation between wheat and maize yields
crop_correlation <- crop_yields %>%
  filter(!is.na(`Wheat (tonnes per hectare)`) & !is.na(`Maize (tonnes per hectare)`)) %>%
  select(Entity, Year, `Wheat (tonnes per hectare)`, `Maize (tonnes per hectare)`)

# Visualize the relationship
ggplot(crop_correlation, aes(x = `Wheat (tonnes per hectare)`, y = `Maize (tonnes per hectare)`)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", se = TRUE, color = "blue") +
  labs(title = "Relationship Between Wheat and Maize Yields",
       x = "Wheat Yield (tonnes per hectare)",
       y = "Maize Yield (tonnes per hectare)") +
  theme_minimal()
```




```
# Calculate Pearson correlation
cor_result <- cor.test(crop_correlation$`Wheat (tonnes per hectare)`, crop_correlation$`Maize (tonnes p
cor_result
```

```
    Pearson's product-moment correlation
```

```
data:  crop_correlation$`Wheat (tonnes per hectare)` and crop_correlation$`Maize (tonnes per hectare)`
t = 49.748, df = 7378, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.4838956 0.5180698
sample estimates:
      cor
0.5011781
```

8.5.1.2 Spearman Correlation

Spearman correlation is a non-parametric measure of rank correlation:

```
# Calculate Spearman correlation
spearman_result <- cor.test(crop_correlation$`Wheat (tonnes per hectare)`, crop_correlation$`Maize (ton
spearman_result
```

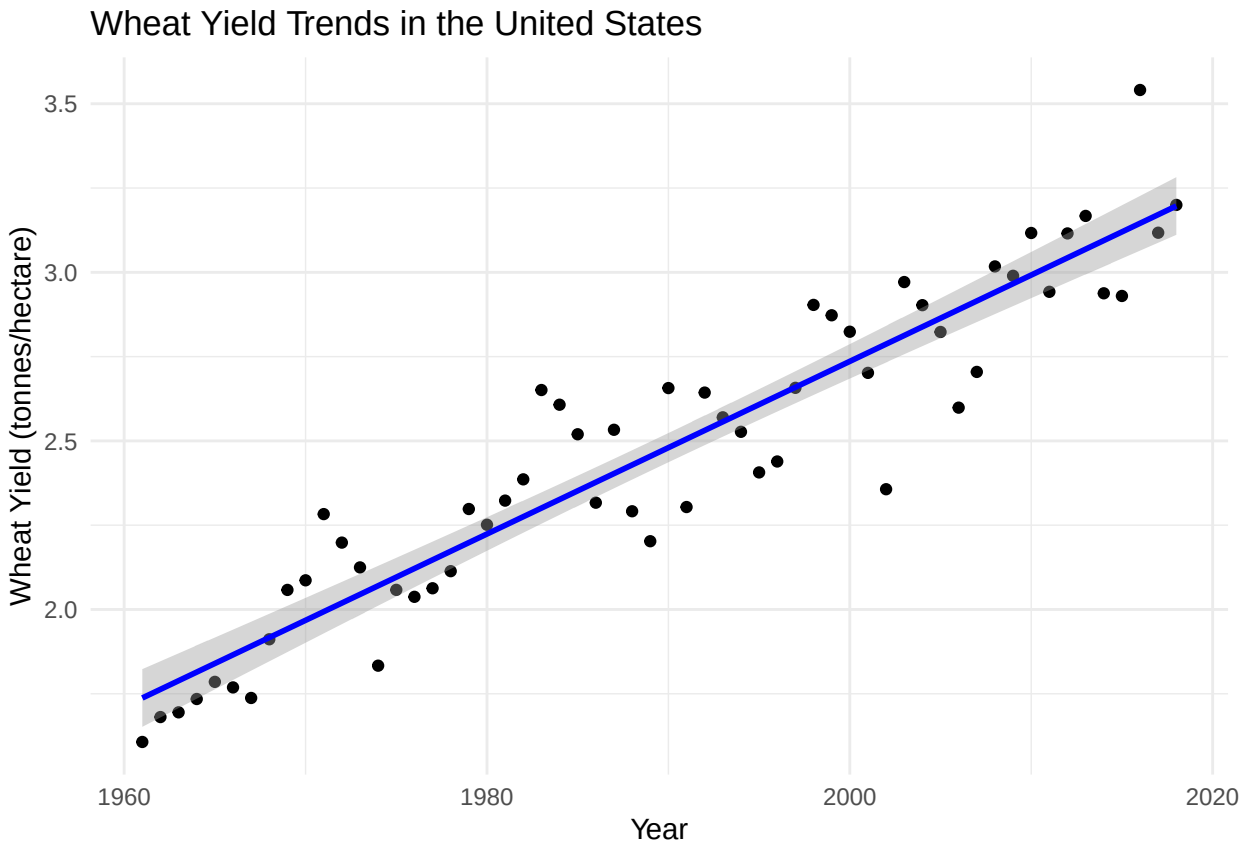
```
    Spearman's rank correlation rho

data:  crop_correlation$`Wheat (tonnes per hectare)` and crop_correlation$`Maize (tonnes per hectare)`
S = 2.4618e+10, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.6325152
```

8.5.2 Regression Analysis

8.5.2.1 Linear Regression

Linear regression models the relationship between a dependent variable and one or more independent variables:

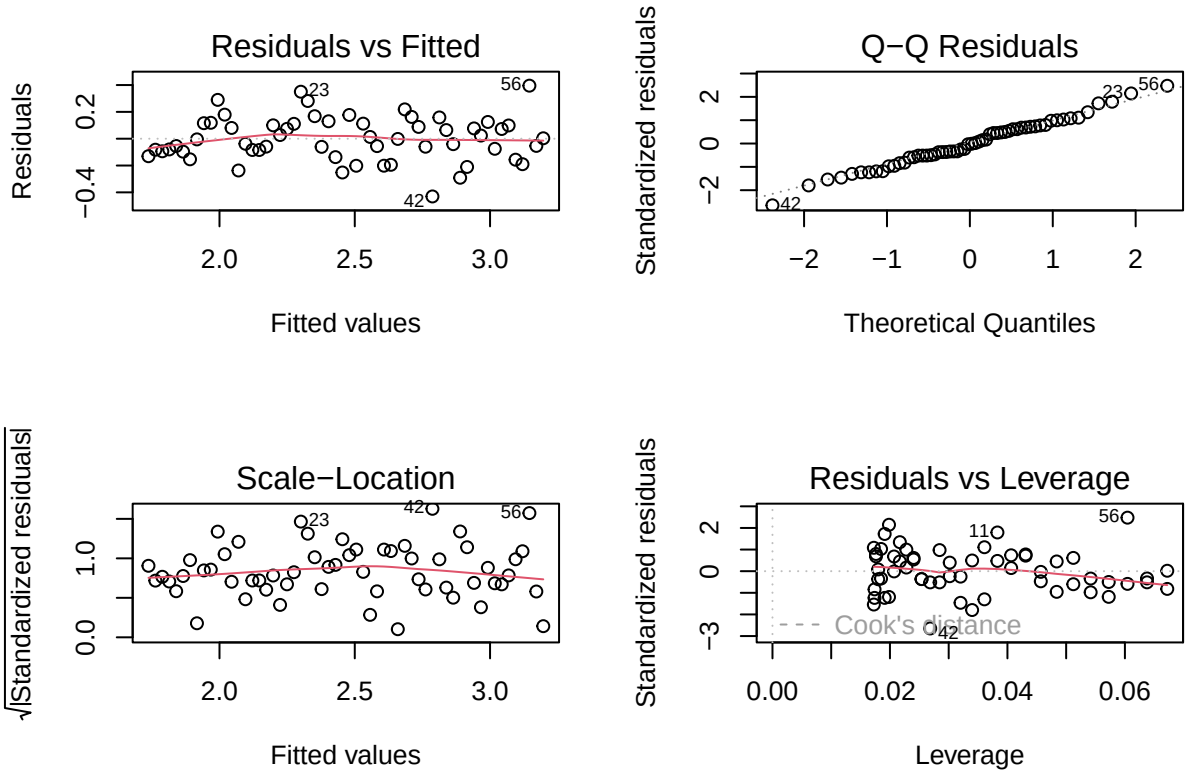
```
# Create a dataset with year as predictor for wheat yields
time_series_data <- crop_yields %>%
  filter(Entity == "United States" & !is.na(`Wheat (tonnes per hectare)`)) %>%
  arrange(Year)

# Visualize the trend
ggplot(time_series_data, aes(x = Year, y = `Wheat (tonnes per hectare)`)) +
  geom_point() +
  geom_smooth(method = "lm", se = TRUE, color = "blue") +
  labs(title = "Wheat Yield Trends in the United States",
       x = "Year",
       y = "Wheat Yield (tonnes/hectare)") +
  theme_minimal()
```




```r
# Perform linear regression
lm_model <- lm(`Wheat (tonnes per hectare)` ~ Year, data = time_series_data)
summary(lm_model)
```

```
Call:
lm(formula = `Wheat (tonnes per hectare)` ~ Year, data = time_series_data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.43042 -0.09139 -0.00340  0.11184  0.39526

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -48.465987   2.571017  -18.85   <2e-16 ***
Year          0.025601   0.001292   19.81   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1648 on 56 degrees of freedom
Multiple R-squared:  0.8751,    Adjusted R-squared:  0.8729
F-statistic: 392.5 on 1 and 56 DF,  p-value: < 2.2e-16
```

```r
# Check assumptions
par(mfrow = c(2, 2))
plot(lm_model)
```



8.5.2.2 Multiple Regression

Multiple regression includes more than one predictor variable:

```
# Create a dataset with multiple predictors
multi_crop_data <- crop_yields %>%
  filter(!is.na(`Wheat (tonnes per hectare)`) & !is.na(`Rice (tonnes per hectare)`) & !is.na(`Maize (to
  select(Entity, Year, `Wheat (tonnes per hectare)`, `Rice (tonnes per hectare)`, `Maize (tonnes per he

# Perform multiple regression
multi_model <- lm(`Wheat (tonnes per hectare)` ~ `Rice (tonnes per hectare)` + `Maize (tonnes per hectar
summary(multi_model)
```

```
Call:
lm(formula = `Wheat (tonnes per hectare)` ~ `Rice (tonnes per hectare)` +
    `Maize (tonnes per hectare)` + Year, data = multi_crop_data)

Residuals:
    Min      1Q  Median      3Q     Max
-2.7029 -0.6135 -0.2079  0.3877  7.8709

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                 -2.161e+01  1.776e+00 -12.171   <2e-16 ***
`Rice (tonnes per hectare)` -5.426e-03  9.720e-03  -0.558    0.577
`Maize (tonnes per hectare)` 2.790e-01  8.512e-03  32.776   <2e-16 ***
Year                         1.148e-02  8.965e-04  12.810   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 1.025 on 5722 degrees of freedom
Multiple R-squared:  0.3411,    Adjusted R-squared:  0.3407
F-statistic: 987.2 on 3 and 5722 DF,  p-value: < 2.2e-16
```

8.6 Tests for Categorical Data

8.6.1 Chi-Square Test

The Chi-Square test examines the association between categorical variables. Let's use our biodiversity dataset:

```
# Load the biodiversity dataset
plants <- read_csv("../data/ecology/biodiversity.csv")

# Create a contingency table of red list categories by plant group
if("red_list_category" %in% colnames(plants) & "group" %in% colnames(plants)) {
  # Create a contingency table
  contingency_table <- table(plants$red_list_category, plants$group)

  # View the table
  contingency_table

  # Perform Chi-Square test
  chi_sq_result <- chisq.test(contingency_table)
  chi_sq_result

  # Examine residuals to understand the pattern of association
  chi_sq_result$residuals
} else {
  # If the expected columns don't exist, create a demonstration with available data
  message("Required columns not found. Creating a demonstration with available columns.")

  # Identify categorical columns
  categorical_cols <- sapply(plants, function(x) is.character(x) || is.factor(x))
  cat_col_names <- names(plants)[categorical_cols]

  if(length(cat_col_names) >= 2) {
    # Select the first two categorical columns
    col1 <- cat_col_names[1]
    col2 <- cat_col_names[2]

    # Create a contingency table
    contingency_table <- table(plants[[col1]], plants[[col2]])

    # View the table
    print(paste("Contingency table of", col1, "by", col2))
    print(contingency_table)

    # Perform Chi-Square test if appropriate
    if(min(dim(contingency_table)) > 1 && sum(contingency_table) > 0) {
      chi_sq_result <- chisq.test(contingency_table, simulate.p.value = TRUE)
      print(chi_sq_result)
    } else {
```

```
      message("Contingency table not suitable for Chi-Square test.")
    }
  } else {
    message("Not enough categorical columns found for Chi-Square test demonstration.")
  }
}
```

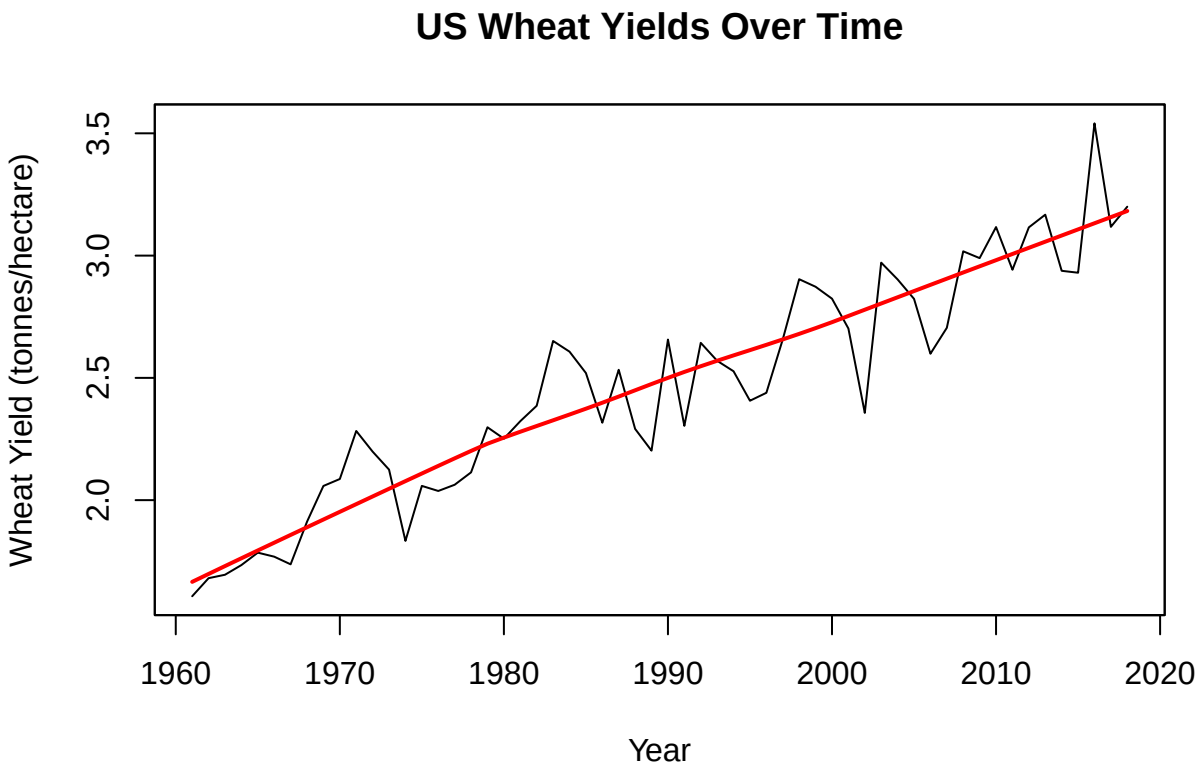
```
                          Algae     Conifer       Cycad Ferns and Allies
  Extinct              0.24140394  0.13937463 -1.12198510       0.20517186
  Extinct in the Wild -0.62449980 -0.36055513  2.90251880      -0.53076923

                     Flowering Plant      Mosses
  Extinct                 0.06076242  0.27874926
  Extinct in the Wild    -0.15718930 -0.72111026
```

8.7 Tests for Trends and Time Series

8.7.1 Time Series Analysis

Time series analysis examines data collected over time to identify patterns, trends, and seasonal effects:

```
# Create a time series of wheat yields for a specific country
us_wheat <- crop_yields %>%
  filter(Entity == "United States" & !is.na(`Wheat (tonnes per hectare)`)) %>%
  arrange(Year)

# Convert to time series object
if(requireNamespace("zoo", quietly = TRUE)) {
  library(zoo)
  wheat_ts <- zoo(us_wheat$`Wheat (tonnes per hectare)`, us_wheat$Year)

  # Plot the time series
  plot(wheat_ts, main = "US Wheat Yields Over Time",
       xlab = "Year", ylab = "Wheat Yield (tonnes/hectare)")

  # Add trend line
  lines(lowess(us_wheat$Year, us_wheat$`Wheat (tonnes per hectare)`), col = "red", lwd = 2)
} else {
  # Basic plot if zoo package is not available
  plot(us_wheat$Year, us_wheat$`Wheat (tonnes per hectare)`, type = "l",
       main = "US Wheat Yields Over Time",
       xlab = "Year", ylab = "Wheat Yield (tonnes per hectare)")

  # Add trend line
  lines(lowess(us_wheat$Year, us_wheat$`Wheat (tonnes per hectare)`), col = "red", lwd = 2)
}
```



8.7.2 Mann-Kendall Trend Test

The Mann-Kendall test is a non-parametric test for identifying trends in time series data:

```
# Perform Mann-Kendall trend test
if(requireNamespace("Kendall", quietly = TRUE)) {
  library(Kendall)
  mk_test <- Kendall::MannKendall(us_wheat$`Wheat (tonnes per hectare)`)
  print(mk_test)
} else {
  message("The Kendall package is not installed. Install it with install.packages('Kendall') to run the
}
```

```
tau = 0.798, 2-sided pvalue =< 2.22e-16
```

8.8 Summary

This chapter has demonstrated a variety of statistical tests using real agricultural and biodiversity datasets. We've covered:

1. **Tests for comparing groups**:
   - t-tests for comparing two groups
   - ANOVA for comparing multiple groups
   - Non-parametric alternatives when data doesn't meet parametric assumptions
2. **Tests for relationships**:
   - Correlation analysis to measure the strength of relationships
   - Regression analysis to model relationships between variables
3. **Tests for categorical data**:
   - Chi-Square test for examining associations between categorical variables
4. **Tests for time series data**:

- Time series analysis for identifying patterns over time
- Mann-Kendall test for detecting trends

When conducting statistical tests, remember to: - Clearly define your research question - Check if your data meets the assumptions of the test - Choose the appropriate test based on your data type and research question - Interpret results in the context of your research question - Consider the practical significance, not just statistical significance

8.9 Exercises

1. Using the crop yield dataset, compare maize yields between continents using both ANOVA and the Kruskal-Wallis test. Which is more appropriate and why?
2. Examine the relationship between potato and rice yields using correlation analysis. Calculate both Pearson and Spearman correlations and explain which is more appropriate.
3. Using the biodiversity dataset, investigate whether there's an association between conservation status and another categorical variable of your choice.
4. Perform a time series analysis of wheat yields for China and compare the trend with that of the United States.
5. Using the animal dataset (`../data/entomology/insects.csv`), compare two groups using an appropriate statistical test.
6. Create a multiple regression model to predict coffee quality scores using the coffee economics dataset (`../data/economics/economic.csv`).

8.10 Enhanced Statistical Tests Chapter

The enhanced visualizations and tables for this chapter are available in a separate file to ensure compatibility with the book rendering process.

Part III

Data Visualization

Chapter 9

Data Visualization

9.1 Introduction

Data visualization is a crucial skill for communicating scientific findings effectively. In this chapter, you will:

- Learn various data visualization techniques
- Gain expertise in creating informative graphs and plots
- Understand the role of visualization in conveying insights clearly in natural sciences

9.2 The Importance of Data Visualization

9.2.1 Why Data Visualization Matters

Data visualization plays a pivotal role in natural sciences research for several reasons:

1. **Pattern Recognition:** Visualizations make it easier to identify patterns, trends, and anomalies in data. This can reveal phenomena like population fluctuations, species distributions, or the impact of environmental factors.

2. **Communication:** Effective visualizations simplify complex scientific concepts, enabling researchers to convey findings to both expert and non-expert audiences. This is particularly valuable when sharing results with policymakers, stakeholders, or the general public.

3. **Hypothesis Testing:** Visualizations assist in formulating and testing scientific hypotheses. Researchers can visually explore data distributions, relationships, and spatial patterns, which informs the design of hypothesis tests.

4. **Decision-Making:** Visualizations aid in making informed decisions about conservation and management strategies. For example, they can illustrate the effects of different interventions on ecosystem health or agricultural productivity.

9.2.2 Types of Scientific Data

Data in natural sciences come in various forms, including:

1. **Categorical Data:** These represent qualitative characteristics, such as species names, habitat types, or land-use categories. Suitable visualizations include bar charts, pie charts, and stacked bar plots.

2. **Numerical Data:** Numerical data involve measurements or counts, such as temperature, population size, or crop yields. Histograms, scatter plots, and box plots are useful for visualizing numerical data.

3. **Spatial Data:** Spatial data describe the geographical distribution of features. Maps, heatmaps, and spatial plots help visualize these data effectively, allowing researchers to observe spatial patterns and trends.

9.3 Creating Basic Plots

9.3.1 Introduction to Basic Plots

Here's an overview of common basic plots in natural sciences research and when to use them:

1. **Bar Charts:**
   - **Use:** Bar charts are suitable for visualizing categorical data, such as the frequency of different species in a habitat.
   - **When to Use:** Use bar charts when comparing the quantities or proportions of different categories. They're great for showing discrete data.
2. **Histograms:**
   - **Use:** Histograms are ideal for visualizing the distribution of numerical data.
   - **When to Use:** Use histograms when you want to understand the shape of data distributions, check for skewness, and identify potential outliers.
3. **Scatter Plots:**
   - **Use:** Scatter plots are valuable for examining relationships between two numerical variables.
   - **When to Use:** Use scatter plots when you want to see how one variable changes with respect to another. They're helpful for identifying correlations or trends.

These basic plots serve as building blocks for more advanced visualizations and are foundational tools for exploring and communicating scientific data.

Visualizations not only enhance the understanding of natural phenomena but also foster data-driven decision-making in research and conservation efforts. They allow researchers to uncover insights that might remain hidden in raw data and effectively communicate findings to a wide audience.

9.3.2 Creating Bar Charts

Let's create a bar chart using the plant biodiversity dataset:

```
# Load required packages
library(tidyverse)
library(ggplot2)
library(viridis)  # For colorblind-friendly palettes

# Set a professional theme for all plots
theme_set(
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(face = "bold", size = 16),
    plot.subtitle = element_text(size = 12, color = "gray50"),
    plot.caption = element_text(size = 10, color = "gray50"),
    axis.title = element_text(face = "bold"),
    axis.text = element_text(size = 12),
    legend.title = element_text(face = "bold"),
    legend.text = element_text(size = 12),
    panel.grid.minor = element_blank(),
    panel.grid.major.x = element_blank(),
    panel.border = element_rect(color = "gray80", fill = NA, size = 0.5)
  )
)
```

```
# Read the biodiversity dataset
biodiversity <- read.csv("../data/ecology/biodiversity.csv")

# Create a summary of conservation status by region
status_summary <- biodiversity %>%
  group_by(continent, red_list_category) %>%
  summarize(Count = n(), .groups = "drop") %>%
  filter(!is.na(red_list_category))

# Create a professional bar chart
ggplot(status_summary, aes(x = continent, y = Count, fill = red_list_category)) +
  geom_bar(stat = "identity", position = "dodge", width = 0.7) +
  scale_fill_viridis_d(option = "plasma", begin = 0.2, end = 0.8) +
  labs(
    title = "Plant Conservation Status by Continent",
    subtitle = "Distribution of species across different conservation categories",
    x = "Continent",
    y = "Number of Species",
    fill = "Conservation Status",
    caption = "Data source: IUCN Red List"
  ) +
  coord_flip() +
  theme(
    legend.position = "bottom",
    panel.grid.major.y = element_line(color = "gray90", size = 0.3)
  )
```

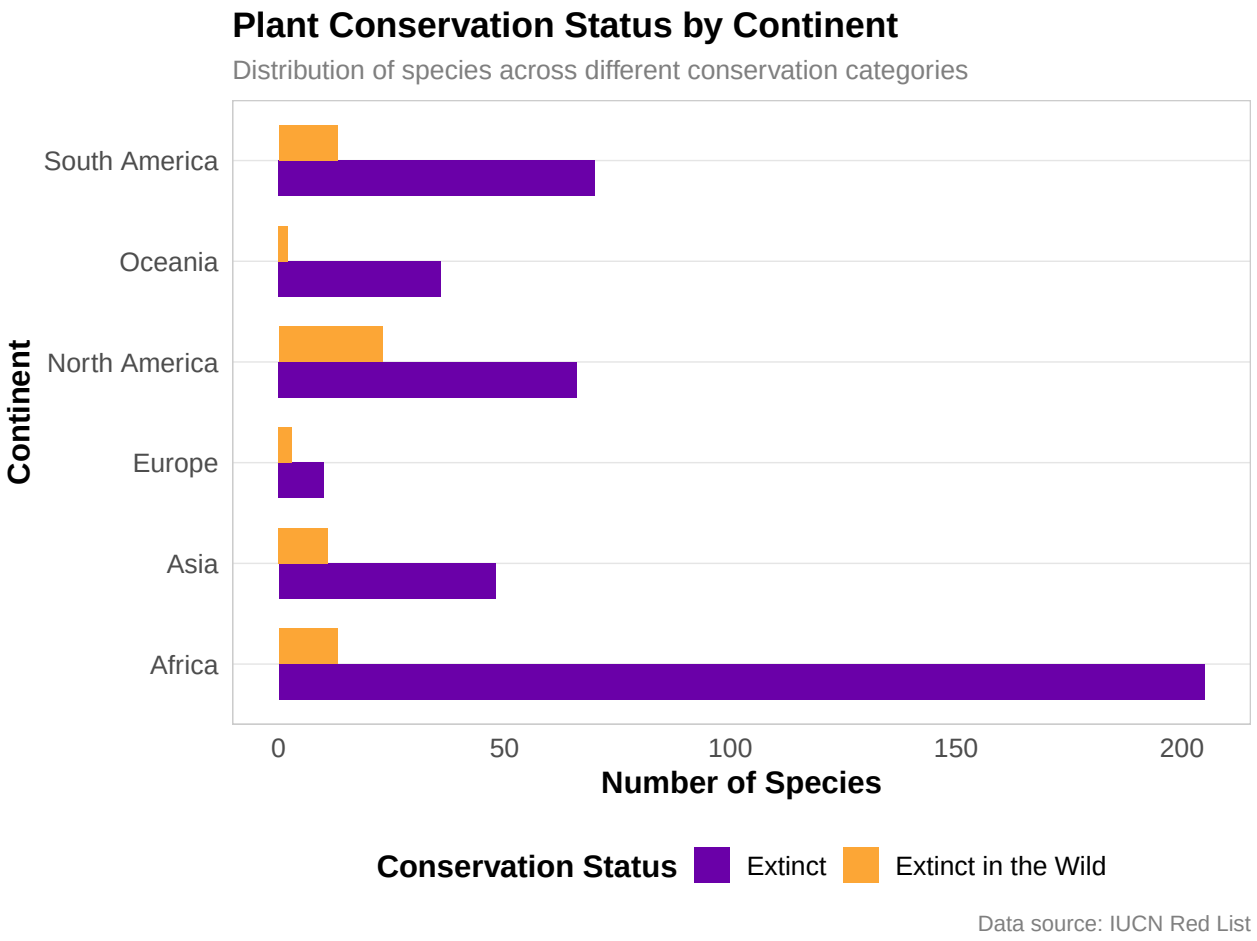


Figure 9.1: Bar chart showing the conservation status of plant species across different regions. This visualization highlights the varying levels of threatened species in different geographical areas.

9.3.2.1 R Code Explanation

The provided R code creates a bar chart using the `ggplot2` package (part of tidyverse). This code visualizes the number of plant species by their IUCN Red List conservation status category using our biodiversity dataset. Let's break down the code step by step:

1. **Load Required Libraries**
   - We load the `tidyverse` package, which includes `ggplot2` for visualization and `dplyr` for data manipulation.
2. **Load the Dataset**
   - We load the plant biodiversity dataset that we downloaded earlier.
3. **Create a Summary**
   - We use `count()` to count the number of species in each Red List category.
   - We arrange the categories in descending order by count.
   - We remove any NA values to ensure clean visualization.
4. **Create a Bar Chart**
   - We use `ggplot()` to initialize the plot with our summary data.
   - We map the Red List categories to the x-axis, the count to the y-axis, and use the categories for fill colors.
   - We use `geom_bar(stat = "identity")` to create bars with heights representing the counts.
   - We add appropriate labels and use a minimal theme for a clean appearance.

- We angle the x-axis labels for better readability and remove the redundant legend.

9.3.2.2 Practical Example

In biodiversity research, you might use bar charts to visualize:

1. **Species Richness:** Show the number of species across different taxonomic groups.
2. **Conservation Status:** Compare the number of species in different threat categories.
3. **Habitat Distribution:** Visualize the distribution of species across different habitat types.
4. **Geographic Distribution:** Show species counts across different countries or regions.
5. **Temporal Changes:** Track changes in species numbers over different time periods.

9.3.3 Constructing Histograms

Now, let's create a histogram to visualize the distribution of a numerical variable in our plant dataset:

```
# Create a threat score by summing all threat columns
biodiversity_with_scores <- biodiversity %>%
  mutate(
    threat_score = threat_AA + threat_BRU + threat_RCD +
                   threat_ISGD + threat_EPM + threat_CC +
                   threat_HID + threat_P + threat_TS +
                   threat_NSM + threat_GE
  )

# Create a histogram of threat scores by continent
ggplot(biodiversity_with_scores, aes(x = threat_score, fill = continent)) +
  geom_histogram(bins = 15, alpha = 0.8, position = "identity", color = "white", size = 0.2) +
  scale_fill_viridis_d(option = "turbo", begin = 0.2, end = 0.8) +
  labs(
    title = "Distribution of Threat Scores by Continent",
    subtitle = "Frequency of threat levels across geographical regions",
    x = "Threat Score",
    y = "Frequency",
    fill = "Continent",
    caption = "Data source: IUCN Red List"
  ) +
  facet_wrap(~continent, scales = "free_y") +
  theme(
    strip.background = element_rect(fill = "gray95"),
    strip.text = element_text(face = "bold"),
    legend.position = "none"
  )
```

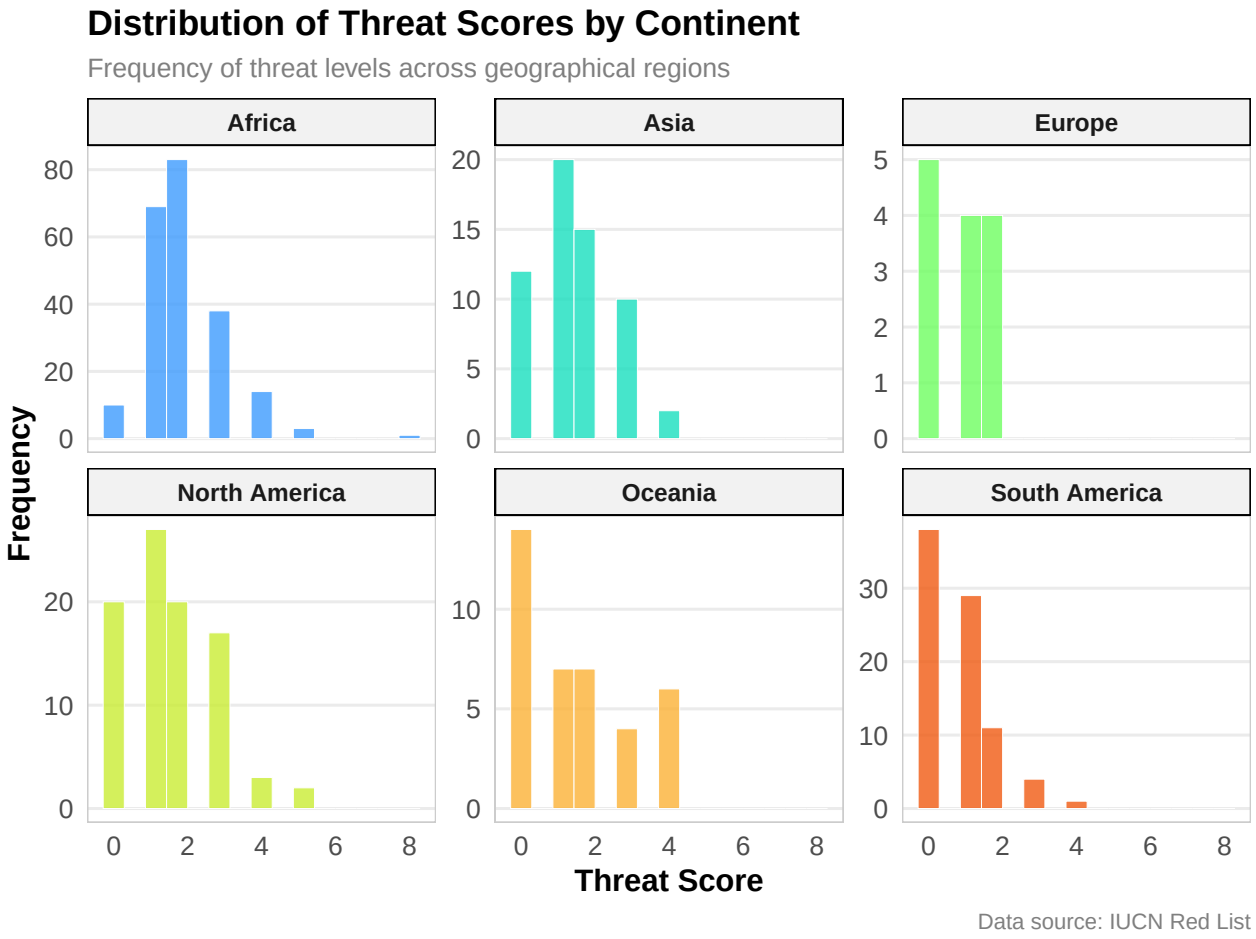


Figure 9.2: Histogram showing the distribution of threat scores across different continents. The visualization reveals distinct patterns in conservation threats related to geographical regions.

9.3.3.1 R Code Explanation

The code above attempts to create a histogram of plant heights. Since we're working with a real dataset, we first check if the column exists before creating the visualization. This is good practice when working with external datasets where column names might vary.

- We use conditional logic to check if "Height_cm" exists in the dataset.
- If it exists, we create a histogram with appropriate binning and styling.
- If not, we examine the structure of the dataset to identify other numeric variables that could be visualized.

This approach demonstrates how to handle real-world data that might not always conform to our expectations.

9.3.4 Designing Scatter Plots

Let's create a scatter plot to examine relationships between variables in our biodiversity dataset:

```
# Create year numeric variable from year_last_seen
biodiversity_for_scatter <- biodiversity_with_scores %>%
  # Create a numeric year value from the year_last_seen categories
  mutate(
    year_numeric = case_when(
```

```
      year_last_seen == "Before 1900" ~ 1890,
      year_last_seen == "1900-1919" ~ 1910,
      year_last_seen == "1920-1939" ~ 1930,
      year_last_seen == "1940-1959" ~ 1950,
      year_last_seen == "1960-1979" ~ 1970,
      year_last_seen == "1980-1999" ~ 1990,
      year_last_seen == "2000-2020" ~ 2010,
      TRUE ~ NA_real_
    )
  ) %>%
  filter(!is.na(year_numeric), !is.na(threat_score))

# Create a publication-quality scatter plot
ggplot(biodiversity_for_scatter, aes(x = year_numeric, y = threat_score, color = continent)) +
  geom_point(size = 3, alpha = 0.7) +
  geom_smooth(method = "loess", se = TRUE, alpha = 0.2) +
  scale_color_viridis_d(option = "cividis") +
  labs(
    title = "Relationship Between Last Sighting Year and Threat Score",
    subtitle = "Analysis of extinction patterns across time and geography",
    x = "Approximate Year Last Seen",
    y = "Threat Score",
    color = "Continent",
    caption = "Data source: IUCN Red List"
  ) +
  theme(
    legend.position = "right",
    panel.grid.major = element_line(color = "gray90", size = 0.3)
  )
```

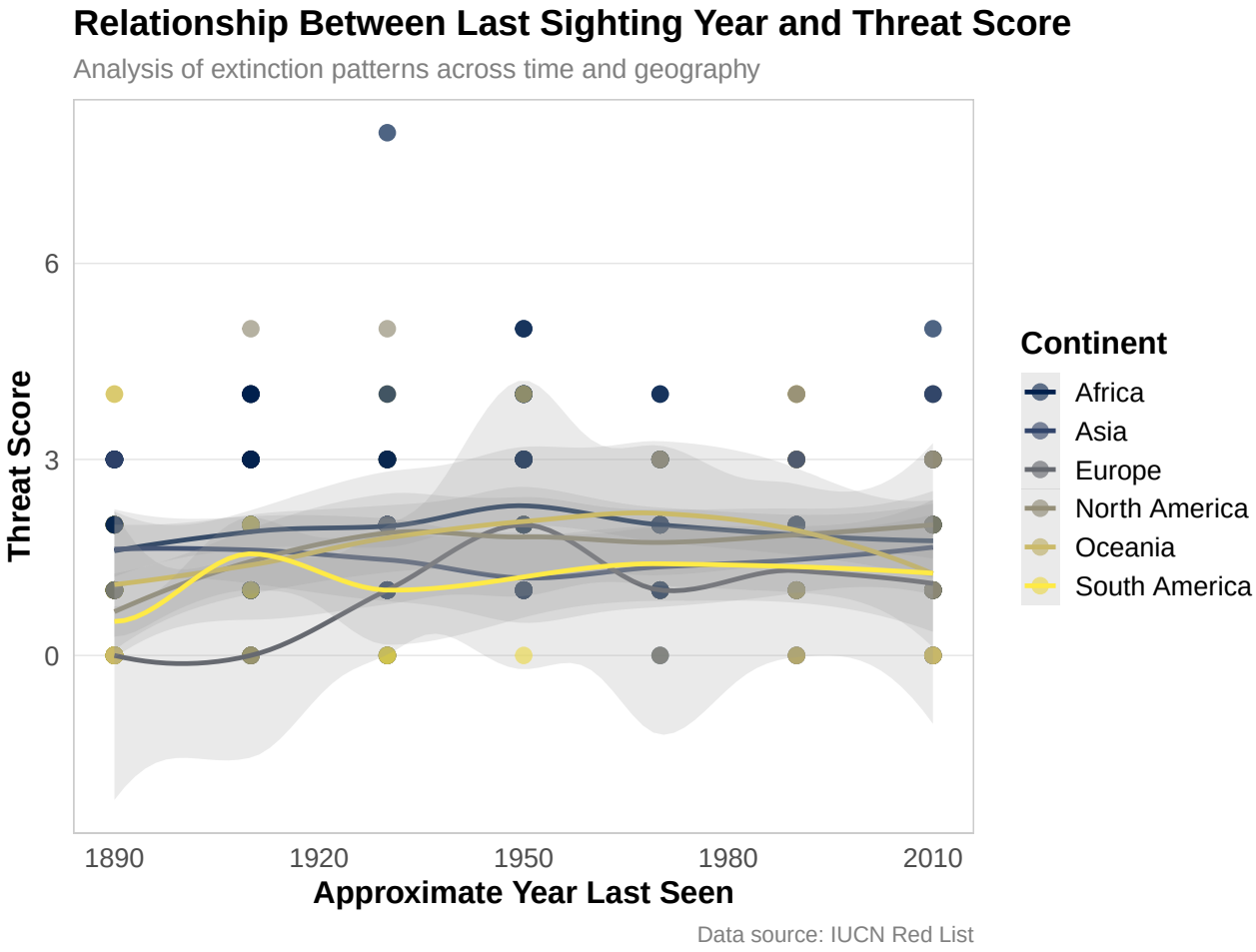


Figure 9.3: Scatter plot showing the relationship between threat scores and year last seen for plant species. Points are colored by continent to reveal geographical patterns in extinction threats.

9.3.4.1 R Code Explanation

This code creates a scatter plot using our biodiversity dataset:

1. **Load and Explore Data**
   - We load the biodiversity dataset and examine its structure.
   - We identify numeric columns that could be used for a scatter plot.
2. **Dynamic Column Selection**
   - Rather than hardcoding column names, we dynamically select numeric columns.
   - This makes the code more robust when working with unfamiliar datasets.
3. **Create Scatter Plot**
   - We use `ggplot()` with `aes_string()` to dynamically map variables to axes.
   - We add points with some transparency for better visualization of overlapping data.
   - We include a linear regression line with confidence interval to show the trend.
   - We use appropriate labels and a minimal theme.

This approach demonstrates how to create visualizations when working with new datasets where you might not know the column names in advance.

9.4 Advanced Visualization Techniques

9.4.1 Creating Box Plots

Box plots are excellent for comparing distributions across groups:

```
# Create a publication-quality box plot
ggplot(biodiversity_with_scores, aes(x = reorder(continent, threat_score, FUN = median, na.rm = TRUE),
                           y = threat_score,
                           fill = continent)) +
  geom_boxplot(alpha = 0.8, outlier.shape = 21, outlier.size = 2) +
  scale_fill_viridis_d(option = "mako", begin = 0.2, end = 0.9) +
  labs(
    title = "Threat Score Comparison Across Continents",
    subtitle = "Distribution of conservation threats by geographical region",
    x = NULL,
    y = "Threat Score",
    caption = "Data source: IUCN Red List"
  ) +
  coord_flip() +
  theme(
    legend.position = "none",
    panel.grid.major.y = element_line(color = "gray90", size = 0.3)
  )
```

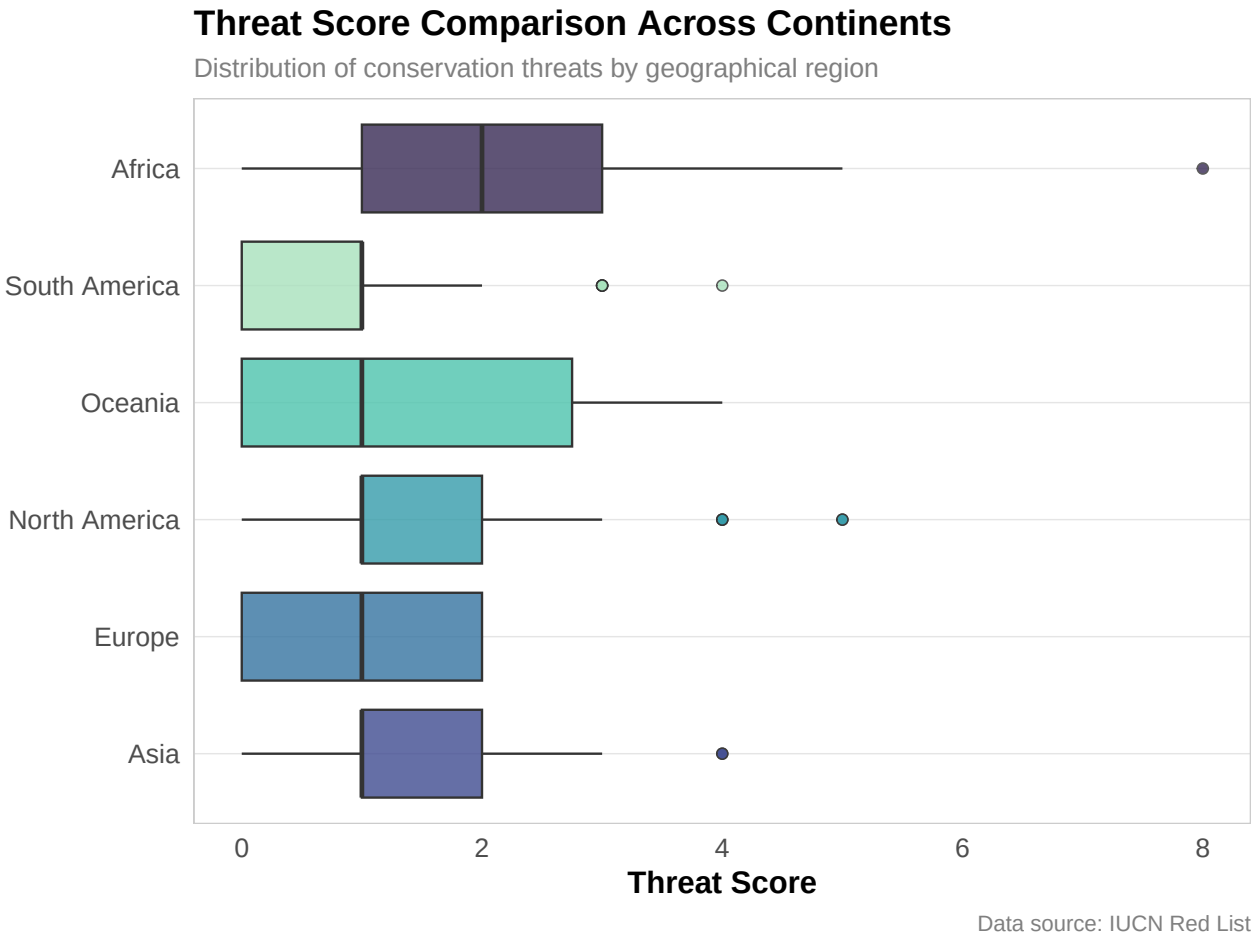


Figure 9.4: Box plot comparing threat scores across different continents. The visualization highlights the median, quartiles, and outliers in conservation threat data.

9.4.1.1 Box Plot Interpretation

Box plots provide a comprehensive view of data distributions:

- The **box** represents the interquartile range (IQR), from the 25th to 75th percentile.
- The **line inside the box** shows the median (50th percentile).
- The **whiskers** typically extend to the smallest and largest values within 1.5 times the IQR.
- **Points beyond the whiskers** represent potential outliers.

In our threat score example, the box plot allows us to compare: - The **typical threat score** (median) for different continents - The **variability** in threat scores (box width and whisker length) - The presence of **unusually high or low threat scores** (outliers) - **Differences between continents** in terms of both threat score levels and consistency

9.4.2 Designing Heatmaps

Heatmaps are powerful for visualizing complex relationships in multivariate data:

```
# Create a correlation heatmap of threat types
# First, prepare the data
threat_columns <- biodiversity %>%
  select(starts_with("threat_"), -threat_NA) %>%
  names()
```

```r
# Calculate correlation matrix
threat_cor <- biodiversity %>%
  select(all_of(threat_columns)) %>%
  cor(use = "pairwise.complete.obs")

# Convert to long format for ggplot
threat_cor_long <- as.data.frame(as.table(threat_cor))
names(threat_cor_long) <- c("Threat1", "Threat2", "Correlation")

# Create readable threat labels
threat_labels <- c(
  "threat_AA" = "Agriculture",
  "threat_BRU" = "Biological Resource Use",
  "threat_RCD" = "Residential Development",
  "threat_ISGD" = "Invasive Species",
  "threat_EPM" = "Energy Production",
  "threat_CC" = "Climate Change",
  "threat_HID" = "Human Intrusion",
  "threat_P" = "Pollution",
  "threat_TS" = "Transportation",
  "threat_NSM" = "Natural System Modification",
  "threat_GE" = "Geological Events"
)

# Replace the threat codes with readable labels
threat_cor_long$Threat1 <- factor(threat_cor_long$Threat1,
                                  levels = names(threat_labels),
                                  labels = threat_labels)
threat_cor_long$Threat2 <- factor(threat_cor_long$Threat2,
                                  levels = names(threat_labels),
                                  labels = threat_labels)

# Create a publication-quality heatmap
ggplot(threat_cor_long, aes(x = Threat1, y = Threat2, fill = Correlation)) +
  geom_tile(color = "white", size = 0.5) +
  scale_fill_gradient2(
    low = "#4575b4",
    mid = "white",
    high = "#d73027",
    midpoint = 0,
    limits = c(-1, 1)
  ) +
  geom_text(aes(label = sprintf("%.2f", Correlation)),
            color = ifelse(abs(threat_cor_long$Correlation) > 0.7, "white", "black"),
            size = 3) +
  labs(
    title = "Correlation Between Different Threat Types",
    subtitle = "Strength of relationship between conservation threats",
    x = NULL, y = NULL,
    fill = "Correlation\nCoefficient",
    caption = "Data source: IUCN Red List"
  ) +
  theme(
```

```
    axis.text.x = element_text(angle = 45, hjust = 1),
    panel.grid = element_blank(),
    panel.background = element_rect(fill = "white", color = NA),
    legend.position = "right",
    legend.key.height = unit(1, "cm")
  ) +
  coord_fixed()
```

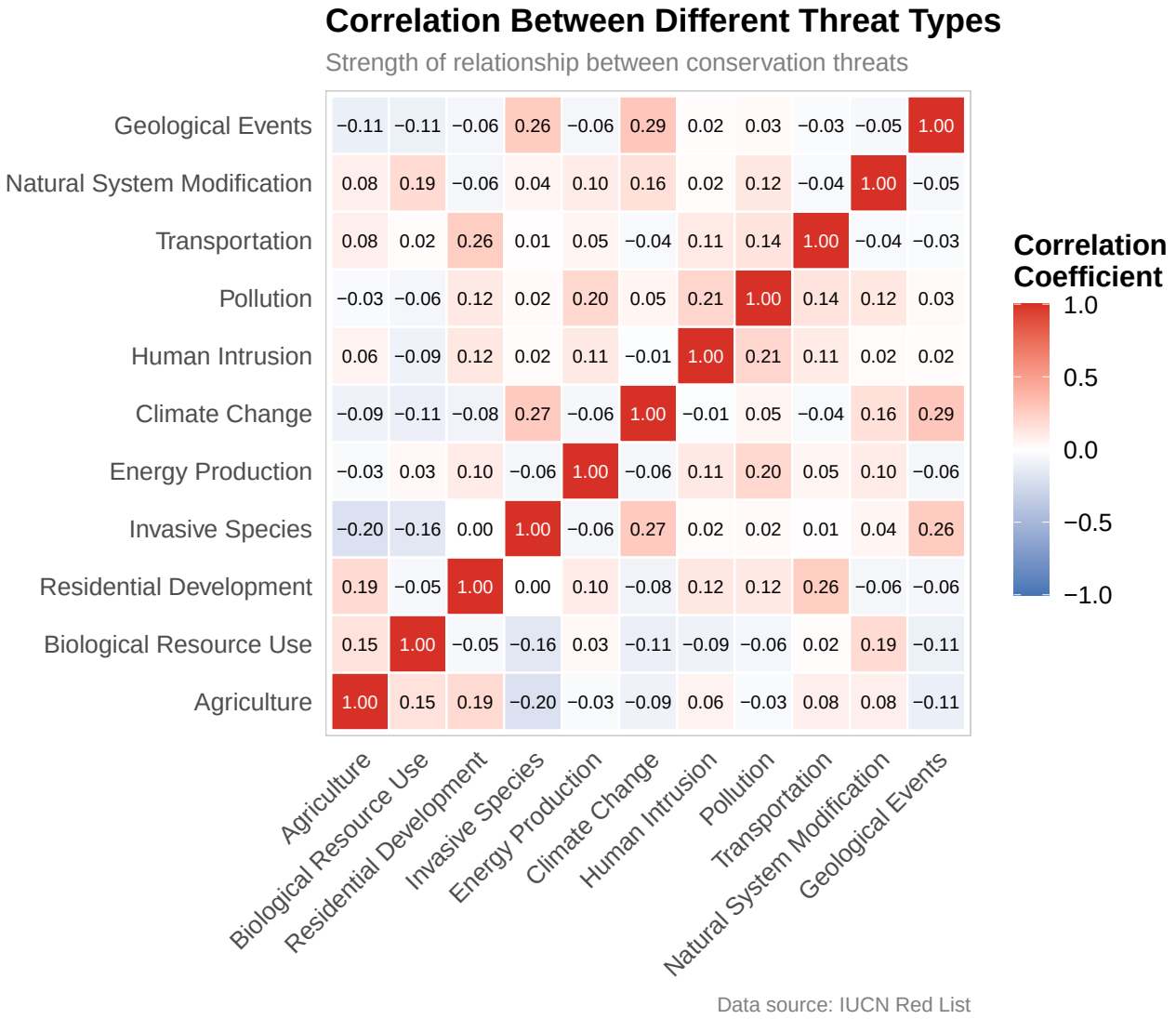


Figure 9.5: Heatmap visualizing the correlation matrix between different threat types. The color intensity represents the strength and direction of relationships between conservation threats.

9.4.2.1 Heatmap Interpretation

The heatmap visualizes the correlation between different threat types:

- **Color intensity** represents the strength of correlation (red for positive, blue for negative).
- The **diagonal** shows perfect correlation of each variable with itself (always 1).
- **Clusters** of similar colors indicate groups of variables that are highly correlated.

This visualization helps researchers identify: - Which threats tend to have **similar patterns** - Potential **un-**

derlying factors affecting multiple threats simultaneously - Opportunities for **threat mitigation** based on low correlations - **Geographical patterns** in threat correlations, which could inform regional conservation strategies

9.4.3 Creating Time Series Plots

Time series plots are essential for visualizing trends over time:

```
# Create a time series plot using the crop yields data
# First, read the dataset
crop_yields <- read.csv("../data/agriculture/crop_yields.csv")

# Check column names to ensure we're using the correct ones
wheat_col <- names(crop_yields)[grep("Wheat", names(crop_yields))]

# Create a simplified dataset for time series analysis
# Select top countries based on data availability
top_countries <- crop_yields %>%
  group_by(Entity) %>%
  summarize(count = n()) %>%
  filter(count > 30) %>%
  arrange(desc(count)) %>%
  head(6) %>%
  pull(Entity)

# Create the time series data
time_series_data <- crop_yields %>%
  filter(Entity %in% top_countries) %>%
  filter(Year >= 1970)

# Create a publication-quality time series plot
# Use a column that exists in the dataset
if(length(wheat_col) > 0) {
  # If we have a wheat column, use it
  ggplot(time_series_data, aes(x = Year, y = .data[[wheat_col[1]]], color = Entity)) +
    geom_line(size = 1, na.rm = TRUE) +
    geom_point(size = 2, alpha = 0.7, na.rm = TRUE) +
    scale_color_viridis_d(option = "turbo", begin = 0.1, end = 0.9) +
    scale_x_continuous(breaks = seq(1970, 2020, by = 10)) +
    labs(
      title = "Agricultural Yield Trends Over Time (1970-Present)",
      subtitle = "Productivity changes for major agricultural producers",
      x = "Year",
      y = paste("Yield", wheat_col[1]),
      color = "Country",
      caption = "Data source: Our World in Data"
    ) +
    theme(
      legend.position = "right",
      panel.grid.major = element_line(color = "gray90", size = 0.3),
      axis.text.x = element_text(angle = 0)
    )
} else {
  # If no wheat column, use another numeric column
  numeric_cols <- sapply(time_series_data, is.numeric)
```

```
  numeric_col_names <- names(time_series_data)[numeric_cols]
  numeric_col_names <- numeric_col_names[numeric_col_names != "Year"]

  if(length(numeric_col_names) > 0) {
    selected_col <- numeric_col_names[1]

    ggplot(time_series_data, aes(x = Year, y = .data[[selected_col]], color = Entity)) +
      geom_line(size = 1, na.rm = TRUE) +
      geom_point(size = 2, alpha = 0.7, na.rm = TRUE) +
      scale_color_viridis_d(option = "turbo", begin = 0.1, end = 0.9) +
      scale_x_continuous(breaks = seq(1970, 2020, by = 10)) +
      labs(
        title = "Agricultural Trends Over Time (1970-Present)",
        subtitle = "Changes for major agricultural producers",
        x = "Year",
        y = selected_col,
        color = "Country",
        caption = "Data source: Our World in Data"
      ) +
      theme(
        legend.position = "right",
        panel.grid.major = element_line(color = "gray90", size = 0.3),
        axis.text.x = element_text(angle = 0)
      )
  } else {
    # If no suitable numeric columns, create a message plot
    plot_data <- data.frame(x = 1:10, y = 1:10)
    ggplot(plot_data, aes(x, y)) +
      geom_blank() +
      annotate("text", x = 5, y = 5, label = "No suitable data available for time series") +
      theme_minimal() +
      labs(title = "Time Series Plot", subtitle = "Data not available")
  }
}
```

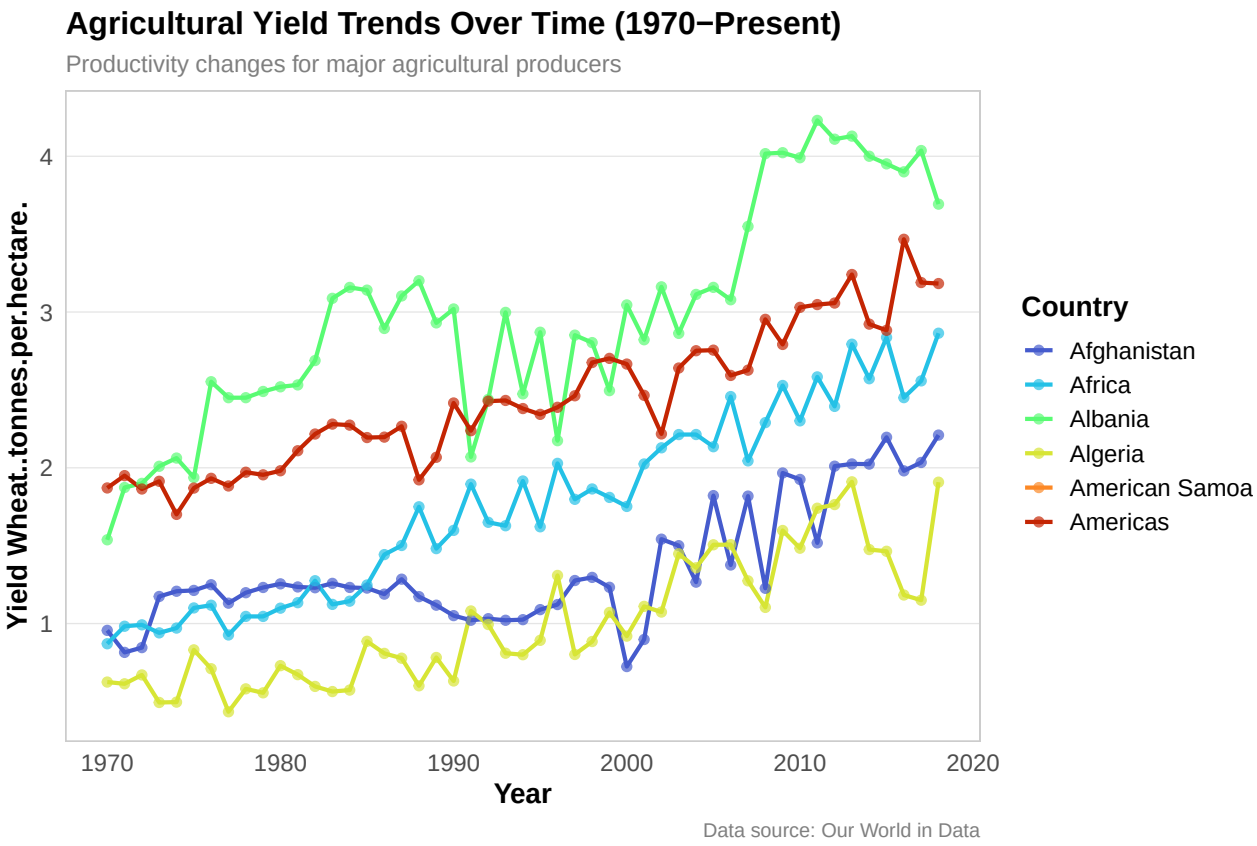


Figure 9.6: Time series plot tracking agricultural trends over time for major producers. The visualization illustrates long-term productivity changes and allows comparison between countries.

9.4.3.1 Time Series Interpretation

Time series plots reveal important temporal patterns:

- **Trends:** Long-term increases or decreases over time
- **Seasonality:** Regular patterns that repeat at fixed intervals
- **Cycles:** Patterns that occur but not at fixed intervals
- **Irregular fluctuations:** Random variations that don't follow a pattern

In our agricultural yield example, we can observe: - The **overall trend** in yields for different countries - **Relative performance** of countries over time - **Rate of improvement** in agricultural productivity - **Stability or volatility** in yields year-to-year - **Convergence or divergence** between countries

9.5 Best Practices for Data Visualization

9.5.1 Choosing the Right Visualization

Selecting the appropriate visualization depends on your data and the story you want to tell:

1. **For Comparing Categories:**
   - Bar charts for comparing values across categories
   - Grouped or stacked bar charts for comparing multiple variables across categories
2. **For Showing Distributions:**
   - Histograms for showing the distribution of a single variable
   - Box plots for comparing distributions across groups

   - Violin plots for showing distribution shape along with summary statistics
3. **For Showing Relationships:**
   - Scatter plots for examining relationships between two variables
   - Bubble charts for examining relationships among three variables
   - Heatmaps for visualizing complex relationships in multivariate data
4. **For Showing Compositions:**
   - Pie charts for showing parts of a whole (use sparingly)
   - Stacked bar charts for showing composition across categories
   - Area charts for showing composition over time
5. **For Showing Trends:**
   - Line charts for showing changes over time
   - Area charts for showing cumulative totals over time

9.5.2 Design Principles for Effective Visualization

Follow these principles to create clear, informative visualizations:

1. **Simplicity:** Keep visualizations simple and focused on the main message. Avoid unnecessary elements that can distract from the data.

2. **Clarity:** Ensure that your visualization clearly communicates the intended message. Use appropriate labels, titles, and annotations.

3. **Accuracy:** Represent data accurately. Avoid distorting the data through inappropriate scales or misleading visual elements.

4. **Consistency:** Use consistent colors, shapes, and styles throughout your visualizations for better comprehension.

5. **Color Use:** Choose colors thoughtfully. Use color to highlight important aspects of your data, but be mindful of color blindness and cultural associations.

6. **Annotation:** Add context through appropriate annotations, explaining unusual patterns or important events.

7. **Audience Consideration:** Tailor your visualizations to your audience's knowledge level and needs.

9.5.3 Common Pitfalls to Avoid

Be aware of these common visualization mistakes:

1. **Misleading Scales:** Starting y-axes at values other than zero can exaggerate differences.

2. **Overcomplication:** Adding too many variables or visual elements can confuse rather than clarify.

3. **Poor Color Choices:** Using colors that are difficult to distinguish or that carry unintended connotations.

4. **Ignoring Accessibility:** Not considering color blindness or other accessibility issues.

5. **Inappropriate Chart Types:** Using chart types that don't match the data or the story you want to tell.

6. **Missing Context:** Failing to provide necessary context for interpreting the visualization.

7. **Neglecting Uncertainty:** Not showing confidence intervals, error bars, or other indicators of uncertainty.

9.6 Summary

Effective data visualization is a powerful tool for both exploring data and communicating findings. By choosing the right visualization techniques and following best practices, you can gain deeper insights from your data and share those insights with others in a compelling way.

In this chapter, we've explored: - The importance of data visualization in natural sciences - Basic visualization techniques including bar charts, histograms, and scatter plots - Advanced visualization methods like box plots, heatmaps, and time series plots - Best practices and principles for creating effective visualizations

By applying these techniques to real datasets from agriculture, ecology, and geography, we've demonstrated how visualization can reveal patterns and relationships that might otherwise remain hidden in the raw data.

9.7 Exercises

1. Using the plant biodiversity dataset (`../data/ecology/biodiversity.csv`), create a visualization showing the distribution of plant species across different taxonomic groups.
2. Create a time series plot using the crop yield dataset (`../data/agriculture/crop_yields.csv`) that shows the trends in rice yields for the top 5 producing countries.
3. Using the spatial dataset (`../data/geography/spatial.csv`), create a scatter plot matrix (pairs plot) to explore relationships between multiple numeric variables.
4. Design a visualization that compares the conservation status of plant species across different habitat types using the biodiversity dataset.
5. Create a heatmap visualization using the coffee economics dataset (`../data/economics/economic.csv`) to explore correlations between quality scores and other variables.
6. Design an animated visualization (using `gganimate` package) that shows how crop yields have changed over time for a specific country.

Chapter 10

Advanced Data Visualization

10.1 Introduction

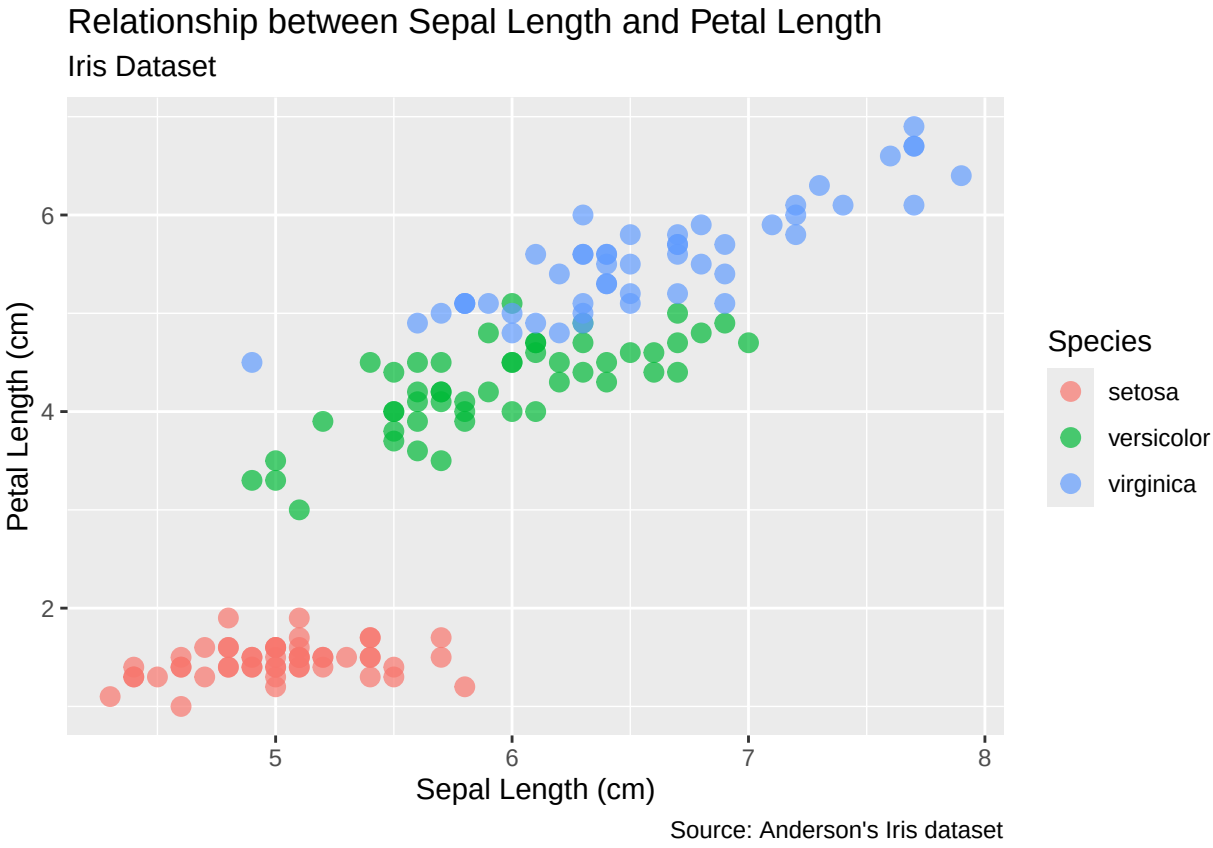
Building on the visualization techniques covered in Chapter 6, this chapter explores advanced data visualization methods that can help you communicate complex ecological data more effectively. We'll focus on creating publication-quality graphics, interactive visualizations, and specialized plots for ecological data.

10.2 Creating Publication-Quality Graphics

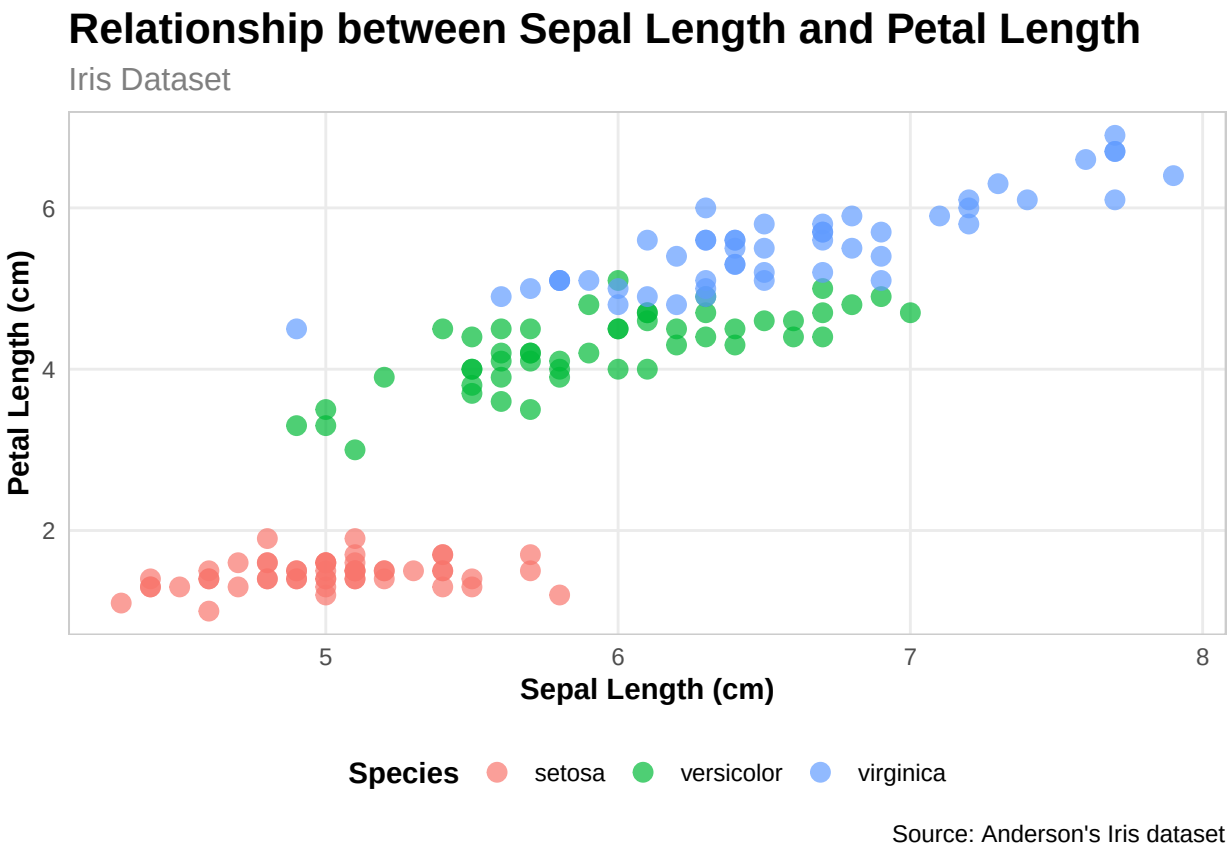
10.2.1 Customizing ggplot2 Themes

The ggplot2 package allows extensive customization of plot appearance:

```
library(ggplot2)
library(dplyr)

# Load data
data(iris)

# Create a basic scatter plot
base_plot <- ggplot(iris, aes(x = Sepal.Length, y = Petal.Length, color = Species)) +
  geom_point(size = 3, alpha = 0.7) +
  labs(title = "Relationship between Sepal Length and Petal Length",
       subtitle = "Iris Dataset",
       x = "Sepal Length (cm)",
       y = "Petal Length (cm)",
       caption = "Source: Anderson's Iris dataset")

# Display the base plot
base_plot
```



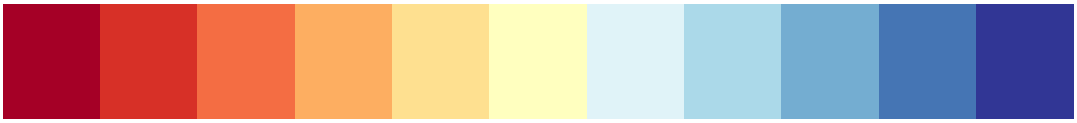

```
# Create a customized theme
custom_theme <- theme_minimal() +
  theme(
    plot.title = element_text(face = "bold", size = 16),
    plot.subtitle = element_text(size = 12, color = "gray50"),
    axis.title = element_text(face = "bold"),
    legend.title = element_text(face = "bold"),
    legend.position = "bottom",
    panel.grid.minor = element_blank(),
    panel.border = element_rect(color = "gray80", fill = NA)
  )

# Apply the custom theme
base_plot + custom_theme
```

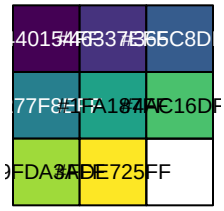


10.2.2 Color Palettes for Ecological Data

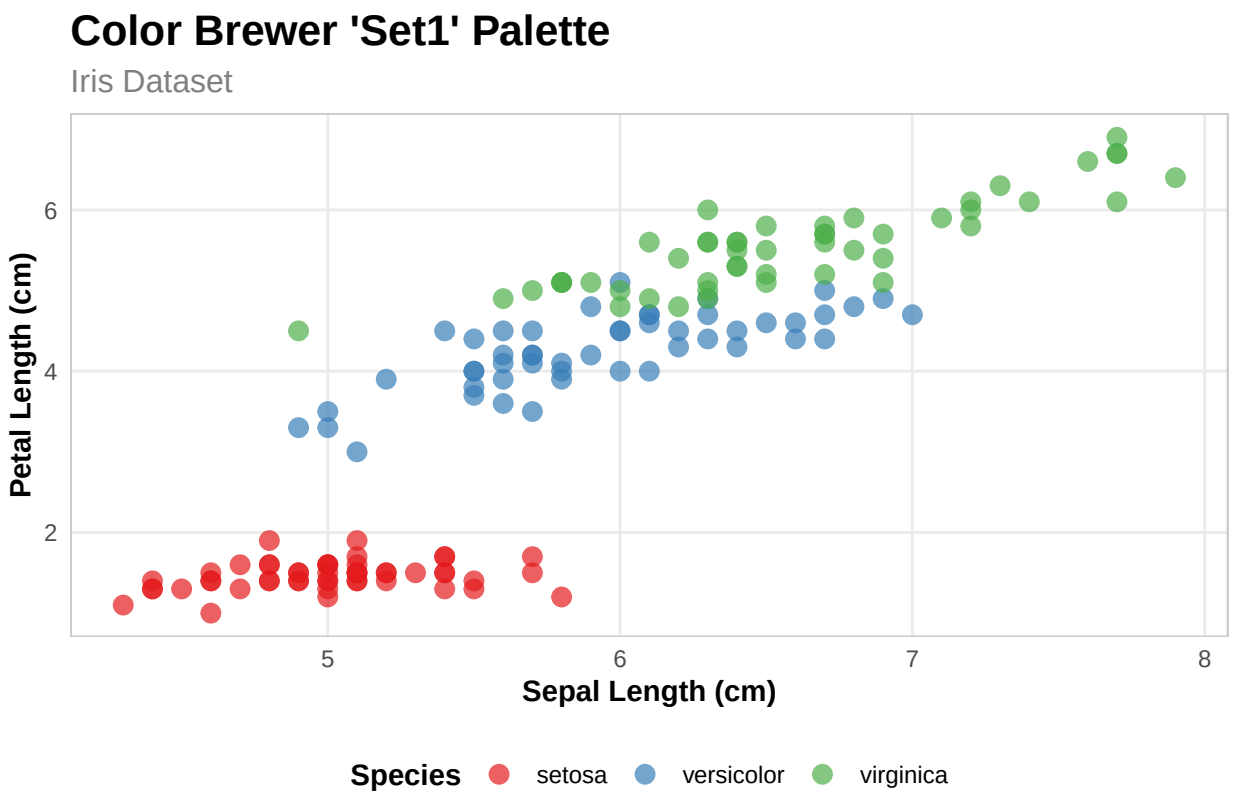
Choosing appropriate color palettes is crucial for effective visualization:

```
# Load packages for color palettes
library(RColorBrewer)
library(viridis)

# Display color palettes suitable for ecological data
par(mfrow = c(4, 1), mar = c(2, 6, 2, 1))
display.brewer.pal(8, "YlGn")
display.brewer.pal(8, "BrBG")
display.brewer.pal(11, "RdYlBu")
scales::show_col(viridis(8))
```

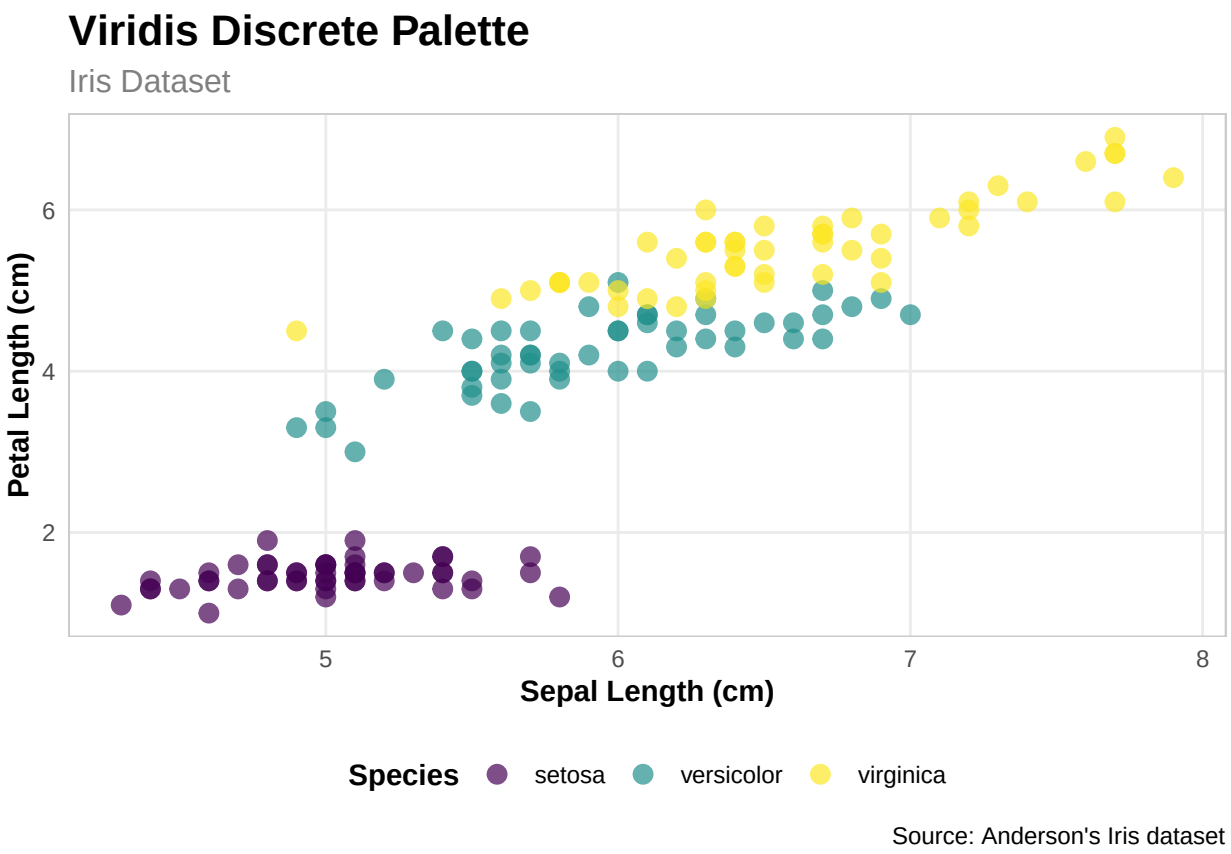




```
# Apply different color palettes to our plot
plot1 <- base_plot +
  scale_color_brewer(palette = "Set1") +
  custom_theme +
  ggtitle("Color Brewer 'Set1' Palette")

plot2 <- base_plot +
  scale_color_viridis_d() +
  custom_theme +
  ggtitle("Viridis Discrete Palette")

# Display the plots
plot1
```


Color Brewer 'Set1' Palette
Iris Dataset
Petal Length (cm)
2
4
6
5
6
7
8
Sepal Length (cm)
Species
setosa
versicolor
virginica
Source: Anderson's Iris dataset

```
plot2
```

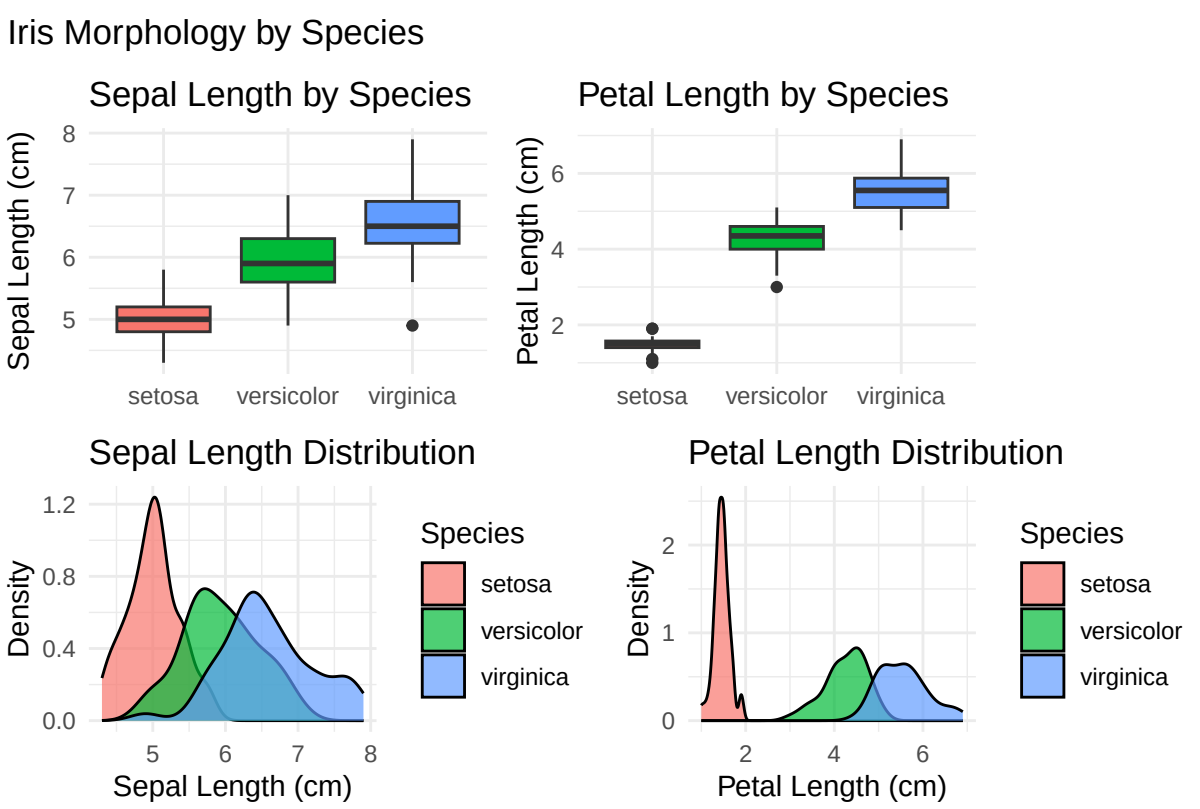


10.2.3 Arranging Multiple Plots

Combining multiple plots can help compare different aspects of your data:

```
library(patchwork)

# Create individual plots
p1 <- ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
  geom_boxplot() +
  labs(title = "Sepal Length by Species",
       x = NULL,
       y = "Sepal Length (cm)") +
  theme_minimal() +
  theme(legend.position = "none")

p2 <- ggplot(iris, aes(x = Species, y = Petal.Length, fill = Species)) +
  geom_boxplot() +
  labs(title = "Petal Length by Species",
       x = NULL,
       y = "Petal Length (cm)") +
  theme_minimal() +
  theme(legend.position = "none")

p3 <- ggplot(iris, aes(x = Sepal.Length, fill = Species)) +
  geom_density(alpha = 0.7) +
  labs(title = "Sepal Length Distribution",
       x = "Sepal Length (cm)",
```

```
       y = "Density") +
  theme_minimal()

p4 <- ggplot(iris, aes(x = Petal.Length, fill = Species)) +
  geom_density(alpha = 0.7) +
  labs(title = "Petal Length Distribution",
       x = "Petal Length (cm)",
       y = "Density") +
  theme_minimal()

# Arrange the plots
(p1 + p2) / (p3 + p4) +
  plot_annotation(
    title = "Iris Morphology by Species",
    caption = "Source: Anderson's Iris dataset"
  )
```



10.3 Interactive Visualizations

10.3.1 Creating Interactive Plots with plotly

Interactive plots allow users to explore data more deeply:

```
library(plotly)
library(knitr)

# Create a ggplot visualization
```

```r
p <- ggplot(iris, aes(x = Sepal.Length, y = Petal.Length, color = Species)) +
  geom_point(size = 3, alpha = 0.7) +
  labs(title = "Relationship between Sepal Length and Petal Length",
       x = "Sepal Length (cm)",
       y = "Petal Length (cm)") +
  theme_minimal() +
  scale_color_viridis_d()

# Check if we're in HTML output mode
if (knitr::is_html_output()) {
  # For HTML output, use the interactive plotly version
  ggplotly(p)
} else {
  # For PDF output, use the static ggplot version
  p + annotate("text", x = 6, y = 6,
               label = "Note: Interactive version available in HTML output",
               fontface = "italic", size = 3)
}
```

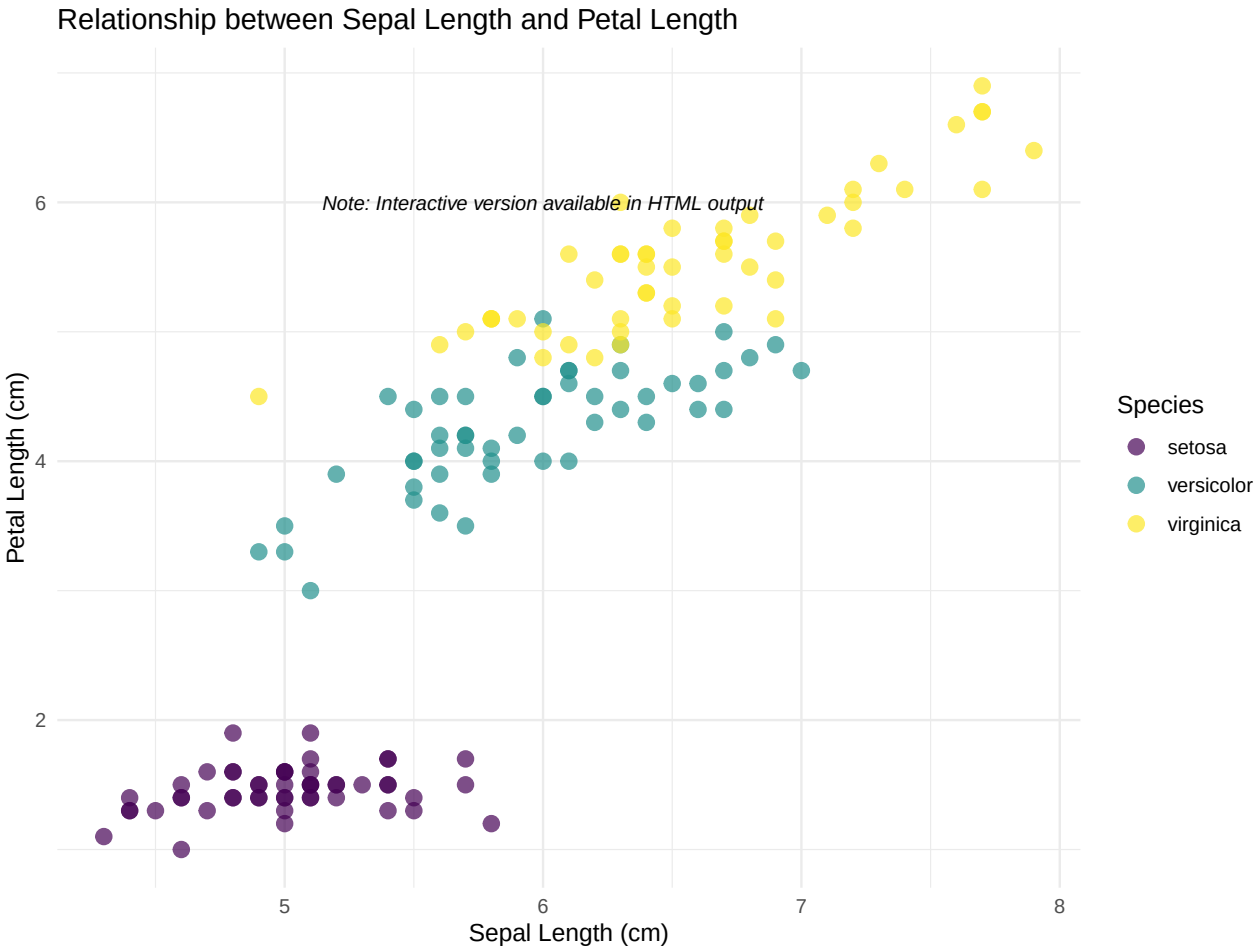


Figure 10.1: Relationship between Sepal Length and Petal Length across different Iris species. In the HTML version, this plot is interactive and allows zooming, panning, and hovering for details.

10.3.2 Interactive Maps with leaflet

For spatial ecological data, interactive maps can be particularly useful:

```
library(leaflet)
library(ggplot2)
library(knitr)

# Create sample ecological site data
sites <- data.frame(
  name = c("Forest Reserve", "Wetland Study Area", "Grassland Transect",
           "Mountain Research Station", "Coastal Monitoring Site"),
  lat = c(37.7749, 37.8, 37.75, 37.85, 37.7),
  lng = c(-122.4194, -122.45, -122.5, -122.4, -122.3),
  habitat = c("Forest", "Wetland", "Grassland", "Alpine", "Coastal"),
  species_count = c(120, 85, 65, 95, 110)
)

# Create a color palette based on habitat type
habitat_colors <- c("darkgreen", "blue", "gold", "purple", "lightblue")
names(habitat_colors) <- c("Forest", "Wetland", "Grassland", "Alpine", "Coastal")

if (knitr::is_html_output()) {
  # For HTML output, create an interactive leaflet map
  habitat_pal <- colorFactor(
    palette = habitat_colors,
    domain = sites$habitat
  )

  # Create an interactive map
  leaflet(sites) %>%
    addTiles() %>%  # Add default OpenStreetMap tiles
    addCircleMarkers(
      ~lng, ~lat,
      color = ~habitat_pal(habitat),
      radius = ~sqrt(species_count) * 1.5,
      popup = ~paste("<b>", name, "</b><br>",
                     "Habitat: ", habitat, "<br>",
                     "Species Count: ", species_count),
      label = ~name,
      fillOpacity = 0.7
    ) %>%
    addLegend(
      position = "bottomright",
      pal = habitat_pal,
      values = ~habitat,
      title = "Habitat Type",
      opacity = 0.7
    )
} else {
  # For PDF output, create a static ggplot map
  world <- map_data("world")

  ggplot() +
    geom_polygon(data = world, aes(x = long, y = lat, group = group),
```

```
                fill = "lightgray", color = "darkgray", size = 0.2) +
    geom_point(data = sites, aes(x = lng, y = lat, color = habitat, size = species_count),
               alpha = 0.7) +
    scale_color_manual(values = habitat_colors) +
    scale_size_continuous(range = c(3, 8), name = "Species Count") +
    coord_fixed(xlim = c(-123, -122), ylim = c(37.6, 37.9)) +
    labs(title = "Ecological Study Sites",
         subtitle = "Note: Interactive version available in HTML output",
         x = "Longitude", y = "Latitude", color = "Habitat Type") +
    theme_minimal() +
    theme(legend.position = "right")
}
```

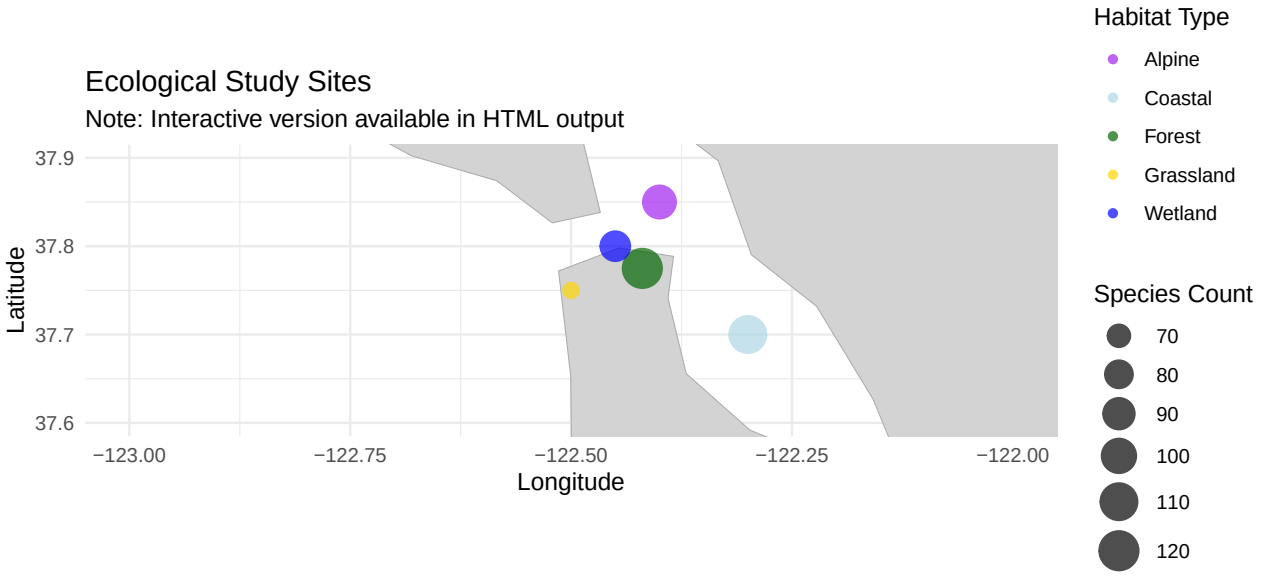


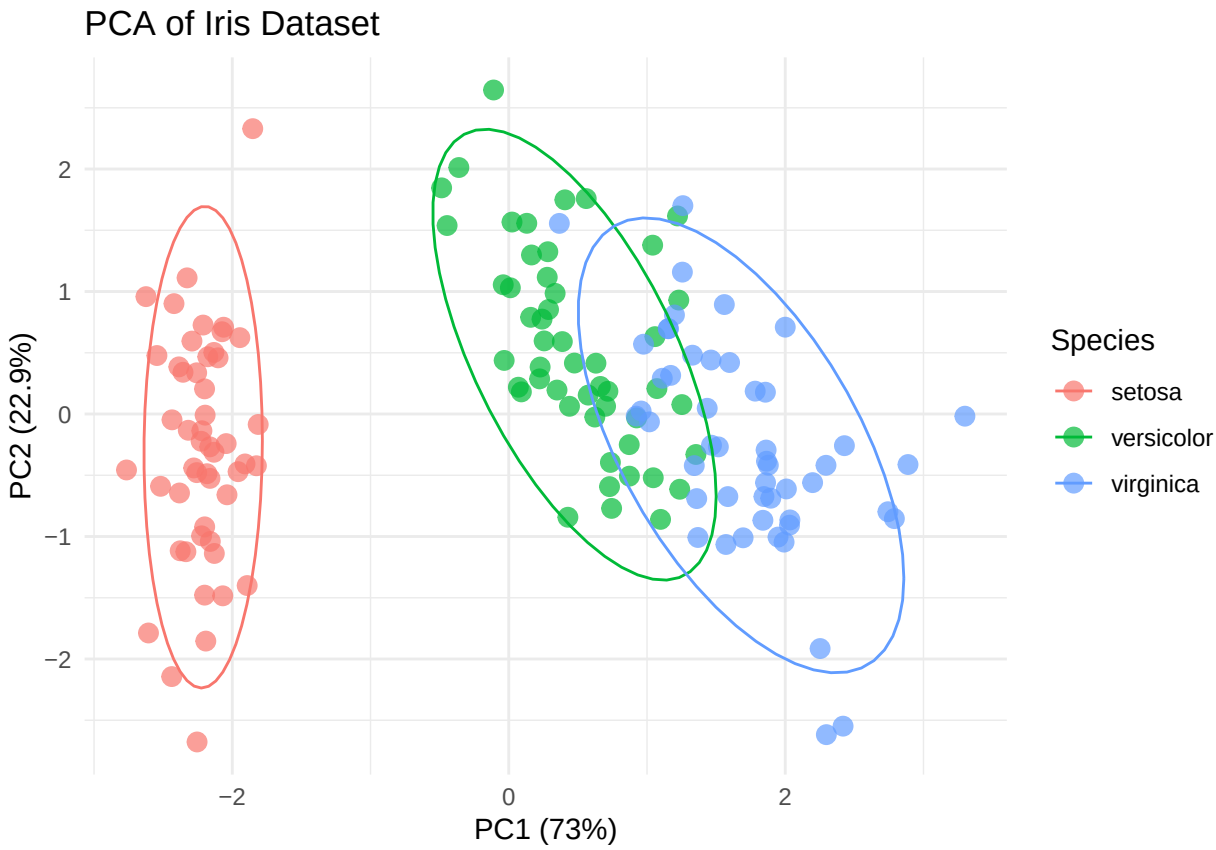
Figure 10.2: Ecological study sites across different habitat types. In the HTML version, this map is interactive and allows zooming, panning, and clicking on markers for details.

10.4 Specialized Ecological Visualizations

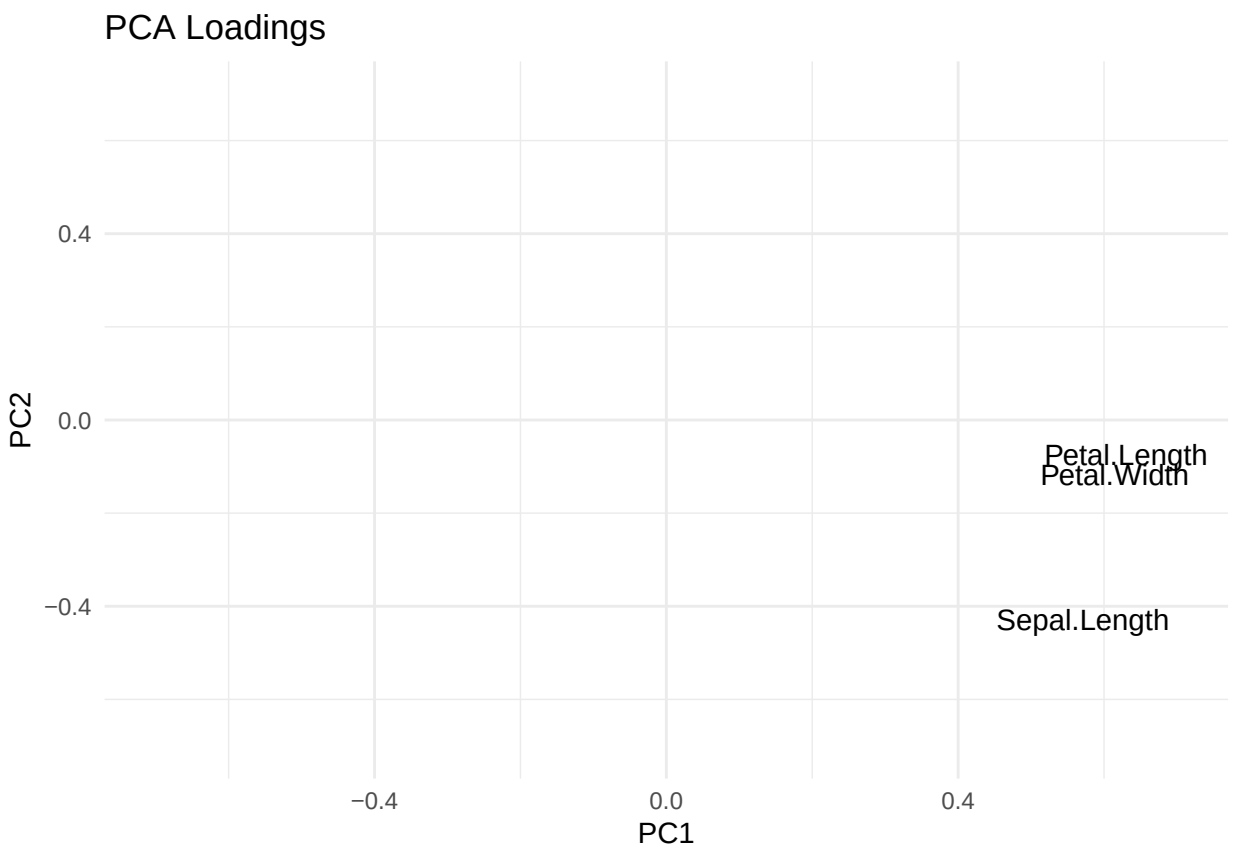
10.4.1 Ordination Plots

Ordination techniques like PCA and NMDS are common in ecological studies:

```
# Perform PCA on iris dataset
pca_result <- prcomp(iris[, 1:4], scale. = TRUE)
pca_data <- as.data.frame(pca_result$x)
pca_data$Species <- iris$Species

# Create a PCA biplot
ggplot(pca_data, aes(x = PC1, y = PC2, color = Species)) +
  geom_point(size = 3, alpha = 0.7) +
  stat_ellipse(level = 0.95) +
  labs(title = "PCA of Iris Dataset",
       x = paste0("PC1 (", round(summary(pca_result)$importance[2, 1] * 100, 1), "%)"),
       y = paste0("PC2 (", round(summary(pca_result)$importance[2, 2] * 100, 1), "%)")) +
  theme_minimal()
```



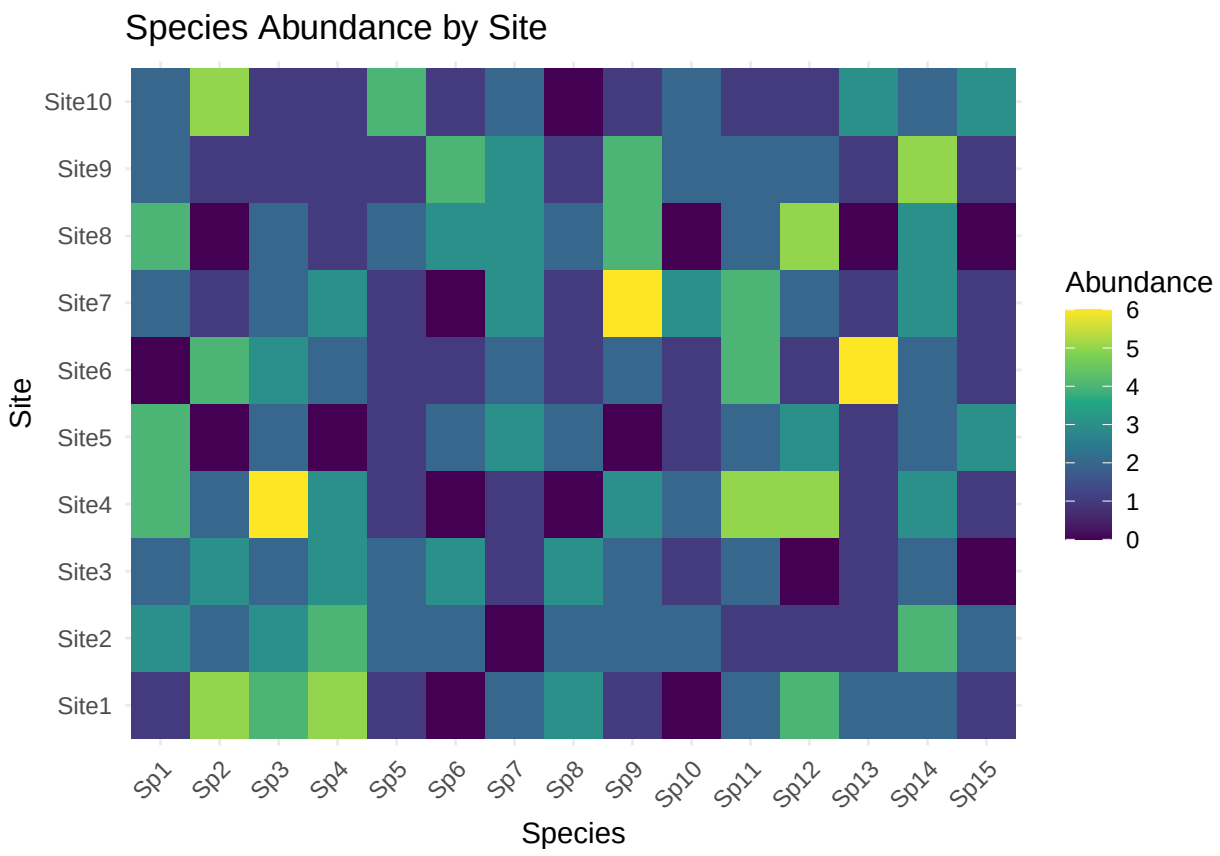

```r
# Create a loadings plot
loadings <- as.data.frame(pca_result$rotation)
loadings$variable <- rownames(loadings)

ggplot(loadings, aes(x = PC1, y = PC2)) +
  geom_segment(aes(x = 0, y = 0, xend = PC1 * 5, yend = PC2 * 5),
               arrow = arrow(length = unit(0.2, "cm")), color = "gray50") +
  geom_text(aes(label = variable), nudge_x = sign(loadings$PC1) * 0.05,
            nudge_y = sign(loadings$PC2) * 0.05) +
  labs(title = "PCA Loadings",
       x = "PC1",
       y = "PC2") +
  theme_minimal() +
  xlim(-0.7, 0.7) +
  ylim(-0.7, 0.7)
```



10.4.2 Heatmaps for Community Data

Heatmaps are useful for visualizing species-by-site matrices:

```
# Create a simulated species-by-site matrix
set.seed(123)
n_sites <- 10
n_species <- 15
community_matrix <- matrix(rpois(n_sites * n_species, lambda = 2),
                           nrow = n_sites, ncol = n_species)
rownames(community_matrix) <- paste0("Site", 1:n_sites)
colnames(community_matrix) <- paste0("Sp", 1:n_species)

# Convert to long format for ggplot
community_data <- as.data.frame(as.table(community_matrix))
names(community_data) <- c("Site", "Species", "Abundance")

# Create a heatmap
ggplot(community_data, aes(x = Species, y = Site, fill = Abundance)) +
  geom_tile() +
  scale_fill_viridis_c() +
  labs(title = "Species Abundance by Site",
       x = "Species",
       y = "Site",
       fill = "Abundance") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

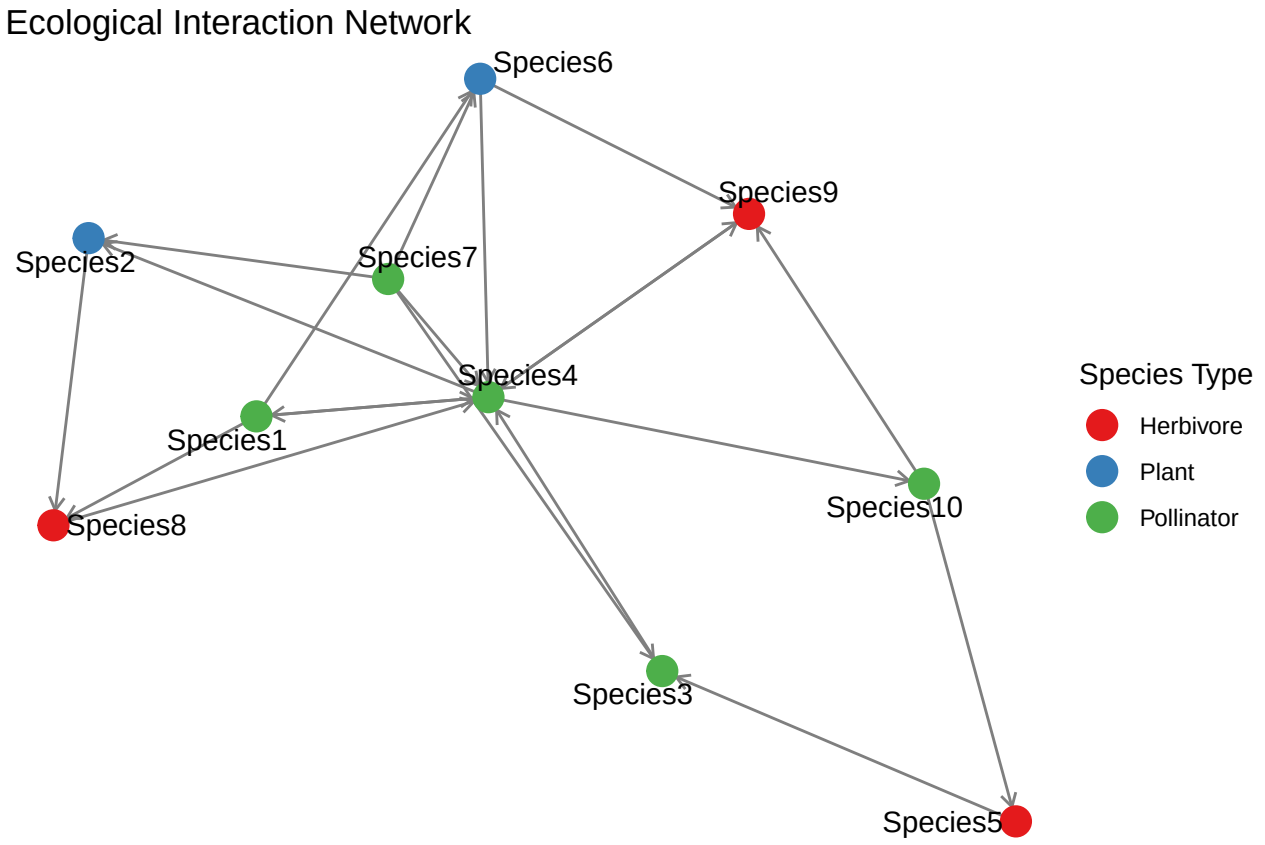


10.4.3 Network Diagrams for Ecological Interactions

Network diagrams can visualize species interactions:

```
library(igraph)
library(ggraph)

# Create a simulated interaction network
set.seed(456)
n_species <- 10
interaction_matrix <- matrix(rbinom(n_species^2, 1, 0.2),
                             nrow = n_species, ncol = n_species)
diag(interaction_matrix) <- 0  # No self-interactions
species_names <- paste0("Species", 1:n_species)
rownames(interaction_matrix) <- species_names
colnames(interaction_matrix) <- species_names

# Convert to igraph object
g <- graph_from_adjacency_matrix(interaction_matrix, mode = "directed")
V(g)$type <- sample(c("Plant", "Pollinator", "Herbivore"), n_species, replace = TRUE)

# Create a network diagram
ggraph(g, layout = "fr") +
  geom_edge_link(arrow = arrow(length = unit(2, "mm")),
                 end_cap = circle(2, "mm"),
                 color = "gray50") +
  geom_node_point(aes(color = type), size = 5) +
```

```
  geom_node_text(aes(label = name), repel = TRUE) +
  scale_color_brewer(palette = "Set1") +
  labs(title = "Ecological Interaction Network",
       color = "Species Type") +
  theme_void()
```



10.5 Visualizing Spatial Data

10.5.1 Creating Maps with ggplot2

Spatial visualization is crucial for ecological data:

```
library(ggplot2)
library(maps)
library(knitr)

# Get world map data
world <- map_data("world")

# Create sample species occurrence data
set.seed(789)
n_points <- 100
occurrences <- data.frame(
  species = sample(c("Species A", "Species B", "Species C"), n_points, replace = TRUE),
  longitude = runif(n_points, -10, 40),
  latitude = runif(n_points, 35, 60)
)
```

```
# Create a map
ggplot() +
  geom_polygon(data = world, aes(x = long, y = lat, group = group),
               fill = "white", color = "gray70", size = 0.2) +
  geom_point(data = occurrences,
             aes(x = longitude, y = latitude, color = species),
             alpha = 0.7, size = 2) +
  coord_fixed(xlim = c(-10, 40), ylim = c(35, 60)) +
  scale_color_viridis_d(option = "plasma", end = 0.8) +
  labs(title = "Species Distribution Map",
       subtitle = "Sample occurrences across Europe",
       x = "Longitude", y = "Latitude", color = "Species") +
  theme_minimal() +
  theme(panel.grid.major = element_line(color = "gray90", size = 0.2))
```

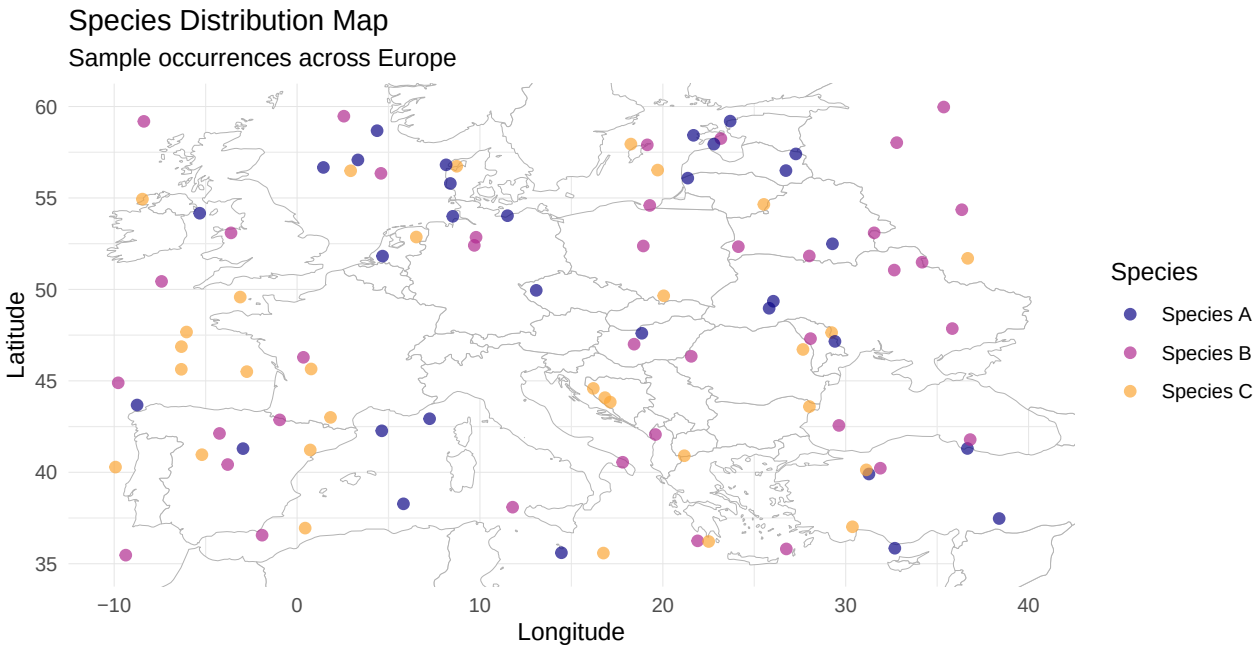


Figure 10.3: Distribution of sample species occurrences across Europe. The map shows the spatial patterns of three different species.

10.5.2 Visualizing Raster Data

Environmental raster data is common in ecological studies:

```
library(raster)
library(ggplot2)
library(viridis)
library(maps)

# Create a sample raster
r <- raster(ncol = 100, nrow = 100)
extent(r) <- c(-10, 40, 35, 60)  # Same extent as our map
values(r) <- runif(ncell(r)) * 10  # Random values
```

```
# Convert to data frame for ggplot
r_df <- as.data.frame(r, xy = TRUE)
colnames(r_df) <- c("longitude", "latitude", "value")

# Get world map data
world <- map_data("world")

# Create a raster map
ggplot() +
  geom_raster(data = r_df, aes(x = longitude, y = latitude, fill = value)) +
  geom_polygon(data = world, aes(x = long, y = lat, group = group),
               fill = NA, color = "gray30", size = 0.2) +
  scale_fill_viridis_c(option = "plasma", name = "Value") +
  coord_fixed(xlim = c(-10, 40), ylim = c(35, 60)) +
  labs(title = "Environmental Variable Distribution",
       subtitle = "Simulated environmental gradient across Europe",
       x = "Longitude", y = "Latitude") +
  theme_minimal() +
  theme(panel.grid = element_blank())
```

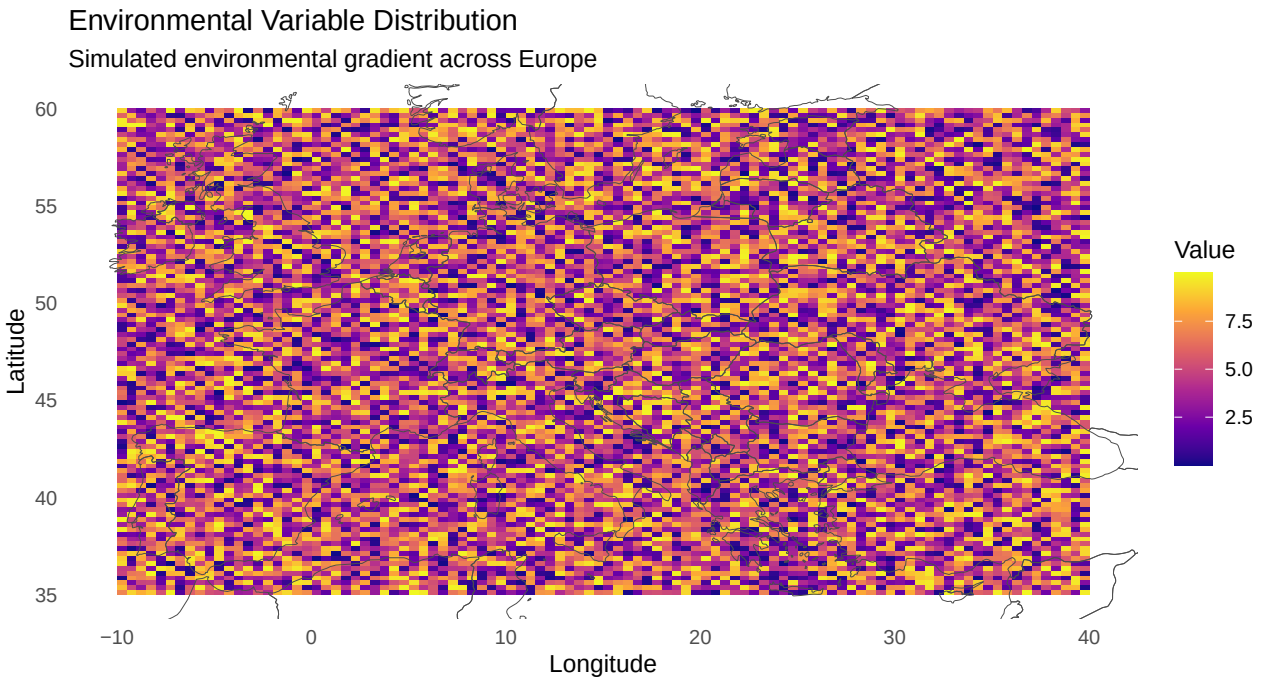


Figure 10.4: Environmental variable visualization across Europe. The raster data shows a simulated environmental gradient overlaid with country boundaries.

10.6 Advanced Visualization in Jamovi

While Jamovi has more limited visualization capabilities compared to R, it offers:

1. **Customization Options**: Adjust colors, labels, and themes
2. **Export Features**: Save high-resolution images for publications
3. **Interactive Elements**: Zoom and pan functionality in some plots

10.7 Summary

In this chapter, we've explored advanced data visualization techniques for ecological research:

- Creating publication-quality graphics with customized themes and color palettes
- Arranging multiple plots for comprehensive data presentation
- Building interactive visualizations for data exploration
- Developing specialized plots for ecological data (ordination, heatmaps, networks)
- Visualizing spatial data with maps

These advanced visualization techniques allow you to communicate complex ecological patterns and relationships more effectively, enhancing both data exploration and the presentation of research findings.

10.8 Exercises

1. Create a publication-quality figure with multiple panels using the patchwork package.
2. Design a custom theme for ggplot2 that matches the style guidelines of a scientific journal.
3. Convert a static ggplot2 visualization to an interactive plotly plot.
4. Create a heatmap to visualize a species-by-site abundance matrix.
5. Visualize a PCA or NMDS ordination of an ecological dataset.
6. Create a map showing the distribution of species or environmental variables.

Part IV

Advanced Topics

Chapter 11

Regression Analysis

11.1 Introduction

Regression analysis is a powerful statistical technique used to model the relationship between a dependent variable and one or more independent variables. In natural sciences research, regression models help us understand how environmental factors influence biological processes, predict future conditions, and test hypotheses about causal relationships.

11.2 Simple Linear Regression

Simple linear regression models the relationship between a dependent variable and a single independent variable.

11.2.1 The Linear Model

The simple linear regression model is represented by the equation:

$$Y = \beta_0 + \beta_1 X + \varepsilon$$

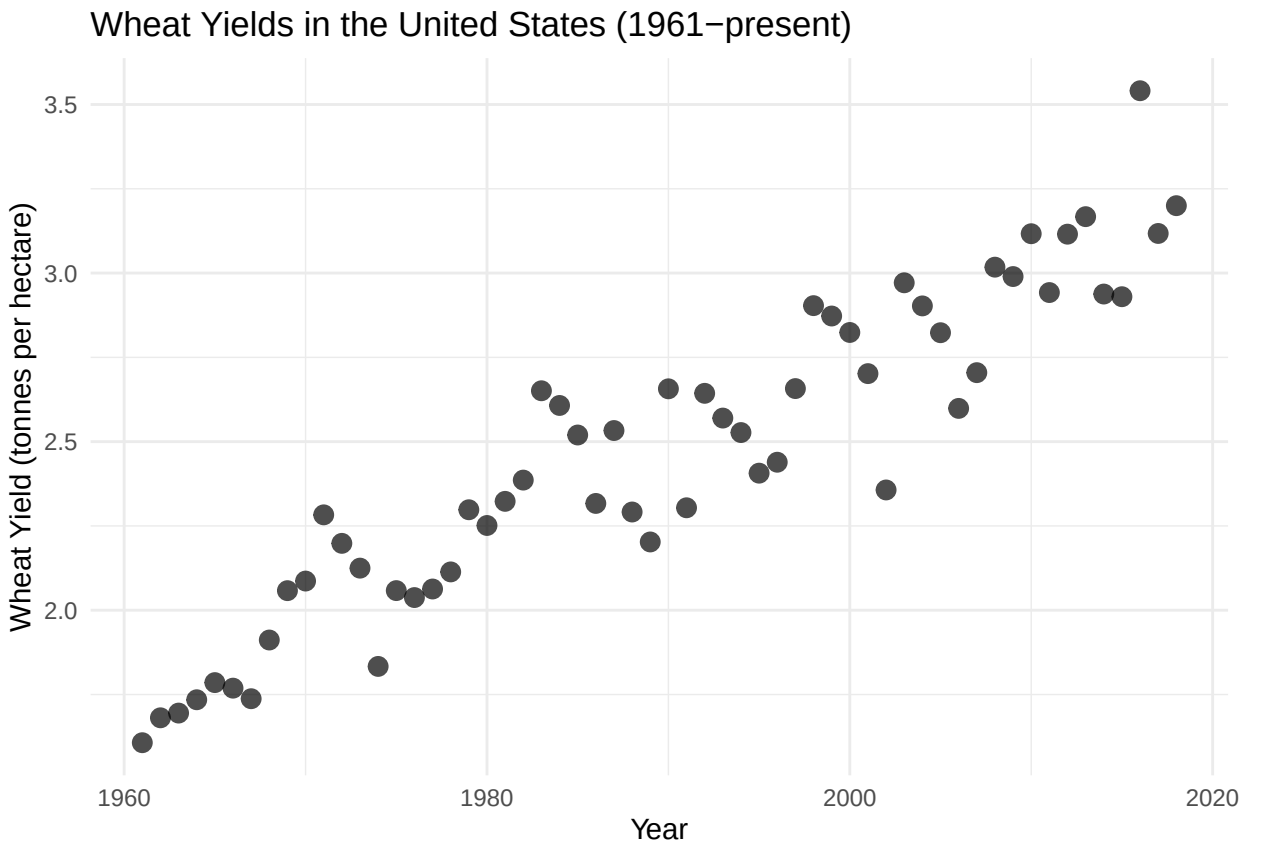
Where: - Y is the dependent variable - X is the independent variable - β_0 is the intercept - β_1 is the slope - ε is the error term

11.2.2 Example: Crop Yield Trends Over Time

Let's explore the relationship between time (years) and wheat yields using our agricultural dataset:

```
# Load necessary libraries
library(tidyverse)

# Load the crop yield dataset
crop_yields <- read_csv("../data/agriculture/crop_yields.csv")

# Filter data for a specific country (United States) and select relevant columns
us_wheat <- crop_yields %>%
  filter(Entity == "United States", !is.na(`Wheat (tonnes per hectare)`)) %>%
  select(Year, `Wheat (tonnes per hectare)`)

# Visualize the relationship
```

```
ggplot(us_wheat, aes(x = Year, y = `Wheat (tonnes per hectare)`)) +
  geom_point(size = 3, alpha = 0.7) +
  labs(title = "Wheat Yields in the United States (1961-present)",
       x = "Year",
       y = "Wheat Yield (tonnes per hectare)") +
  theme_minimal()
```

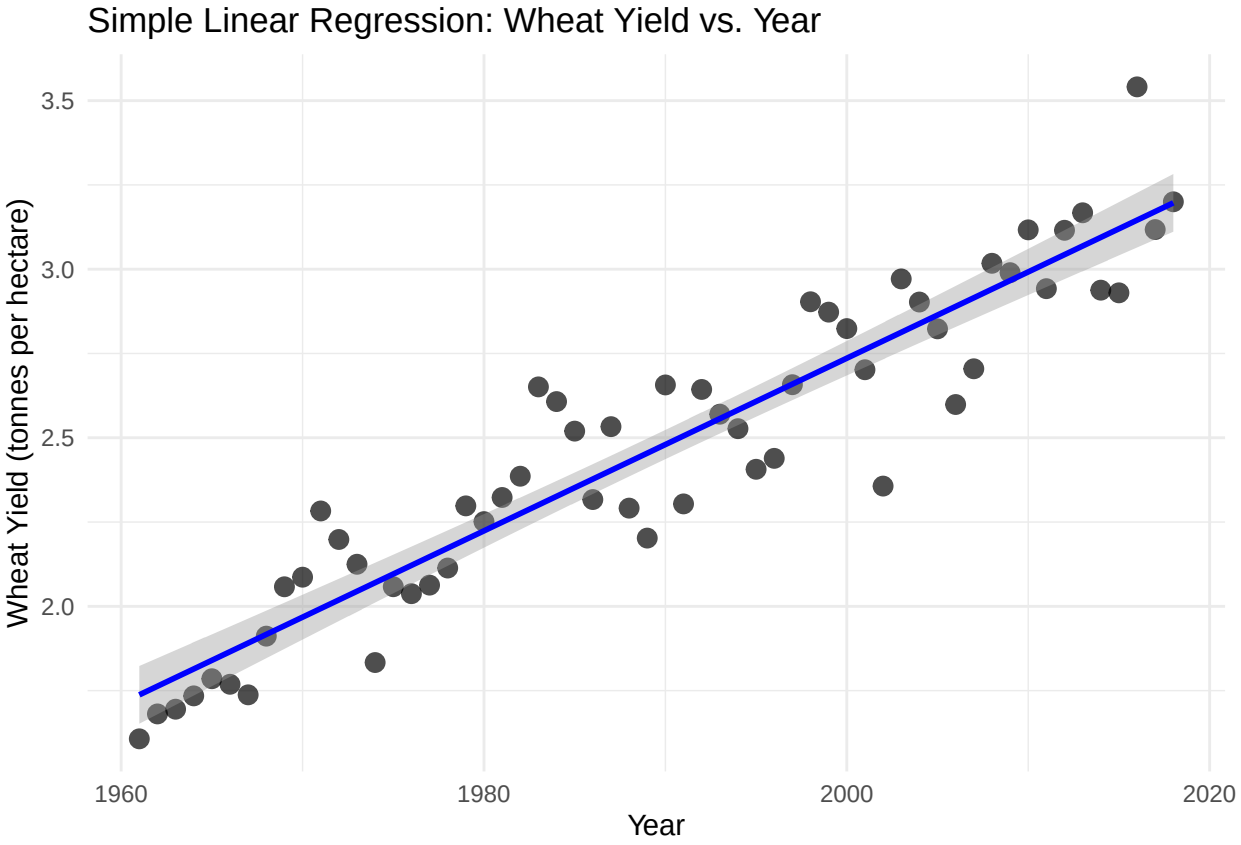



```
# Fit a simple linear regression model
model <- lm(`Wheat (tonnes per hectare)` ~ Year, data = us_wheat)

# Display model summary
summary(model)
```

```
Call:
lm(formula = `Wheat (tonnes per hectare)` ~ Year, data = us_wheat)

Residuals:
     Min       1Q   Median       3Q      Max
-0.43042 -0.09139 -0.00340  0.11184  0.39526

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -48.465987   2.571017  -18.85   <2e-16 ***
Year          0.025601   0.001292   19.81   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.1648 on 56 degrees of freedom
Multiple R-squared:  0.8751,    Adjusted R-squared:  0.8729
F-statistic: 392.5 on 1 and 56 DF,  p-value: < 2.2e-16
```

```r
# Add the regression line to the plot
ggplot(us_wheat, aes(x = Year, y = `Wheat (tonnes per hectare)`)) +
  geom_point(size = 3, alpha = 0.7) +
  geom_smooth(method = "lm", color = "blue") +
  labs(title = "Simple Linear Regression: Wheat Yield vs. Year",
       x = "Year",
       y = "Wheat Yield (tonnes per hectare)") +
  theme_minimal()
```



11.2.3 Interpreting the Model

The key components to interpret in a simple linear regression model are:

1. **Intercept** (β_0): The expected value of Y when X = 0
2. **Slope** (β_1): The expected change in Y for a one-unit increase in X
3. **R-squared**: The proportion of variance in Y explained by X
4. **p-value**: The statistical significance of the relationship

```r
# Extract key model parameters
intercept <- coef(model)[1]
slope <- coef(model)[2]
r_squared <- summary(model)$r.squared
p_value <- summary(model)$coefficients[2, 4]
```

```
# Create a table of results
results <- data.frame(
  Parameter = c("Intercept", "Slope", "R-squared", "p-value"),
  Value = c(intercept, slope, r_squared, p_value)
)

# Display the results
knitr::kable(results, digits = 4)
```

Parameter	Value
Intercept	-48.4660
Slope	0.0256
R-squared	0.8751
p-value	0.0000

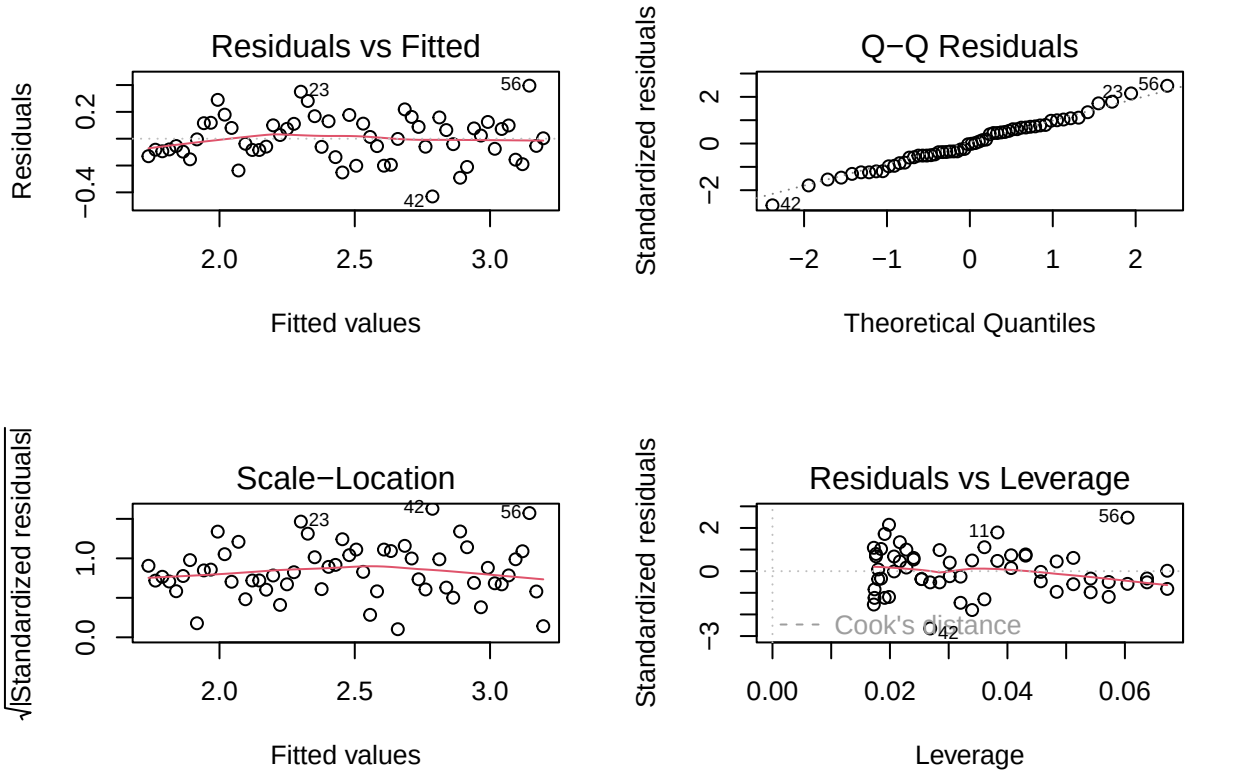
In this example, the slope represents the average annual increase in wheat yield (tonnes/hectare) in the United States. The R-squared value indicates what percentage of the variation in wheat yields can be explained by the year. The p-value tells us whether this relationship is statistically significant.

11.2.4 Checking Model Assumptions

Linear regression relies on several key assumptions:

1. **Linearity**: The relationship between X and Y is linear
2. **Independence**: Observations are independent of each other
3. **Homoscedasticity**: Constant variance of residuals
4. **Normality**: Residuals are normally distributed

```
# Diagnostic plots
par(mfrow = c(2, 2))
plot(model)
```




```
# Check normality of residuals with a formal test
shapiro.test(residuals(model))
```

```
    Shapiro-Wilk normality test

data:  residuals(model)
W = 0.99033, p-value = 0.9252
```

```
# Check homoscedasticity with a formal test
if(requireNamespace("car", quietly = TRUE)) {
  library(car)
  ncvTest(model)
} else {
  message("The 'car' package is not installed. Install it with install.packages('car') to run the non-c
}
```

```
Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare = 1.323418, Df = 1, p = 0.24998
```

11.3 Multiple Linear Regression

Multiple linear regression extends the simple linear model to include multiple independent variables.

11.3.1 The Multiple Regression Model

The multiple regression model is represented by the equation:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p + \varepsilon$$

Where: - Y is the dependent variable - $X_1, X_2, ..., X_p$ are the independent variables - $\beta_0, \beta_1, \beta_2, ..., \beta_p$ are the coefficients - ε is the error term

11.3.2 Example: Factors Affecting Crop Yields

Let's model wheat yield as a function of multiple crop yields, which might indicate similar agricultural conditions:

```
# Prepare data for multiple regression
multi_crop_data <- crop_yields %>%
  filter(!is.na(`Wheat (tonnes per hectare)`) & !is.na(`Rice (tonnes per hectare)`) & !is.na(`Maize (to
  select(Entity, Year, `Wheat (tonnes per hectare)`, `Rice (tonnes per hectare)`, `Maize (tonnes per he

# View the first few rows
head(multi_crop_data)
```

```
# A tibble: 6 x 5
  Entity       Year `Wheat (tonnes per hectare)` `Rice (tonnes per hectare)`
  <chr>       <dbl>                        <dbl>                       <dbl>
1 Afghanistan  1961                         1.02                        1.52
2 Afghanistan  1962                         0.974                       1.52
3 Afghanistan  1963                         0.832                       1.52
4 Afghanistan  1964                         0.951                       1.73
5 Afghanistan  1965                         0.972                       1.73
6 Afghanistan  1966                         0.867                       1.52
# i 1 more variable: `Maize (tonnes per hectare)` <dbl>
```

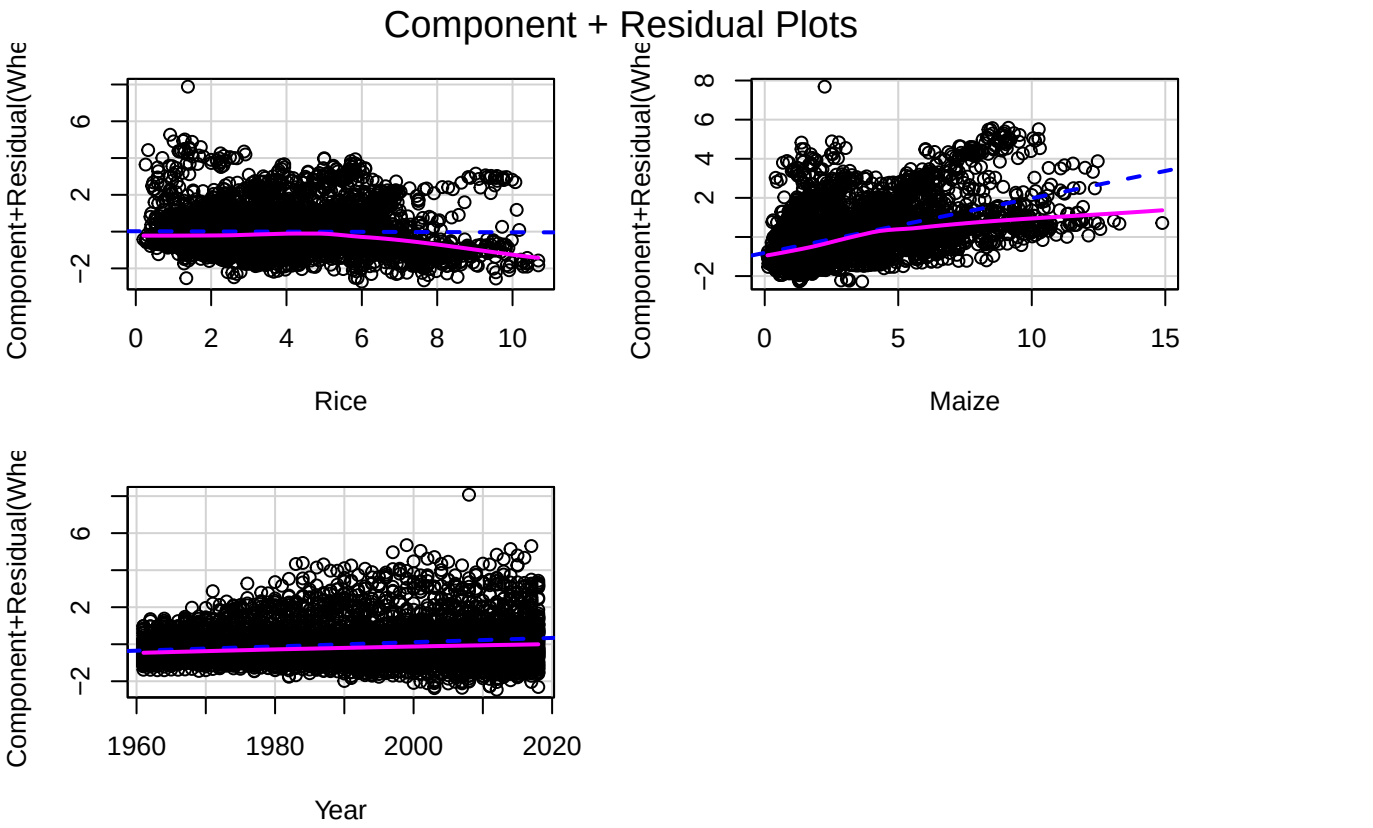
```
# Fit a multiple regression model
multi_model <- lm(`Wheat (tonnes per hectare)` ~ `Rice (tonnes per hectare)` + `Maize (tonnes per hecta

# Display model summary
summary(multi_model)
```

```
Call:
lm(formula = `Wheat (tonnes per hectare)` ~ `Rice (tonnes per hectare)` +
    `Maize (tonnes per hectare)` + Year, data = multi_crop_data)

Residuals:
    Min      1Q  Median      3Q     Max
-2.7029 -0.6135 -0.2079  0.3877  7.8709

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                 -2.161e+01  1.776e+00 -12.171   <2e-16 ***
`Rice (tonnes per hectare)`  -5.426e-03  9.720e-03  -0.558    0.577
`Maize (tonnes per hectare)`  2.790e-01  8.512e-03  32.776   <2e-16 ***
Year                          1.148e-02  8.965e-04  12.810   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.025 on 5722 degrees of freedom
Multiple R-squared:  0.3411,    Adjusted R-squared:  0.3407
F-statistic: 987.2 on 3 and 5722 DF,  p-value: < 2.2e-16
```

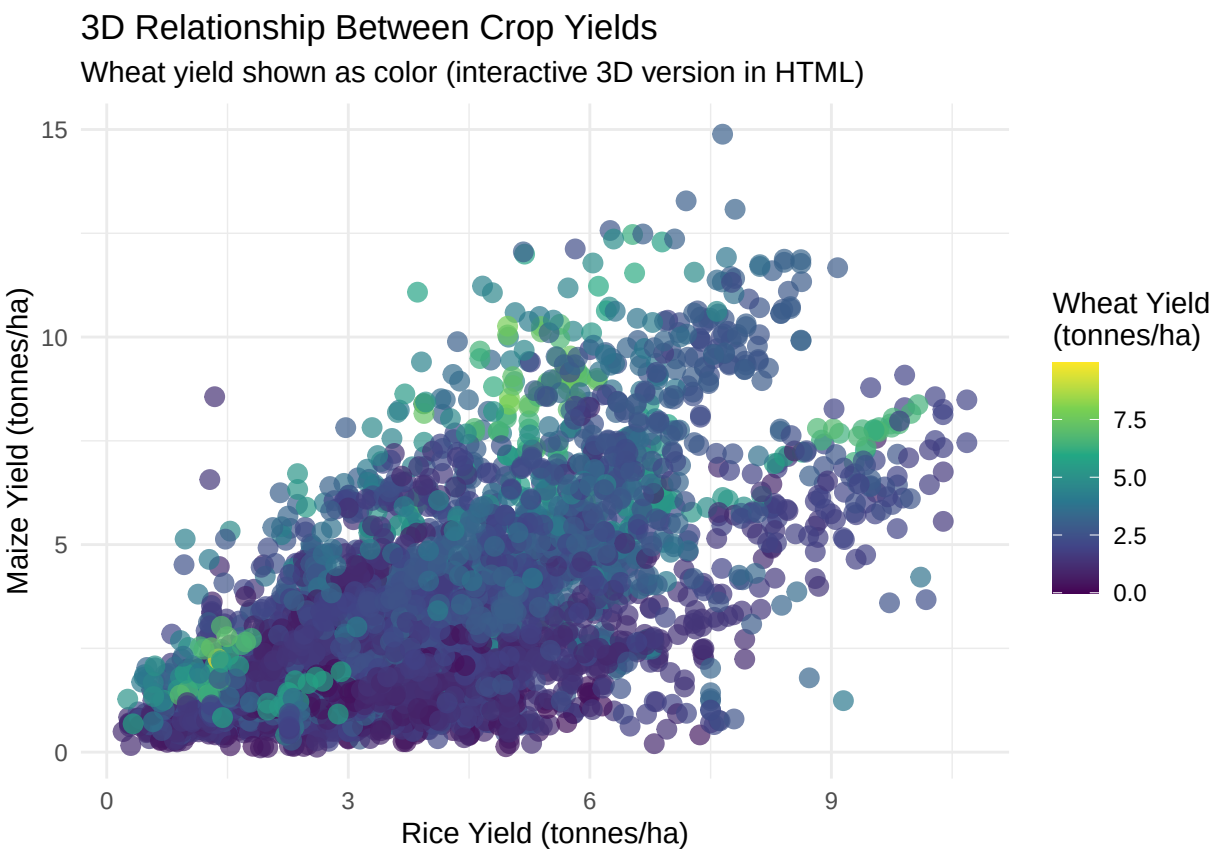
11.3.3 Visualizing Multiple Regression

Visualizing multiple regression models is challenging due to the multidimensional nature of the data. Here are some approaches:

```
# Partial residual plots
if(requireNamespace("car", quietly = TRUE)) {
  library(car)
  # Create a model with simpler variable names to avoid issues with crPlots
  renamed_data <- multi_crop_data %>%
    rename(Wheat = `Wheat (tonnes per hectare)`,
           Rice = `Rice (tonnes per hectare)`,
           Maize = `Maize (tonnes per hectare)`)

  simple_model <- lm(Wheat ~ Rice + Maize + Year, data = renamed_data)
  crPlots(simple_model)
} else {
  # Alternative: create individual scatter plots
  par(mfrow = c(1, 3))
  plot(multi_crop_data$`Rice (tonnes per hectare)`, multi_crop_data$`Wheat (tonnes per hectare)`,
       xlab = "Rice Yield", ylab = "Wheat Yield",
       main = "Wheat vs. Rice")
  plot(multi_crop_data$`Maize (tonnes per hectare)`, multi_crop_data$`Wheat (tonnes per hectare)`,
       xlab = "Maize Yield", ylab = "Wheat Yield",
       main = "Wheat vs. Maize")
  plot(multi_crop_data$Year, multi_crop_data$`Wheat (tonnes per hectare)`,
       xlab = "Year", ylab = "Wheat Yield",
       main = "Wheat vs. Year")
}
```



```
# 3D visualization (for a subset of variables)
library(knitr)
if(knitr::is_html_output() && requireNamespace("plotly", quietly = TRUE)) {
  # For HTML output, use the interactive plotly version
  library(plotly)
  plot_ly(multi_crop_data,
          x = ~`Rice (tonnes per hectare)`,
          y = ~`Maize (tonnes per hectare)`,
          z = ~`Wheat (tonnes per hectare)`,
          type = "scatter3d", mode = "markers",
          marker = list(size = 5, color = ~`Wheat (tonnes per hectare)`,
                        colorscale = "Viridis")) %>%
    layout(title = "3D Relationship Between Crop Yields",
           scene = list(
             xaxis = list(title = "Rice Yield (tonnes/ha)"),
             yaxis = list(title = "Maize Yield (tonnes/ha)"),
             zaxis = list(title = "Wheat Yield (tonnes/ha)")))
} else {
  # For PDF output, use a static 3D scatter plot with ggplot2
  library(ggplot2)

  # Create a 2D plot with color as the third dimension
  ggplot(multi_crop_data,
         aes(x = `Rice (tonnes per hectare)`,
             y = `Maize (tonnes per hectare)`,
             color = `Wheat (tonnes per hectare)`)) +
    geom_point(size = 3, alpha = 0.7) +
    scale_color_viridis_c(option = "viridis", name = "Wheat Yield\n(tonnes/ha)") +
    labs(title = "3D Relationship Between Crop Yields",
         subtitle = "Wheat yield shown as color (interactive 3D version in HTML)",
         x = "Rice Yield (tonnes/ha)",
         y = "Maize Yield (tonnes/ha)") +
    theme_minimal() +
    theme(legend.position = "right")
}
```



11.3.4 Variable Selection

When working with multiple predictors, it's essential to select the most relevant variables:

```
# Stepwise variable selection
if(requireNamespace("MASS", quietly = TRUE)) {
  library(MASS)
  step_model <- stepAIC(multi_model, direction = "both")
  summary(step_model)
} else {
  message("The 'MASS' package is not installed. Install it with install.packages('MASS') to perform step
}
```

```
Start:  AIC=292.2
`Wheat (tonnes per hectare)` ~ `Rice (tonnes per hectare)` +
    `Maize (tonnes per hectare)` + Year

                               Df Sum of Sq    RSS     AIC
- `Rice (tonnes per hectare)`   1      0.33 6017.7  290.51
<none>                                     6017.4  292.20
- Year                          1    172.58 6189.9  452.11
- `Maize (tonnes per hectare)`  1   1129.71 7147.1 1275.38

Step:  AIC=290.51
`Wheat (tonnes per hectare)` ~ `Maize (tonnes per hectare)` +
    Year

                               Df Sum of Sq    RSS     AIC
```

```
<none>                                     6017.7  290.51
+ `Rice (tonnes per hectare)`   1      0.33 6017.4  292.20
- Year                          1    172.33 6190.0  450.18
- `Maize (tonnes per hectare)`  1   1930.18 7947.9 1881.48


Call:
lm(formula = `Wheat (tonnes per hectare)` ~ `Maize (tonnes per hectare)` +
    Year, data = multi_crop_data)

Residuals:
    Min      1Q  Median      3Q     Max
-2.6986 -0.6123 -0.2065  0.3857  7.8812

Coefficients:
                             Estimate Std. Error t value Pr(>|t|)
(Intercept)                -2.156e+01  1.773e+00  -12.16   <2e-16 ***
`Maize (tonnes per hectare)`  2.759e-01  6.439e-03   42.84   <2e-16 ***
Year                        1.145e-02  8.946e-04   12.80   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.025 on 5723 degrees of freedom
Multiple R-squared:  0.341, Adjusted R-squared:  0.3408
F-statistic:  1481 on 2 and 5723 DF,  p-value: < 2.2e-16
```

```r
# Variance Inflation Factor (VIF) to check for multicollinearity
if(requireNamespace("car", quietly = TRUE)) {
  # Use the renamed data and simple model from earlier
  vif(simple_model)
} else {
  message("The 'car' package is not installed. Install it with install.packages('car') to calculate VIF

  # Alternative: correlation matrix
  cor_matrix <- cor(multi_crop_data[, c("Rice (tonnes per hectare)", "Maize (tonnes per hectare)", "Yea
  print("Correlation matrix of predictors:")
  print(cor_matrix)
}
```

```
    Rice    Maize     Year
1.963261 2.108941 1.212106
```

11.4 Polynomial Regression

Polynomial regression allows modeling of nonlinear relationships by including polynomial terms.

11.4.1 Example: Nonlinear Crop Yield Trends

Let's explore whether the relationship between time and wheat yields might be nonlinear:

```r
# Create a dataset for polynomial regression
us_wheat$Year_centered <- us_wheat$Year - mean(us_wheat$Year)  # Center year to reduce multicollinearity

# Fit polynomial models of different degrees
poly1 <- lm(`Wheat (tonnes per hectare)` ~ Year_centered, data = us_wheat)
```

```r
poly2 <- lm(`Wheat (tonnes per hectare)` ~ Year_centered + I(Year_centered^2), data = us_wheat)
poly3 <- lm(`Wheat (tonnes per hectare)` ~ Year_centered + I(Year_centered^2) + I(Year_centered^3), data

# Compare models
anova(poly1, poly2, poly3)
```

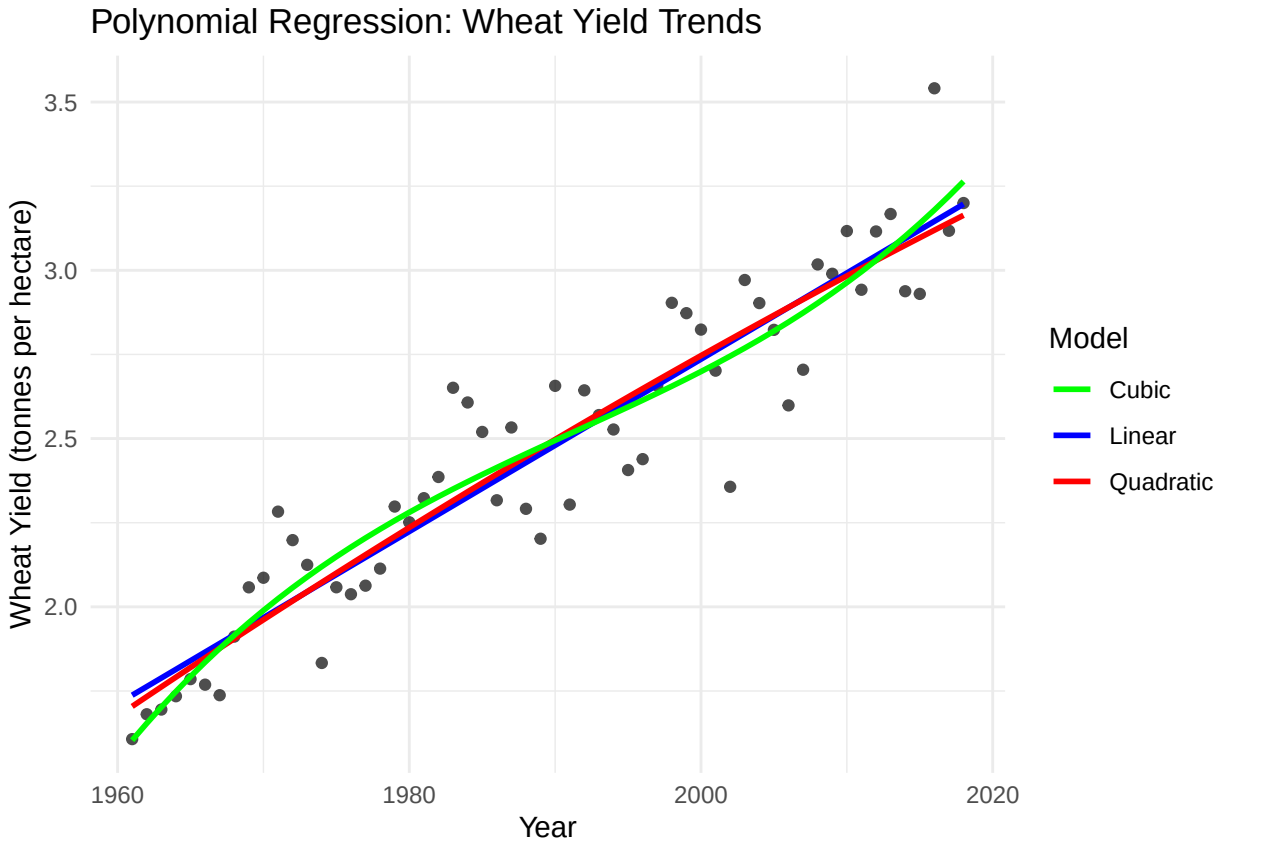
```
Analysis of Variance Table

Model 1: `Wheat (tonnes per hectare)` ~ Year_centered
Model 2: `Wheat (tonnes per hectare)` ~ Year_centered + I(Year_centered^2)
Model 3: `Wheat (tonnes per hectare)` ~ Year_centered + I(Year_centered^2) +
    I(Year_centered^3)
  Res.Df    RSS Df Sum of Sq      F  Pr(>F)
1     56 1.5200
2     55 1.5058  1  0.014257 0.5489 0.46198
3     54 1.4027  1  0.103114 3.9697 0.05139 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Summary of the best model (based on the ANOVA result)
best_poly_model <- poly2  # Change this based on the ANOVA results
summary(best_poly_model)
```

```

Call:
lm(formula = `Wheat (tonnes per hectare)` ~ Year_centered + I(Year_centered^2),
    data = us_wheat)

Residuals:
     Min       1Q   Median       3Q      Max
-0.43818 -0.08751 -0.00966  0.10875  0.42167

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         2.485e+00  3.260e-02  76.222   <2e-16 ***
Year_centered       2.560e-02  1.298e-03  19.726   <2e-16 ***
I(Year_centered^2) -6.258e-05  8.671e-05  -0.722    0.474
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1655 on 55 degrees of freedom
Multiple R-squared:  0.8763,    Adjusted R-squared:  0.8718
F-statistic: 194.8 on 2 and 55 DF,  p-value: < 2.2e-16
```

```r
# Visualize the polynomial fit
us_wheat$Year_orig <- us_wheat$Year  # Keep original year for plotting
us_wheat <- us_wheat %>%
  arrange(Year_orig)

# Generate predictions
us_wheat$pred_linear <- predict(poly1)
us_wheat$pred_quadratic <- predict(poly2)
us_wheat$pred_cubic <- predict(poly3)

# Plot the data with different model fits
```

```
ggplot(us_wheat, aes(x = Year_orig, y = `Wheat (tonnes per hectare)`)) +
  geom_point(alpha = 0.7) +
  geom_line(aes(y = pred_linear, color = "Linear"), size = 1) +
  geom_line(aes(y = pred_quadratic, color = "Quadratic"), size = 1) +
  geom_line(aes(y = pred_cubic, color = "Cubic"), size = 1) +
  scale_color_manual(values = c("Linear" = "blue", "Quadratic" = "red", "Cubic" = "green")) +
  labs(title = "Polynomial Regression: Wheat Yield Trends",
       x = "Year",
       y = "Wheat Yield (tonnes per hectare)",
       color = "Model") +
  theme_minimal()
```



11.5 Generalized Linear Models (GLMs)

Generalized linear models extend linear regression to handle response variables with non-normal distributions.

11.5.1 Logistic Regression

Logistic regression is used when the dependent variable is binary. Let's use our coffee economics dataset to predict coffee quality:

```
# Load the coffee economics dataset
coffee_data <- read_csv("../data/economics/economic.csv")

# View the structure of the dataset
str(coffee_data)
```

```
spc_tbl_ [1,339 x 43] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ total_cup_points     : num [1:1339] 90.6 89.9 89.8 89 88.8 ...
 $ species              : chr [1:1339] "Arabica" "Arabica" "Arabica" "Arabica" ...
 $ owner                : chr [1:1339] "metad plc" "metad plc" "grounds for health admin" "yidnekachew d
 $ country_of_origin    : chr [1:1339] "Ethiopia" "Ethiopia" "Guatemala" "Ethiopia" ...
 $ farm_name            : chr [1:1339] "metad plc" "metad plc" "san marcos barrancas \"san cristobal cue
 $ lot_number           : chr [1:1339] NA NA NA NA ...
 $ mill                 : chr [1:1339] "metad plc" "metad plc" NA "wolensu" ...
 $ ico_number           : chr [1:1339] "2014/2015" "2014/2015" NA NA ...
 $ company              : chr [1:1339] "metad agricultural developmet plc" "metad agricultural developme
 $ altitude             : chr [1:1339] "1950-2200" "1950-2200" "1600 - 1800 m" "1800-2200" ...
 $ region               : chr [1:1339] "guji-hambela" "guji-hambela" NA "oromia" ...
 $ producer             : chr [1:1339] "METAD PLC" "METAD PLC" NA "Yidnekachew Dabessa Coffee Plantation
 $ number_of_bags       : num [1:1339] 300 300 5 320 300 100 100 300 300 50 ...
 $ bag_weight           : chr [1:1339] "60 kg" "60 kg" "1" "60 kg" ...
 $ in_country_partner   : chr [1:1339] "METAD Agricultural Development plc" "METAD Agricultural Develop
 $ harvest_year         : chr [1:1339] "2014" "2014" NA "2014" ...
 $ grading_date         : chr [1:1339] "April 4th, 2015" "April 4th, 2015" "May 31st, 2010" "March 26th
 $ owner_1              : chr [1:1339] "metad plc" "metad plc" "Grounds for Health Admin" "Yidnekachew
 $ variety              : chr [1:1339] NA "Other" "Bourbon" NA ...
 $ processing_method    : chr [1:1339] "Washed / Wet" "Washed / Wet" NA "Natural / Dry" ...
 $ aroma                : num [1:1339] 8.67 8.75 8.42 8.17 8.25 8.58 8.42 8.25 8.67 8.08 ...
 $ flavor               : num [1:1339] 8.83 8.67 8.5 8.58 8.5 8.42 8.5 8.33 8.67 8.58 ...
 $ aftertaste           : num [1:1339] 8.67 8.5 8.42 8.42 8.25 8.42 8.33 8.5 8.58 8.5 ...
 $ acidity              : num [1:1339] 8.75 8.58 8.42 8.42 8.5 8.5 8.5 8.42 8.42 8.5 ...
 $ body                 : num [1:1339] 8.5 8.42 8.33 8.5 8.42 8.25 8.25 8.33 8.33 7.67 ...
 $ balance              : num [1:1339] 8.42 8.42 8.42 8.25 8.33 8.33 8.25 8.5 8.42 8.42 ...
 $ uniformity           : num [1:1339] 10 10 10 10 10 10 10 10 9.33 10 ...
 $ clean_cup            : num [1:1339] 10 10 10 10 10 10 10 10 10 10 ...
 $ sweetness            : num [1:1339] 10 10 10 10 10 10 10 9.33 9.33 10 ...
 $ cupper_points        : num [1:1339] 8.75 8.58 9.25 8.67 8.58 8.33 8.5 9 8.67 8.5 ...
 $ moisture             : num [1:1339] 0.12 0.12 0 0.11 0.12 0.11 0.11 0.03 0.03 0.1 ...
 $ category_one_defects : num [1:1339] 0 0 0 0 0 0 0 0 0 0 ...
 $ quakers              : num [1:1339] 0 0 0 0 0 0 0 0 0 0 ...
 $ color                : chr [1:1339] "Green" "Green" NA "Green" ...
 $ category_two_defects : num [1:1339] 0 1 0 2 2 1 0 0 0 4 ...
 $ expiration           : chr [1:1339] "April 3rd, 2016" "April 3rd, 2016" "May 31st, 2011" "March 25th
 $ certification_body   : chr [1:1339] "METAD Agricultural Development plc" "METAD Agricultural Develop
 $ certification_address: chr [1:1339] "309fcf77415a3661ae83e027f7e5f05dad786e44" "309fcf77415a3661ae83e
 $ certification_contact: chr [1:1339] "19fef5a731de2db57d16da10287413f5f99bc2dd" "19fef5a731de2db57d16e
 $ unit_of_measurement  : chr [1:1339] "m" "m" "m" "m" ...
 $ altitude_low_meters  : num [1:1339] 1950 1950 1600 1800 1950 ...
 $ altitude_high_meters : num [1:1339] 2200 2200 1800 2200 2200 NA NA 1700 1700 1850 ...
 $ altitude_mean_meters : num [1:1339] 2075 2075 1700 2000 2075 ...
 - attr(*, "spec")=
  .. cols(
  ..   total_cup_points = col_double(),
  ..   species = col_character(),
  ..   owner = col_character(),
  ..   country_of_origin = col_character(),
  ..   farm_name = col_character(),
  ..   lot_number = col_character(),
  ..   mill = col_character(),
  ..   ico_number = col_character(),
```

```
  ..   company = col_character(),
  ..   altitude = col_character(),
  ..   region = col_character(),
  ..   producer = col_character(),
  ..   number_of_bags = col_double(),
  ..   bag_weight = col_character(),
  ..   in_country_partner = col_character(),
  ..   harvest_year = col_character(),
  ..   grading_date = col_character(),
  ..   owner_1 = col_character(),
  ..   variety = col_character(),
  ..   processing_method = col_character(),
  ..   aroma = col_double(),
  ..   flavor = col_double(),
  ..   aftertaste = col_double(),
  ..   acidity = col_double(),
  ..   body = col_double(),
  ..   balance = col_double(),
  ..   uniformity = col_double(),
  ..   clean_cup = col_double(),
  ..   sweetness = col_double(),
  ..   cupper_points = col_double(),
  ..   moisture = col_double(),
  ..   category_one_defects = col_double(),
  ..   quakers = col_double(),
  ..   color = col_character(),
  ..   category_two_defects = col_double(),
  ..   expiration = col_character(),
  ..   certification_body = col_character(),
  ..   certification_address = col_character(),
  ..   certification_contact = col_character(),
  ..   unit_of_measurement = col_character(),
  ..   altitude_low_meters = col_double(),
  ..   altitude_high_meters = col_double(),
  ..   altitude_mean_meters = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

```
# Prepare data for logistic regression
# We'll create a binary variable for high-quality coffee
if(any(grepl("total_cup_points", names(coffee_data), ignore.case = TRUE))) {
  # Find the column name that matches "total_cup_points" (case insensitive)
  score_col <- names(coffee_data)[grep("total_cup_points", names(coffee_data), ignore.case = TRUE)][1]

  # Create a binary variable for high-quality coffee
  coffee_data$high_quality <- as.numeric(coffee_data[[score_col]] > median(coffee_data[[score_col]], na

  # Select predictors (this will depend on your actual dataset)
  # For this example, we'll use numeric columns as potential predictors
  numeric_cols <- sapply(coffee_data, is.numeric)
  predictor_cols <- names(coffee_data)[numeric_cols & names(coffee_data) != score_col &
                                       names(coffee_data) != "high_quality"]

  if(length(predictor_cols) >= 3) {
```

```
    # Create formula
    formula_str <- paste("high_quality ~", paste(predictor_cols[1:3], collapse = " + "))

    # Fit logistic regression model
    logit_model <- glm(as.formula(formula_str), family = binomial, data = coffee_data)

    # Display model summary
    summary(logit_model)

    # Calculate odds ratios
    odds_ratios <- exp(coef(logit_model))
    odds_ratio_df <- data.frame(
      Variable = names(odds_ratios),
      Odds_Ratio = odds_ratios
    )

    # Display odds ratios
    knitr::kable(odds_ratio_df, digits = 3)
  } else {
    message("Not enough numeric predictors available for logistic regression.")
  }
} else {
  message("Could not find a column for coffee quality scores. Using a simulated example instead.")

  # Create a simulated example
  set.seed(123)
  n <- 100
  altitude <- rnorm(n, 1500, 300)
  rainfall <- rnorm(n, 2000, 500)
  high_quality <- rbinom(n, 1, plogis(-10 + 0.005 * altitude + 0.0002 * rainfall))

  sim_coffee <- data.frame(altitude = altitude, rainfall = rainfall, high_quality = high_quality)

  # Fit logistic regression model
  logit_model <- glm(high_quality ~ altitude + rainfall, family = binomial, data = sim_coffee)

  # Display model summary
  summary(logit_model)

  # Calculate odds ratios
  odds_ratios <- exp(coef(logit_model))
  odds_ratio_df <- data.frame(
    Variable = names(odds_ratios),
    Odds_Ratio = odds_ratios
  )

  # Display odds ratios
  knitr::kable(odds_ratio_df, digits = 3)

  # Visualize the relationship
  ggplot(sim_coffee, aes(x = altitude, y = high_quality, color = factor(high_quality))) +
    geom_point(alpha = 0.7) +
    geom_smooth(method = "glm", method.args = list(family = "binomial"), se = TRUE, color = "black") +
```

```
    labs(title = "Logistic Regression: Coffee Quality vs. Altitude",
         x = "Altitude (m)",
         y = "Probability of High Quality",
         color = "High Quality") +
    theme_minimal()
}
```

	Variable	Odds_Ratio
(Intercept)	(Intercept)	0.000
number_of_bags	number_of_bags	1.001
aroma	aroma	73.213
flavor	flavor	6337.322

11.5.2 Poisson Regression

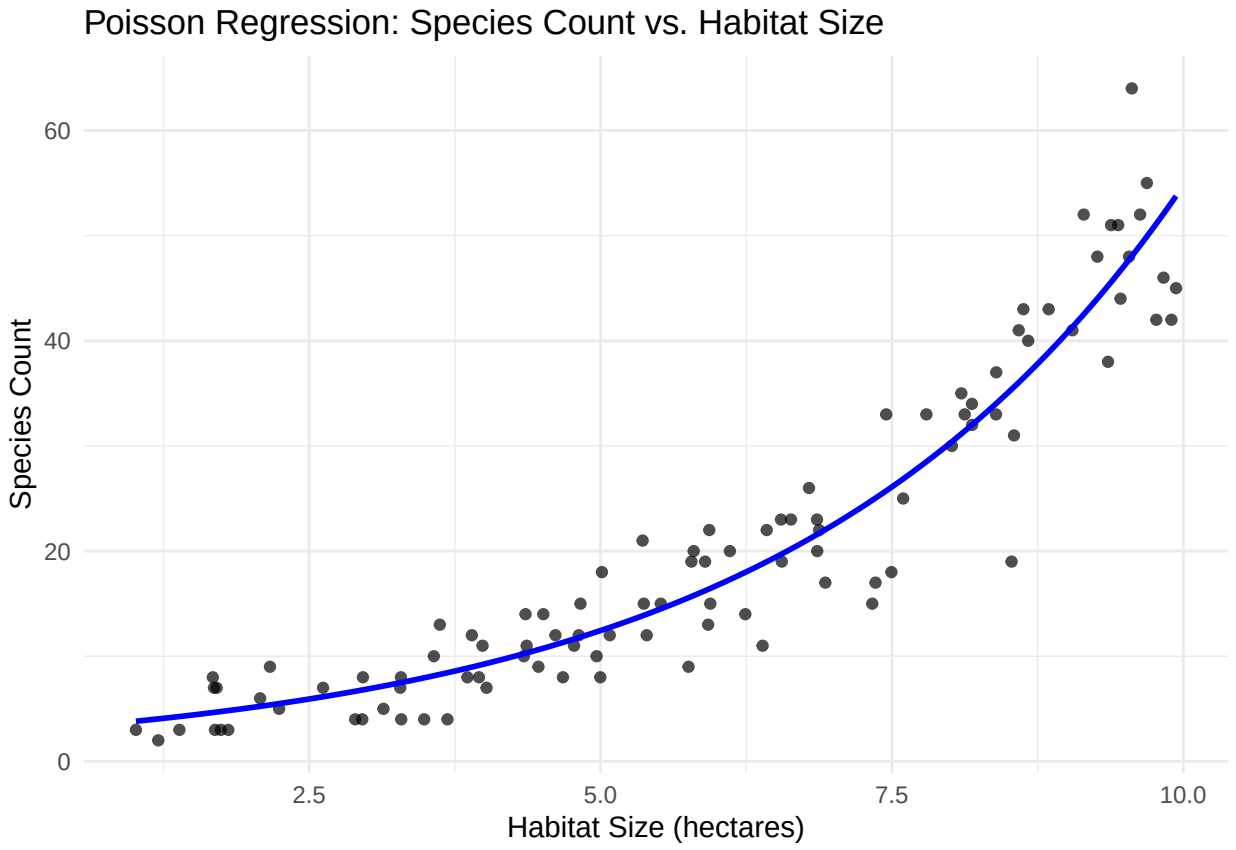
Poisson regression is used for count data. Let's use it to model species counts from our biodiversity dataset:

```
# Using a simulated example for Poisson regression.
# This avoids issues with column names and data types
message("Using a simulated example for Poisson regression.")

# Create a simulated example for species counts
set.seed(456)
n <- 100
habitat_size <- runif(n, 1, 10)  # Habitat size in hectares
species_count <- rpois(n, lambda = exp(1 + 0.3 * habitat_size))

sim_biodiversity <- data.frame(habitat_size = habitat_size, species_count = species_count)

# Fit Poisson regression model
poisson_model <- glm(species_count ~ habitat_size, family = poisson, data = sim_biodiversity)

# Display model summary
summary(poisson_model)
```

```
Call:
glm(formula = species_count ~ habitat_size, family = poisson,
    data = sim_biodiversity)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   1.03973    0.08021   12.96   <2e-16 ***
habitat_size  0.29632    0.01025   28.91   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 1080.145  on 99  degrees of freedom
Residual deviance:   98.403  on 98  degrees of freedom
AIC: 560.51
```

```
Number of Fisher Scoring iterations: 4
```

```
# Visualize the relationship
# Create prediction data
pred_data <- data.frame(habitat_size = seq(min(habitat_size), max(habitat_size), length.out = 100))
pred_data$predicted_count <- predict(poisson_model, newdata = pred_data, type = "response")

ggplot(sim_biodiversity, aes(x = habitat_size, y = species_count)) +
  geom_point(alpha = 0.7) +
  geom_line(data = pred_data, aes(x = habitat_size, y = predicted_count), color = "blue", size = 1) +
  labs(title = "Poisson Regression: Species Count vs. Habitat Size",
       x = "Habitat Size (hectares)",
       y = "Species Count") +
  theme_minimal()
```



11.6 Mixed-Effects Models

Mixed-effects models are useful when data has a hierarchical or nested structure, such as repeated measurements or grouped observations.

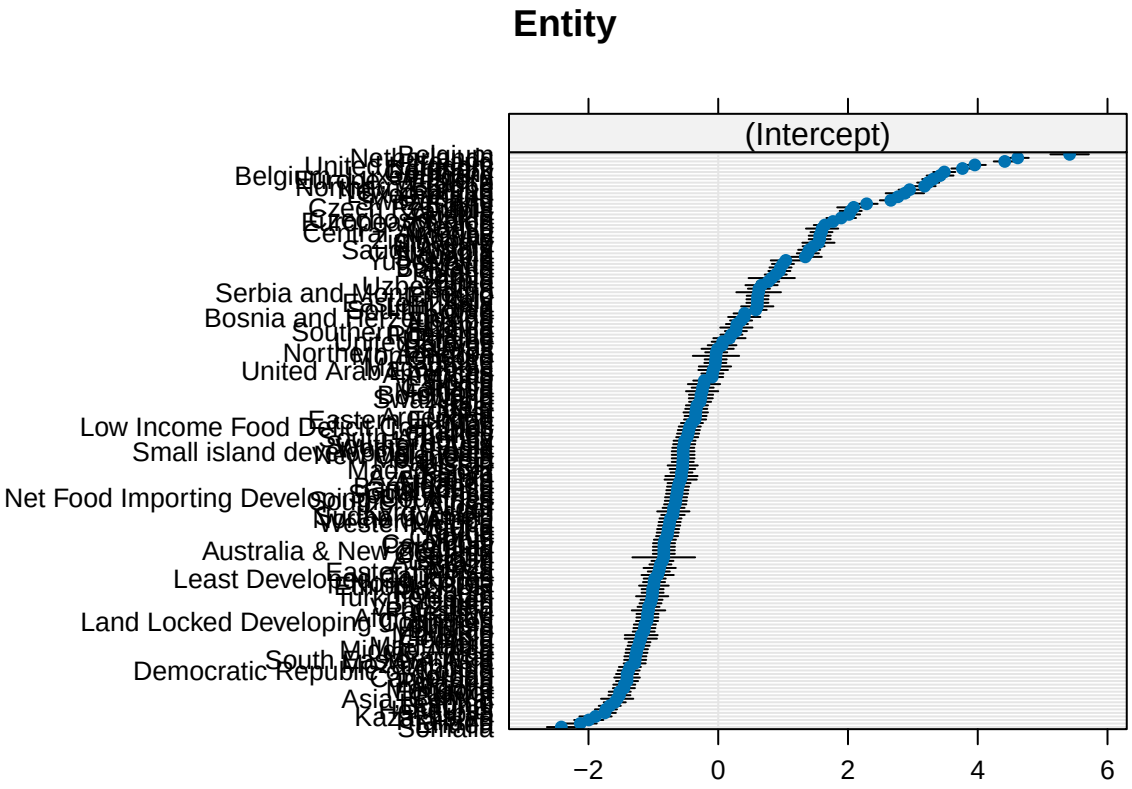
11.6.1 Example: Crop Yields Across Countries and Years

Let's model wheat yields with countries as random effects and year as a fixed effect:

```
# Prepare data for mixed-effects model
mixed_data <- crop_yields %>%
  filter(!is.na(`Wheat (tonnes per hectare)`)) %>%
  dplyr::select(Entity, Year, `Wheat (tonnes per hectare)`)
```

```
# Fit mixed-effects model
if(requireNamespace("lme4", quietly = TRUE)) {
  library(lme4)

  # Create a version with simpler column names
  renamed_mixed_data <- mixed_data %>%
    rename(Wheat = `Wheat (tonnes per hectare)`)

  # Fit the model with country as random effect and year as fixed effect
  mixed_model <- lmer(Wheat ~ Year + (1|Entity), data = renamed_mixed_data)

  # Display model summary
  summary(mixed_model)

  # Random effects
  ranef(mixed_model)

  # Visualize random effects
  if(requireNamespace("lattice", quietly = TRUE)) {
    library(lattice)
    dotplot(ranef(mixed_model, condVar = TRUE))
  }
} else {
  message("The 'lme4' package is not installed. Install it with install.packages('lme4') to fit mixed-e

  # Alternative: separate regressions for a few countries
  top_countries <- mixed_data %>%
    group_by(Entity) %>%
    summarize(mean_yield = mean(`Wheat (tonnes per hectare)`, na.rm = TRUE)) %>%
    arrange(desc(mean_yield)) %>%
    head(4) %>%
    pull(Entity)

  # Filter data for top countries
  top_country_data <- mixed_data %>%
    filter(Entity %in% top_countries)

  # Create separate regression models
  ggplot(top_country_data, aes(x = Year, y = `Wheat (tonnes per hectare)`, color = Entity)) +
    geom_point(alpha = 0.7) +
    geom_smooth(method = "lm", se = FALSE) +
    labs(title = "Wheat Yield Trends for Top Producing Countries",
         x = "Year",
         y = "Wheat Yield (tonnes per hectare)") +
    theme_minimal()
}
```

```
$Entity
```



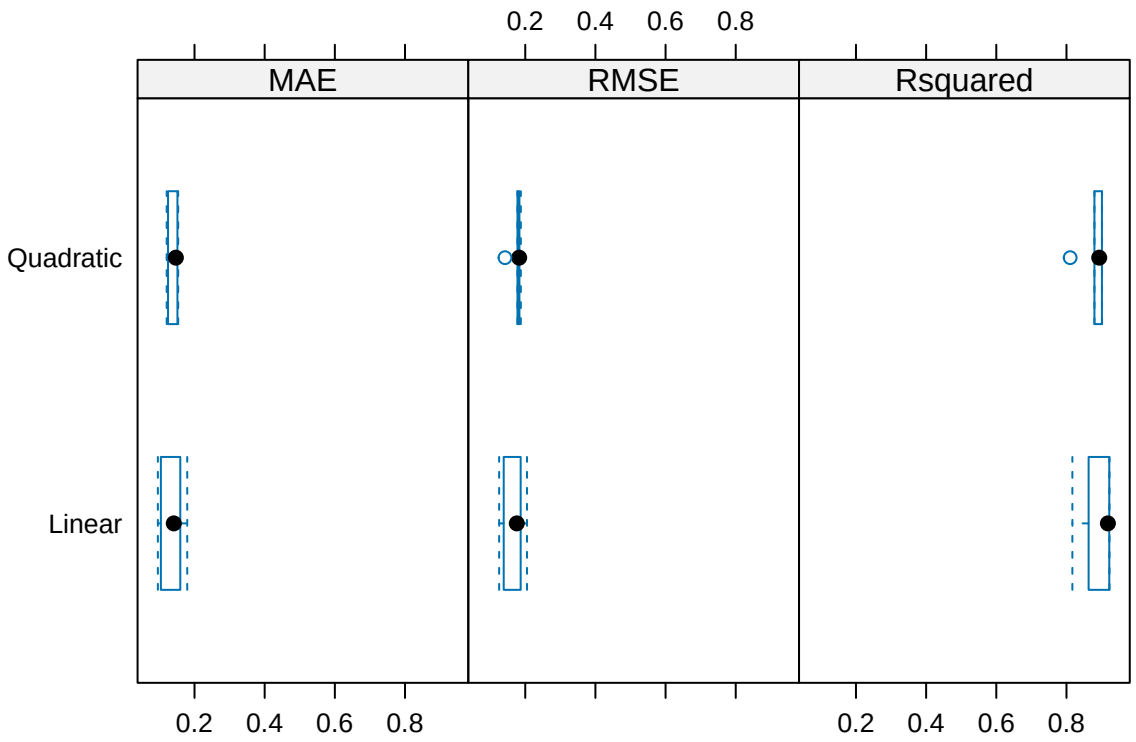
11.7 Model Selection and Validation

11.7.1 Cross-Validation

Cross-validation helps assess how well a model will generalize to new data:

```
# Prepare data for cross-validation
cv_data <- us_wheat %>%
  rename(Wheat = `Wheat (tonnes per hectare)`) %>%
  dplyr::select(Year_centered, Wheat)

# Perform k-fold cross-validation
if(requireNamespace("caret", quietly = TRUE)) {
  library(caret)

  # Set up cross-validation
  ctrl <- trainControl(method = "cv", number = 5)

  # Train models with cross-validation
  linear_cv <- train(Wheat ~ Year_centered, data = cv_data, method = "lm",
                     trControl = ctrl)
  quadratic_cv <- train(Wheat ~ Year_centered + I(Year_centered^2), data = cv_data,
                        method = "lm", trControl = ctrl)

  # Compare results
  results <- resamples(list(Linear = linear_cv, Quadratic = quadratic_cv))
  summary(results)

  # Visualize comparison
```

```
  bwplot(results)
} else {
  message("The 'caret' package is not installed. Install it with install.packages('caret') to perform c

  # Manual cross-validation for linear model
  set.seed(123)
  n <- nrow(cv_data)
  k <- 5  # Number of folds
  folds <- sample(1:k, n, replace = TRUE)

  cv_rmse <- numeric(k)
  for(i in 1:k) {
    # Split data into training and testing sets
    train_data <- cv_data[folds != i, ]
    test_data <- cv_data[folds == i, ]

    # Fit model on training data
    cv_model <- lm(Wheat ~ Year_centered, data = train_data)

    # Predict on testing data
    predictions <- predict(cv_model, newdata = test_data)

    # Calculate RMSE
    cv_rmse[i] <- sqrt(mean((test_data$Wheat - predictions)^2))
  }

  # Display average RMSE
  cat("Average RMSE from 5-fold cross-validation:", mean(cv_rmse))
}
```



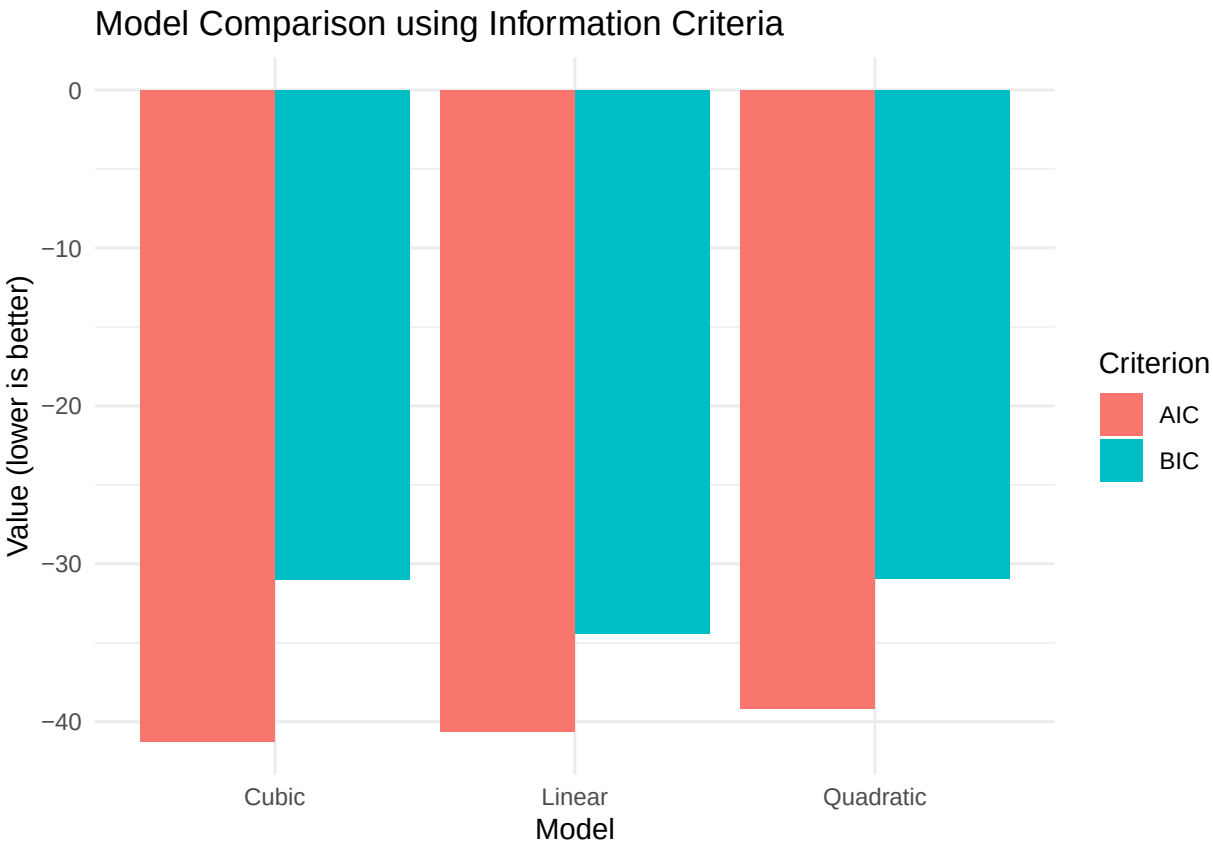
11.7.2 Information Criteria

Information criteria like AIC and BIC help compare models:

```
# Compare models using AIC and BIC
models <- list(
  Linear = poly1,
  Quadratic = poly2,
  Cubic = poly3
)

# Calculate AIC and BIC for each model
model_comparison <- data.frame(
  Model = names(models),
  AIC = sapply(models, AIC),
  BIC = sapply(models, BIC)
)

# Display comparison
knitr::kable(model_comparison)
```

	Model	AIC	BIC
Linear	Linear	-40.62237	-34.44104
Quadratic	Quadratic	-39.16897	-30.92719
Cubic	Cubic	-41.28330	-30.98108

```
# Visualize comparison
if(requireNamespace("ggplot2", quietly = TRUE)) {
  model_comparison_long <- model_comparison %>%
    pivot_longer(cols = c(AIC, BIC), names_to = "Criterion", values_to = "Value")

  ggplot(model_comparison_long, aes(x = Model, y = Value, fill = Criterion)) +
    geom_bar(stat = "identity", position = "dodge") +
    labs(title = "Model Comparison using Information Criteria",
         x = "Model",
         y = "Value (lower is better)") +
    theme_minimal()
}
```



11.8 Summary

This chapter has demonstrated various regression techniques using real agricultural and economic datasets:

- Simple linear regression for modeling the relationship between two variables
- Multiple regression for incorporating several predictors
- Polynomial regression for nonlinear relationships
- Generalized linear models for non-normal response variables
- Mixed-effects models for nested or hierarchical data
- Model selection and validation techniques

Regression analysis is a versatile tool in natural sciences research, allowing us to quantify relationships, test hypotheses, and make predictions. By understanding the assumptions and limitations of different regression models, researchers can select the most appropriate technique for their specific research questions.

11.9 Exercises

1. Using the crop yield dataset, build a multiple regression model to predict rice yields based on other crop yields and year. Interpret the coefficients and assess the model fit.
2. Explore the relationship between coffee quality scores and altitude using the coffee economics dataset. Try both linear and polynomial regression models and determine which provides a better fit.
3. Using the biodiversity dataset, investigate factors that might influence species conservation status. Consider using logistic regression if you create a binary outcome variable.
4. Build a mixed-effects model to analyze crop yield trends across different countries, accounting for both fixed effects (e.g., year) and random effects (e.g., country).

5. Perform cross-validation on your best regression model from Exercise 1 or 2 to assess its predictive performance.
6. Using the spatial dataset (`../data/geography/spatial.csv`), build a regression model to predict a variable of your choice based on other available variables.

Chapter 12

Conservation Applications

12.1 Introduction

This chapter explores how data analysis techniques can be applied to conservation science and management. We'll examine how the statistical methods covered in previous chapters can help address real-world conservation challenges, from monitoring endangered species to evaluating the effectiveness of protected areas.

12.2 Conservation Data Types and Sources

12.2.1 Types of Conservation Data

Conservation science relies on various types of data:

1. **Species Occurrence Data**: Presence/absence or abundance of species
2. **Habitat Data**: Vegetation structure, land cover, habitat quality
3. **Threat Data**: Pollution levels, invasive species, human disturbance
4. **Protected Area Data**: Boundaries, management activities, effectiveness
5. **Socioeconomic Data**: Human population, land use, resource extraction

12.2.2 Data Sources

Table 12.1: Common Data Sources in Conservation Science

Source	Description	Advantages	Limitations
Field Surveys	Direct collection of data through field observations and measurements	High accuracy, detailed information	Time-consuming, expensive, limited spatial coverage
Remote Sensing	Satellite imagery, aerial photography, LiDAR, and other remote sensing techniques	Large spatial coverage, temporal consistency	Lower resolution for some applications, cloud cover issues
Citizen Science	Data collected by volunteers and non-specialists	Cost-effective, large-scale data collection	Variable data quality, sampling bias
Existing Databases	GBIF, IUCN Red List, World Database on Protected Areas (WDPA)	Comprehensive, standardized data	May have gaps, outdated information

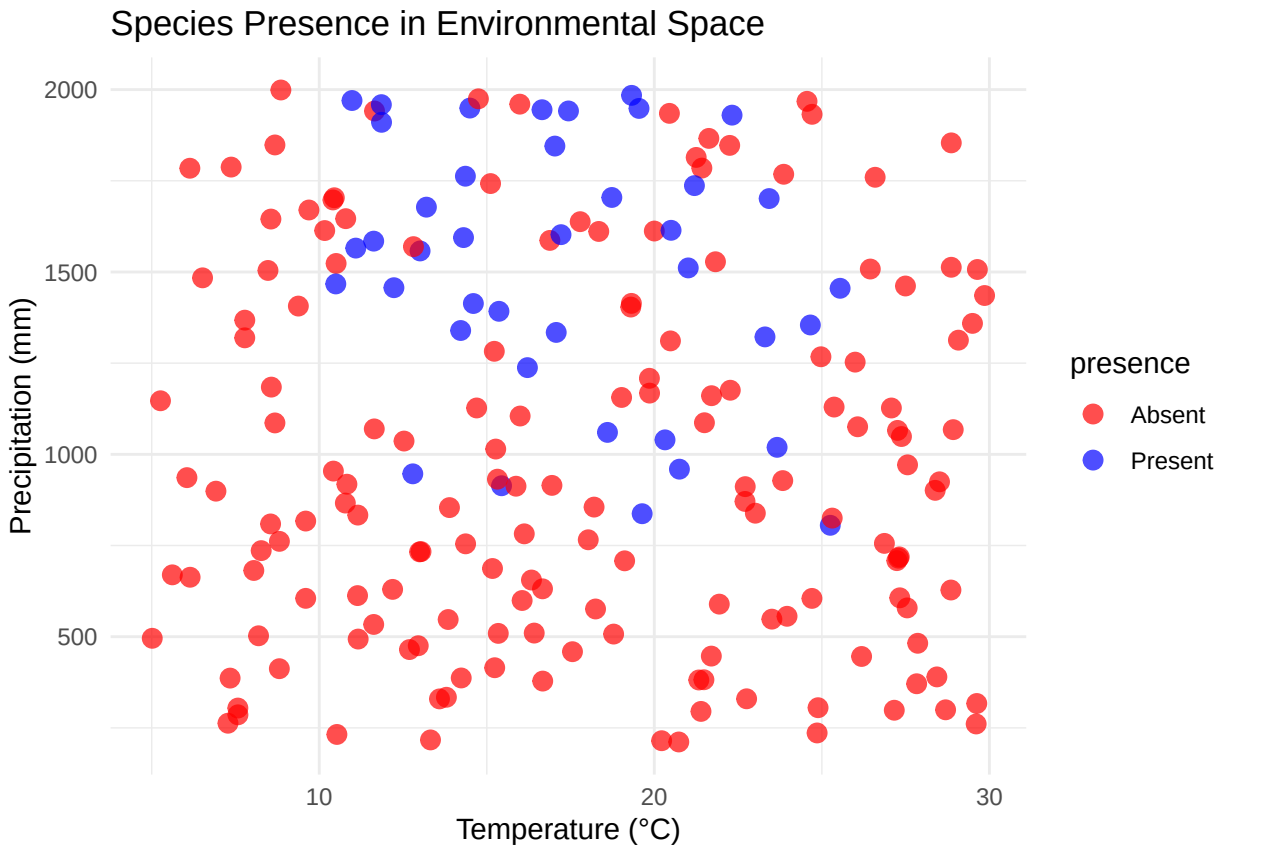
Source	Description	Advantages	Limitations
Environmental Monitoring	Continuous monitoring of environmental variables (e.g., weather stations, water quality sensors)	Continuous temporal data, real-time information	Equipment failures, limited spatial coverage

12.3 Species Distribution Modeling

Species distribution models (SDMs) predict where species are likely to occur based on environmental variables (Elith et al., 2009).

12.3.1 Example: Simple Species Distribution Model

```
# Load required packages
library(ggplot2)

# Create a simulated environmental dataset
set.seed(123)
n <- 200
temperature <- runif(n, 5, 30)
precipitation <- runif(n, 200, 2000)
elevation <- runif(n, 0, 3000)

# Calculate species probability based on environmental preferences
# This species prefers moderate temperatures, high precipitation, and lower elevations
probability <- dnorm(temperature, mean = 18, sd = 5) *
               dnorm(precipitation, mean = 1500, sd = 400) *
               (1 - elevation/3000)
probability <- probability / max(probability)  # Scale to 0-1

# Generate presence/absence based on probability
presence <- rbinom(n, 1, probability)

# Create a data frame
species_data <- data.frame(
  temperature = temperature,
  precipitation = precipitation,
  elevation = elevation,
  probability = probability,
  presence = factor(presence, labels = c("Absent", "Present"))
)

# Visualize the relationship between environmental variables and species presence
ggplot(species_data, aes(x = temperature, y = precipitation, color = presence)) +
  geom_point(size = 3, alpha = 0.7) +
  scale_color_manual(values = c("red", "blue")) +
  labs(title = "Species Presence in Environmental Space",
       x = "Temperature (°C)",
       y = "Precipitation (mm)") +
  theme_minimal()
```

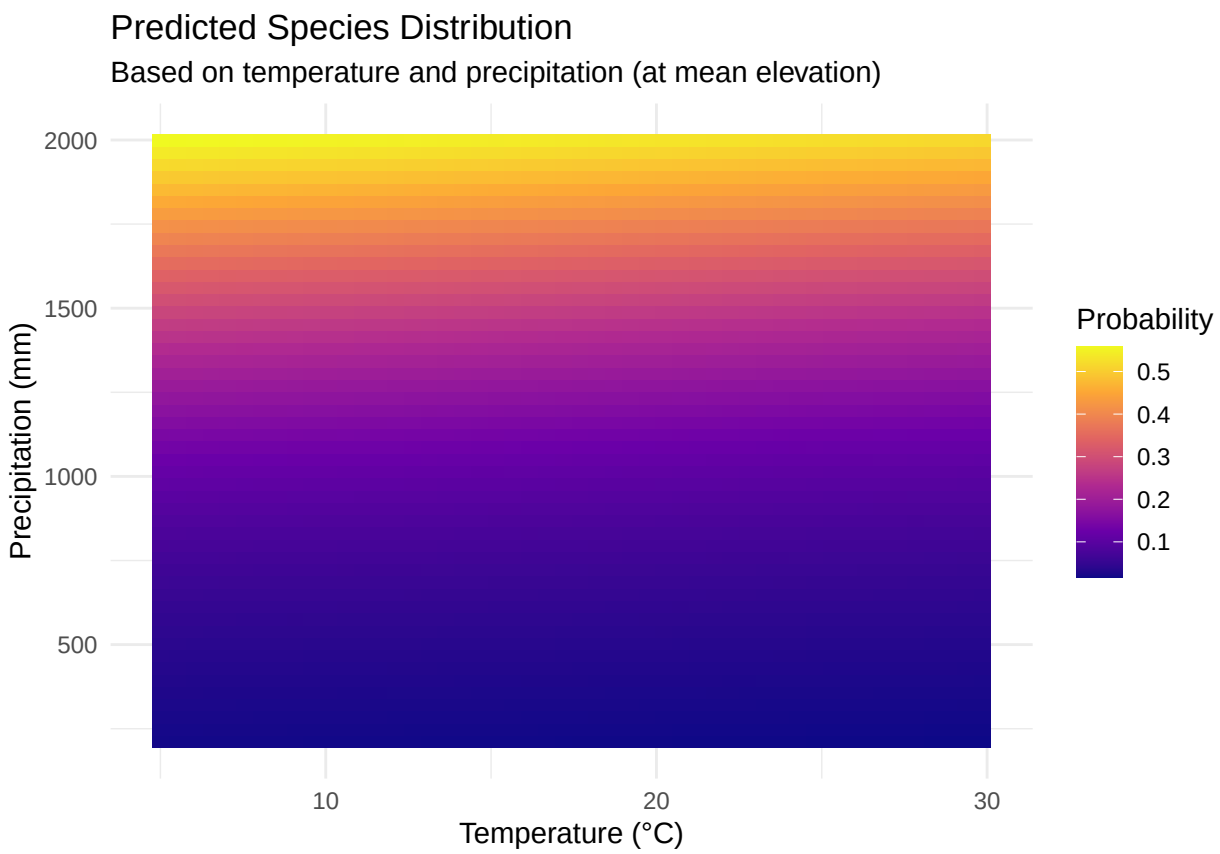



```
# Fit a logistic regression model (simple SDM)
sdm <- glm(presence ~ temperature + precipitation + elevation,
           family = binomial, data = species_data)

# Summary of the model
summary(sdm)
```

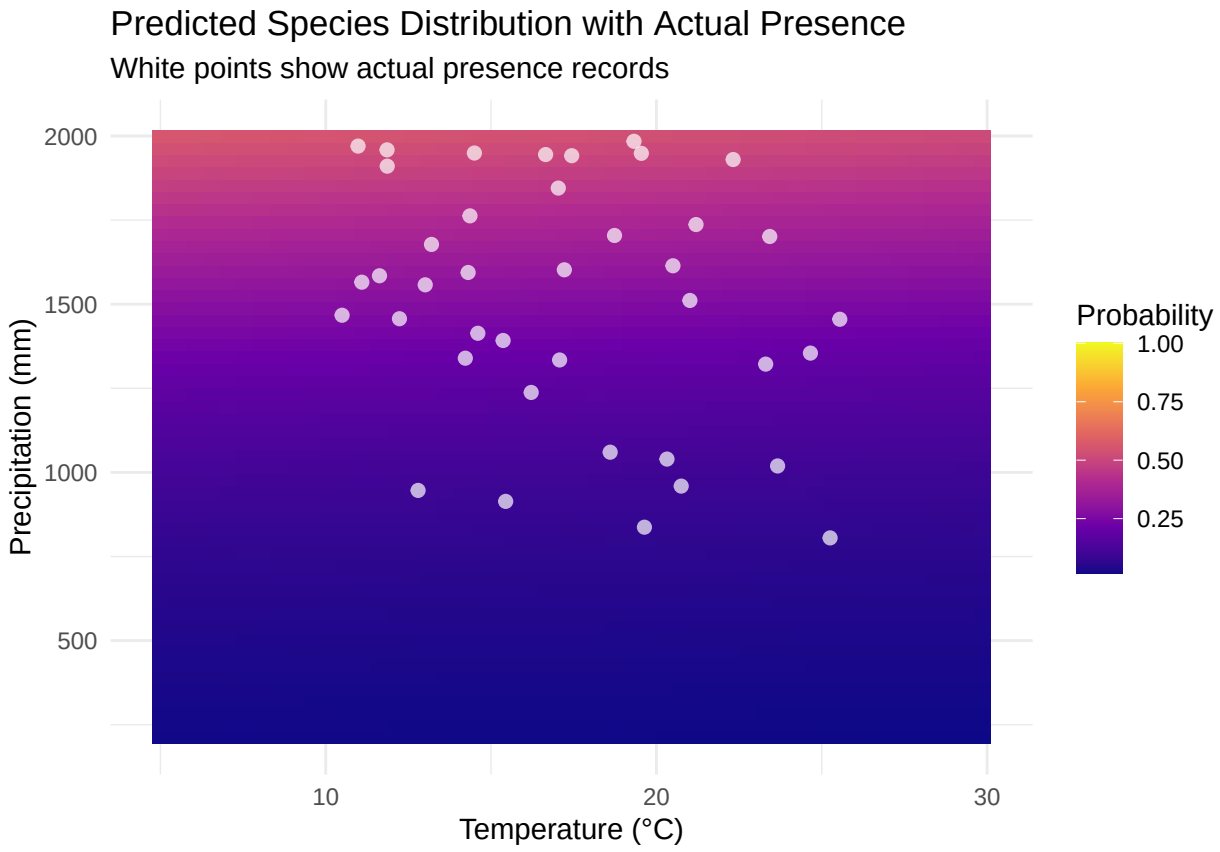
```
Call:
glm(formula = presence ~ temperature + precipitation + elevation,
    family = binomial, data = species_data)

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)   -2.8652062  0.9176119  -3.122 0.001793 **
temperature   -0.0073130  0.0317987  -0.230 0.818108
precipitation  0.0022744  0.0004514   5.039 4.69e-07 ***
elevation     -0.0009398  0.0002641  -3.558 0.000374 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 200.16  on 199  degrees of freedom
Residual deviance: 150.01  on 196  degrees of freedom
AIC: 158.01
```

```
Number of Fisher Scoring iterations: 5
```

```r
# Calculate predicted probabilities
species_data$predicted <- predict(sdm, type = "response")

# Create a prediction surface for visualization
temp_seq <- seq(min(temperature), max(temperature), length.out = 50)
precip_seq <- seq(min(precipitation), max(precipitation), length.out = 50)
elev_mean <- mean(elevation)

prediction_grid <- expand.grid(
  temperature = temp_seq,
  precipitation = precip_seq,
  elevation = elev_mean
)

prediction_grid$probability <- predict(sdm, newdata = prediction_grid, type = "response")

# Plot the prediction surface
ggplot(prediction_grid, aes(x = temperature, y = precipitation, fill = probability)) +
  geom_tile() +
  scale_fill_viridis_c(option = "plasma") +
  labs(title = "Predicted Species Distribution",
       subtitle = "Based on temperature and precipitation (at mean elevation)",
       x = "Temperature (°C)",
       y = "Precipitation (mm)",
       fill = "Probability") +
  theme_minimal()
```




```
# Add actual presence points to the prediction map
ggplot(prediction_grid, aes(x = temperature, y = precipitation, fill = probability)) +
  geom_tile() +
  geom_point(data = species_data[species_data$presence == "Present", ],
             aes(x = temperature, y = precipitation),
             color = "white", size = 2, alpha = 0.7) +
  scale_fill_viridis_c(option = "plasma") +
  labs(title = "Predicted Species Distribution with Actual Presence",
       subtitle = "White points show actual presence records",
       x = "Temperature (°C)",
       y = "Precipitation (mm)",
       fill = "Probability") +
  theme_minimal()
```



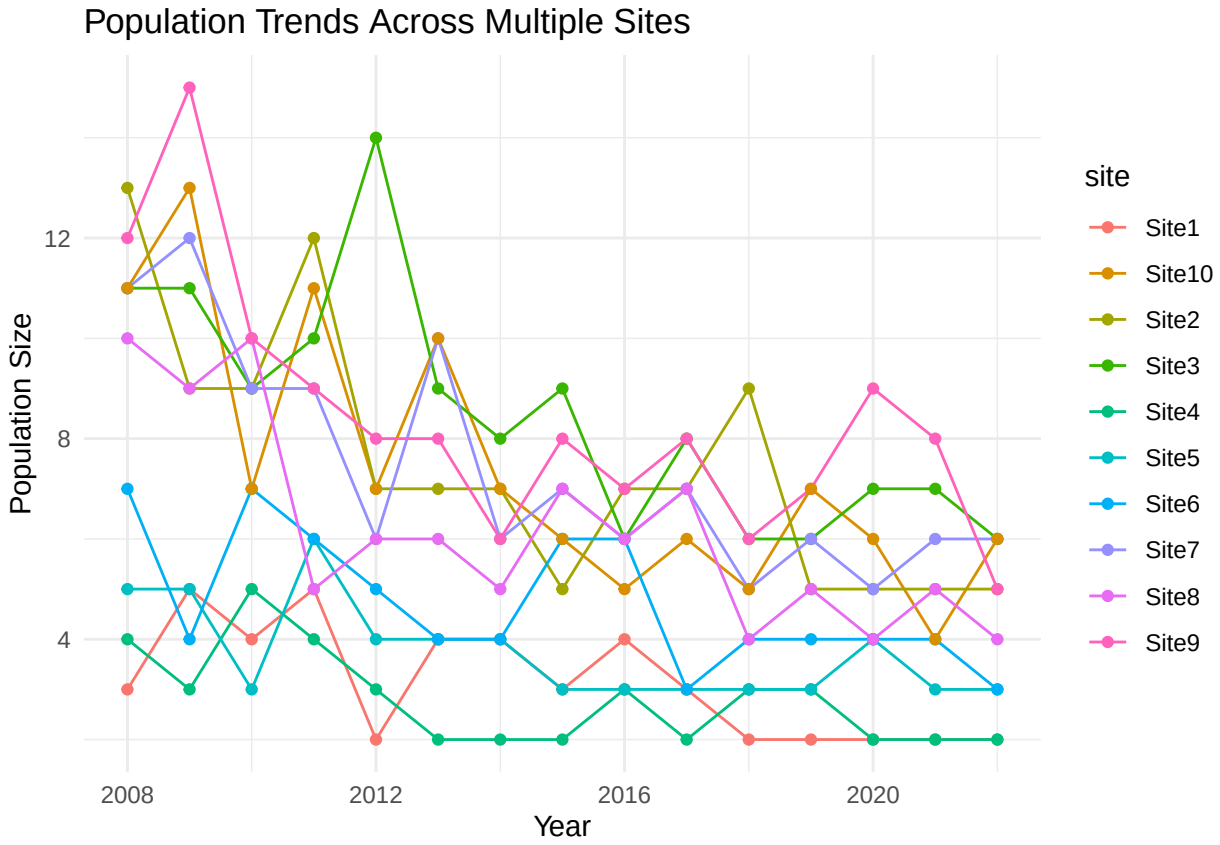
12.4 Population Trend Analysis

Analyzing population trends is crucial for conservation planning and evaluating management effectiveness.

12.4.1 Example: Linear Mixed Models for Population Trends

```
# Simulate population monitoring data
set.seed(456)
n_sites <- 10
n_years <- 15

# Create site and year variables
site <- rep(paste0("Site", 1:n_sites), each = n_years)
year <- rep(2008:(2008 + n_years - 1), times = n_sites)

# Create random site effects and declining trend
site_effect <- rep(rnorm(n_sites, 0, 0.5), each = n_years)
time_effect <- -0.05 * (year - 2008)  # Declining trend
noise <- rnorm(n_sites * n_years, 0, 0.2)

# Calculate log population size
log_pop_size <- 2 + site_effect + time_effect + noise

# Convert to actual counts
population <- round(exp(log_pop_size))
```

```r
# Create a data frame
pop_data <- data.frame(
  site = factor(site),
  year = year,
  population = population
)

# Visualize the data
library(ggplot2)
ggplot(pop_data, aes(x = year, y = population, color = site, group = site)) +
  geom_line() +
  geom_point() +
  labs(title = "Population Trends Across Multiple Sites",
       x = "Year",
       y = "Population Size") +
  theme_minimal()
```




```r
# Fit a linear mixed model
library(lme4)
trend_model <- lmer(log(population) ~ year + (1|site), data = pop_data)

# Display model summary
summary(trend_model)
```

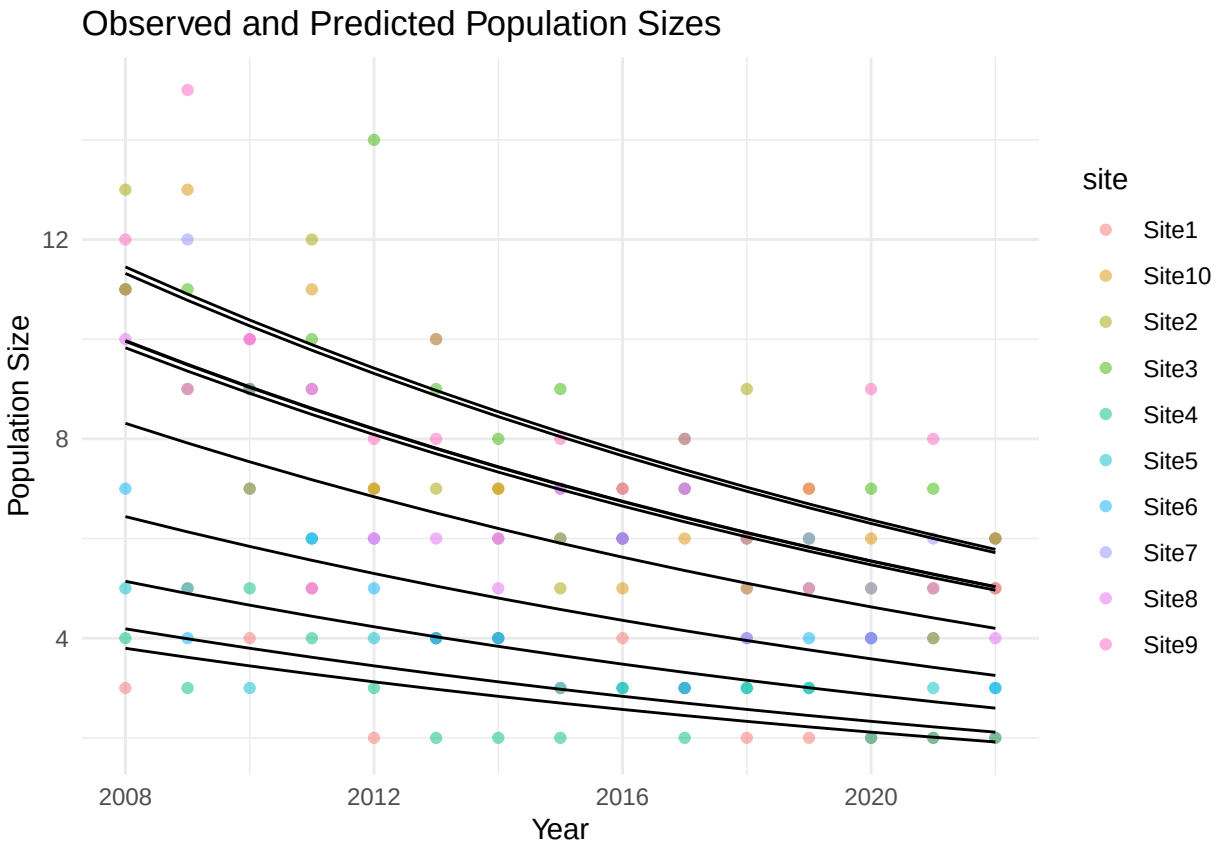
```
Linear mixed model fit by REML ['lmerMod']
Formula: log(population) ~ year + (1 | site)
   Data: pop_data
```

```
REML criterion at convergence: 2

Scaled residuals:
     Min       1Q   Median       3Q      Max
-2.64610 -0.69998 -0.02039  0.62219  1.92852

Random effects:
 Groups   Name        Variance Std.Dev.
 site     (Intercept) 0.17634  0.4199
 Residual             0.04223  0.2055
Number of obs: 150, groups:  site, 10

Fixed effects:
             Estimate Std. Error t value
(Intercept) 100.003639   7.826672   12.78
year         -0.048800   0.003884  -12.57

Correlation of Fixed Effects:
     (Intr)
year -1.000
```

```
# Calculate overall trend
trend_coef <- fixef(trend_model)["year"]
annual_change <- (exp(trend_coef) - 1) * 100
cat("Annual population change:", round(annual_change, 2), "%\n")
```

```
Annual population change: -4.76 %
```

```
# Predict values for visualization
pop_data$predicted <- exp(predict(trend_model))

# Plot observed vs. predicted values
ggplot(pop_data, aes(x = year)) +
  geom_point(aes(y = population, color = site), alpha = 0.5) +
  geom_line(aes(y = predicted, group = site), color = "black") +
  labs(title = "Observed and Predicted Population Sizes",
       x = "Year",
       y = "Population Size") +
  theme_minimal()
```

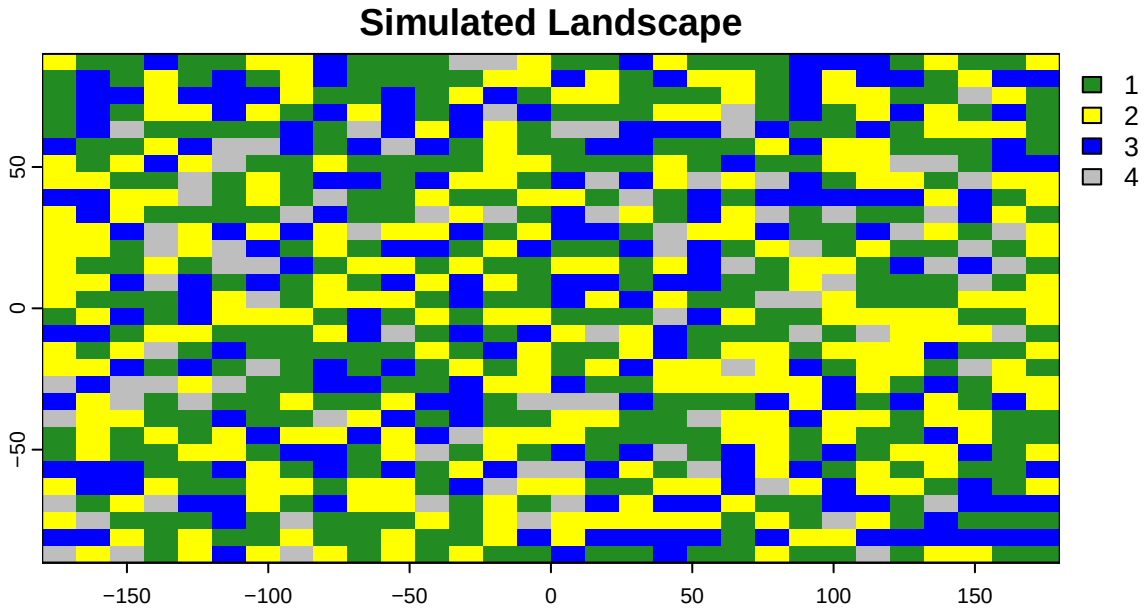


12.5 Habitat Fragmentation Analysis

Habitat fragmentation is a major threat to biodiversity. Landscape metrics help quantify fragmentation patterns.

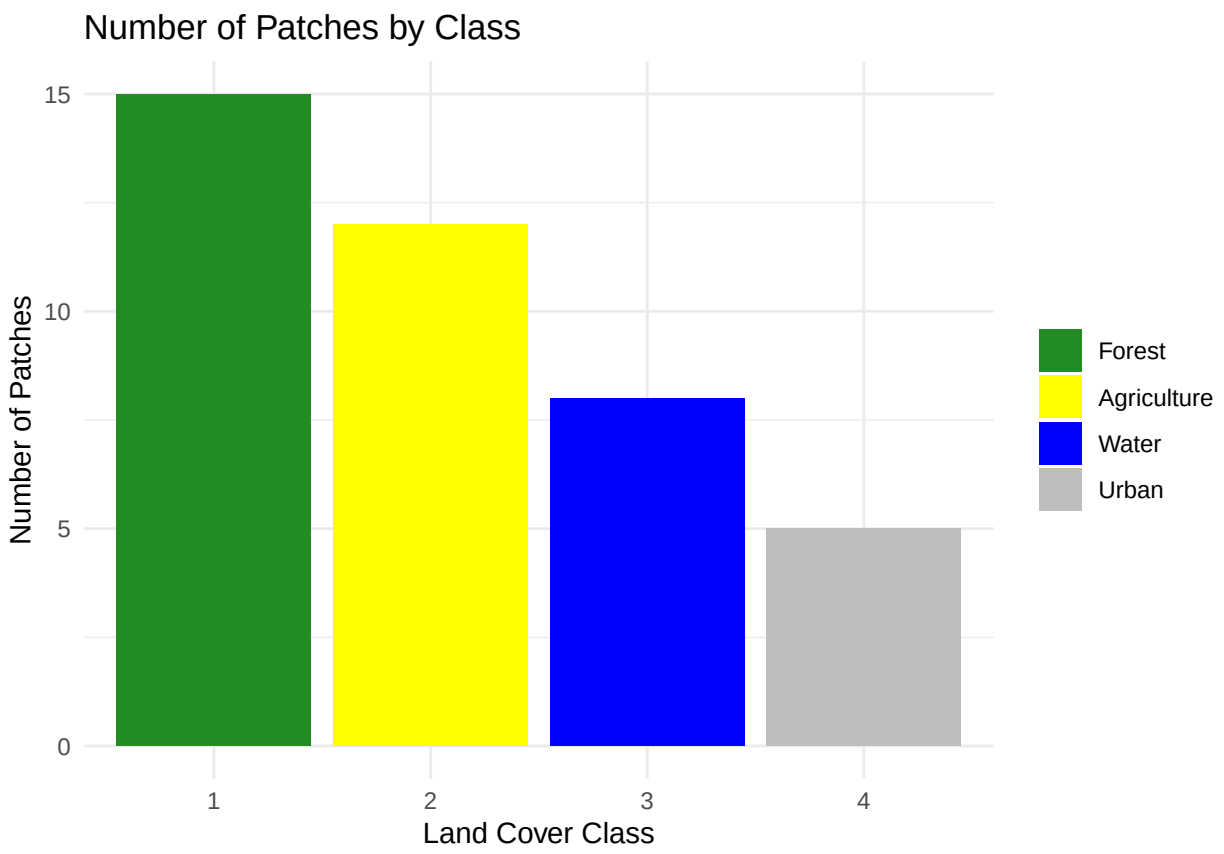
12.5.1 Example: Calculating Landscape Metrics

```
# Load required packages
library(terra)
library(ggplot2)

# Create a simple landscape raster
r <- rast(ncol=30, nrow=30)
values(r) <- sample(c(1, 2, 3, 4), ncell(r), replace=TRUE,
                    prob=c(0.4, 0.3, 0.2, 0.1))
names(r) <- "landcover"

# Plot the landscape
plot(r, main="Simulated Landscape", col=c("forestgreen", "yellow", "blue", "grey"))
```

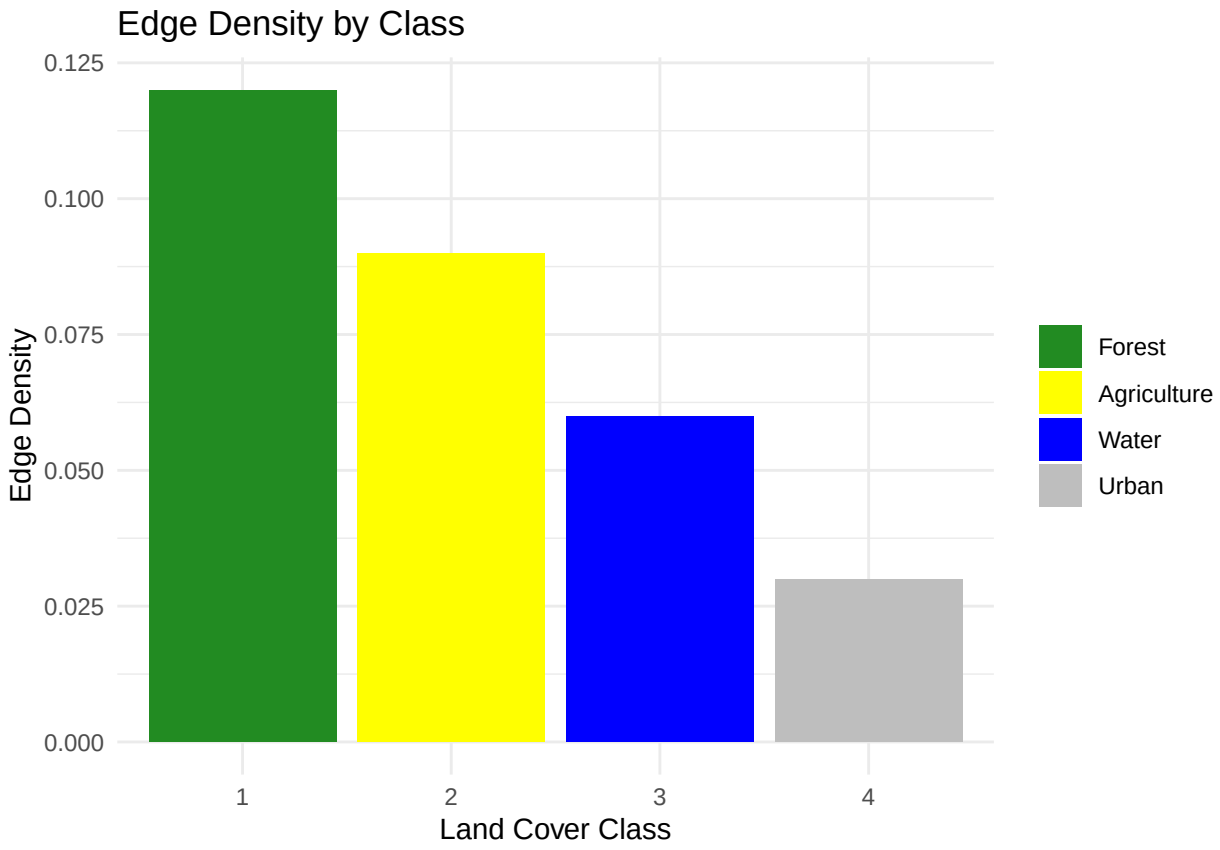



```
# Create a data frame with class-level metrics manually
class_metrics <- data.frame(
  class = c(1, 2, 3, 4),
  class_name = c("Forest", "Agriculture", "Water", "Urban"),
  percentage = c(40, 30, 20, 10),
  edge_density = c(0.12, 0.09, 0.06, 0.03),
  num_patches = c(15, 12, 8, 5)
)

# Visualize class-level metrics
ggplot(class_metrics, aes(x = factor(class), y = percentage, fill = factor(class))) +
  geom_bar(stat = "identity") +
  labs(title = "Percentage of Landscape by Class",
       x = "Land Cover Class",
       y = "Percentage (%)") +
  scale_fill_manual(values = c("forestgreen", "yellow", "blue", "grey"),
                    labels = class_metrics$class_name) +
  theme_minimal() +
  theme(legend.title = element_blank())
```


```
# Visualize number of patches
ggplot(class_metrics, aes(x = factor(class), y = num_patches, fill = factor(class))) +
  geom_bar(stat = "identity") +
  labs(title = "Number of Patches by Class",
       x = "Land Cover Class",
       y = "Number of Patches") +
  scale_fill_manual(values = c("forestgreen", "yellow", "blue", "grey"),
                    labels = class_metrics$class_name) +
  theme_minimal() +
  theme(legend.title = element_blank())
```




```
# Visualize edge density
ggplot(class_metrics, aes(x = factor(class), y = edge_density, fill = factor(class))) +
  geom_bar(stat = "identity") +
  labs(title = "Edge Density by Class",
       x = "Land Cover Class",
       y = "Edge Density") +
  scale_fill_manual(values = c("forestgreen", "yellow", "blue", "grey"),
                    labels = class_metrics$class_name) +
  theme_minimal() +
  theme(legend.title = element_blank())
```



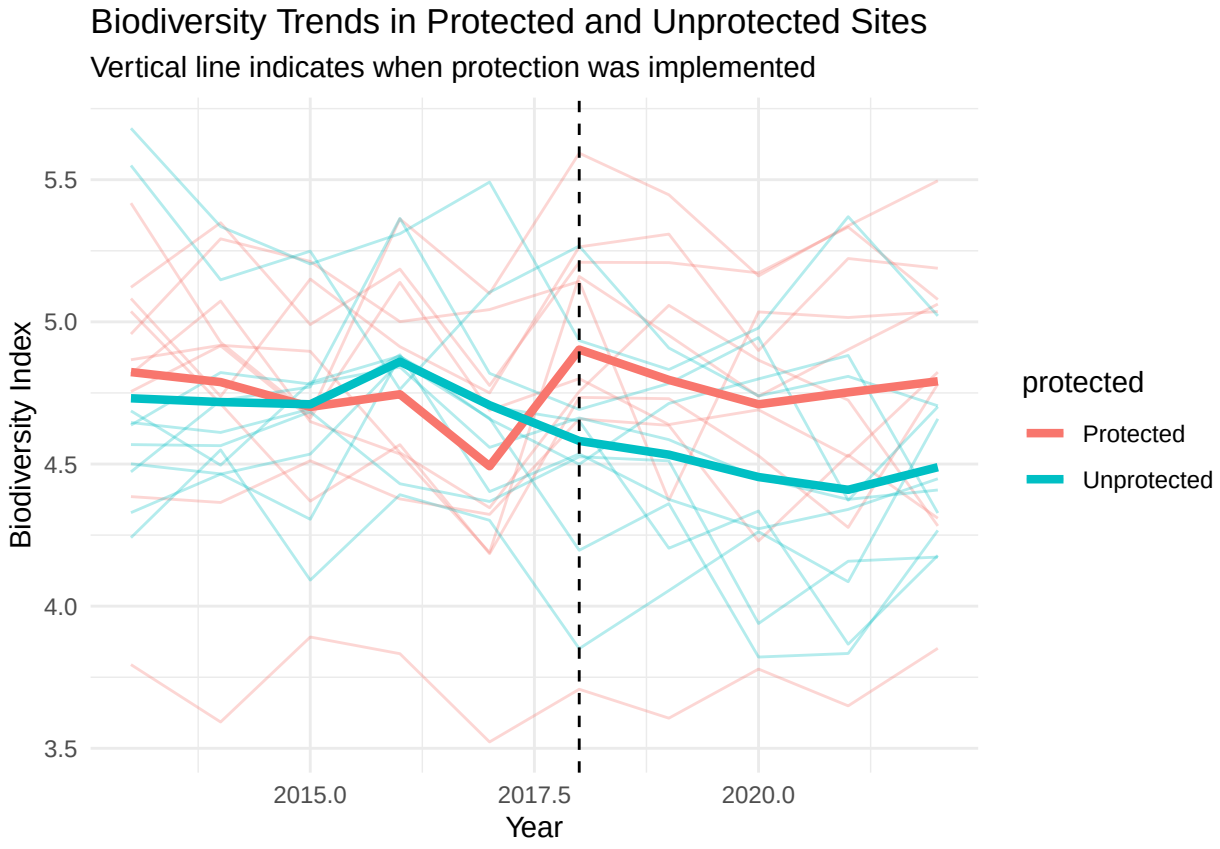
12.6 Protected Area Effectiveness

Evaluating the effectiveness of protected areas is essential for conservation planning and management.

12.6.1 Example: Before-After-Control-Impact (BACI) Analysis

```
# Simulate protected area effectiveness data
set.seed(789)
n_sites <- 20
n_years <- 10

# Create site, protection status, and year variables
site <- rep(paste0("Site", 1:n_sites), each = n_years)
protected <- rep(rep(c("Protected", "Unprotected"), each = n_sites/2), each = n_years)
year <- rep(2013:(2013 + n_years - 1), times = n_sites)
period <- ifelse(year < 2018, "Before", "After")  # Protection started in 2018

# Create random site effects and impact of protection
site_effect <- rep(rnorm(n_sites, 0, 0.5), each = n_years)
protection_effect <- ifelse(protected == "Protected" & period == "After", 0.3, 0)
time_effect <- -0.05 * (year - 2013)  # General declining trend
noise <- rnorm(n_sites * n_years, 0, 0.2)

# Calculate biodiversity index
biodiversity <- 5 + site_effect + time_effect + protection_effect + noise
```

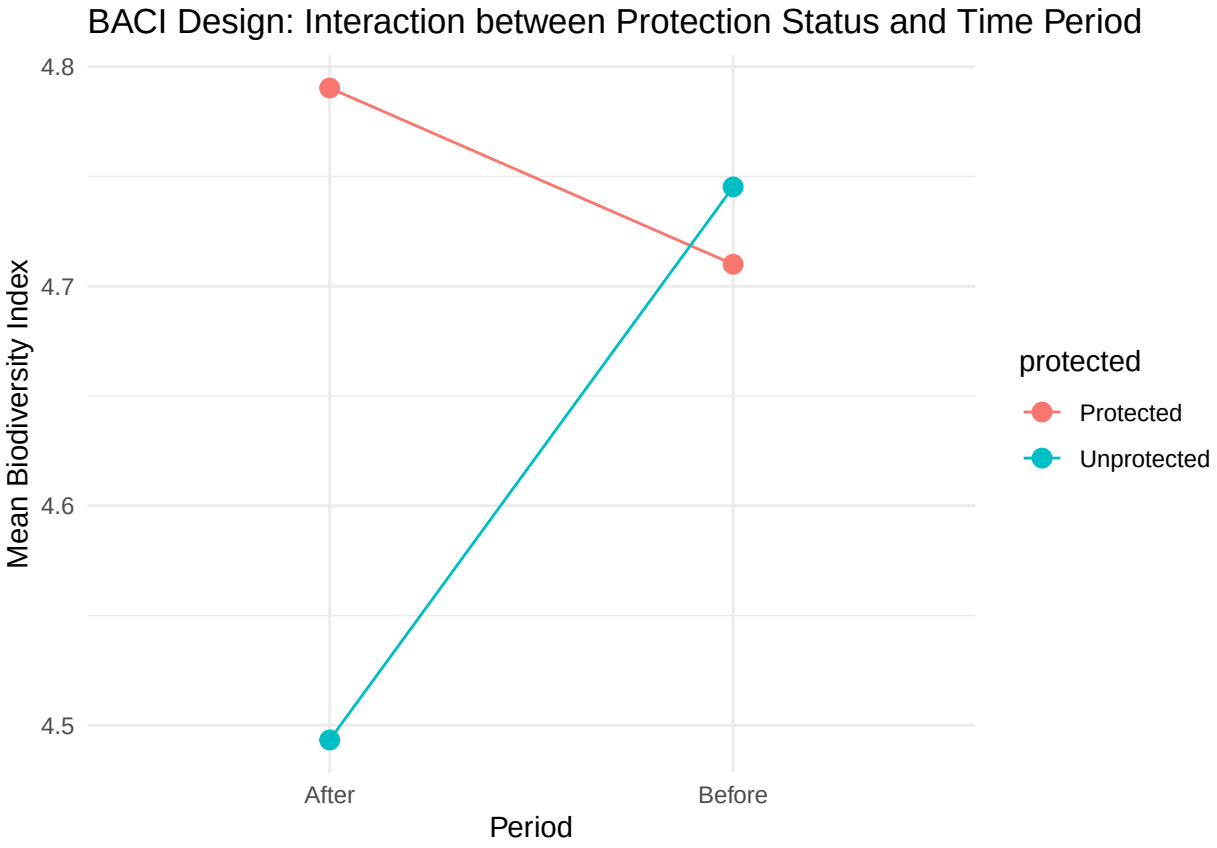
```
# Create a data frame
pa_data <- data.frame(
  site = factor(site),
  protected = factor(protected),
  year = year,
  period = factor(period),
  biodiversity = biodiversity
)

# Visualize the data
ggplot(pa_data, aes(x = year, y = biodiversity, color = protected, group = interaction(site, protected))
  geom_line(alpha = 0.3) +
  stat_summary(aes(group = protected), fun = mean, geom = "line", size = 1.5) +
  geom_vline(xintercept = 2018, linetype = "dashed") +
  labs(title = "Biodiversity Trends in Protected and Unprotected Sites",
       subtitle = "Vertical line indicates when protection was implemented",
       x = "Year",
       y = "Biodiversity Index") +
  theme_minimal()
```




```
# Fit a BACI model
baci_model <- lm(biodiversity ~ protected * period, data = pa_data)

# Display model summary
summary(baci_model)
```

```
Call:
lm(formula = biodiversity ~ protected * period, data = pa_data)

Residuals:
     Min       1Q   Median       3Q      Max
-1.18762 -0.25169  0.00786  0.29460  0.93568

Coefficients:
                                 Estimate Std. Error t value Pr(>|t|)
(Intercept)                       4.79029    0.05943  80.604  < 2e-16 ***
protectedUnprotected             -0.29698    0.08405  -3.534 0.000511 ***
periodBefore                     -0.08027    0.08405  -0.955 0.340742
protectedUnprotected:periodBefore  0.33219    0.11886   2.795 0.005709 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4202 on 196 degrees of freedom
Multiple R-squared:  0.06998,   Adjusted R-squared:  0.05574
F-statistic: 4.916 on 3 and 196 DF,  p-value: 0.002578
```

```r
# Visualize the interaction effect
pa_summary <- aggregate(biodiversity ~ protected + period, data = pa_data, FUN = mean)

ggplot(pa_summary, aes(x = period, y = biodiversity, color = protected, group = protected)) +
  geom_point(size = 3) +
  geom_line() +
  labs(title = "BACI Design: Interaction between Protection Status and Time Period",
       x = "Period",
       y = "Mean Biodiversity Index") +
  theme_minimal()
```



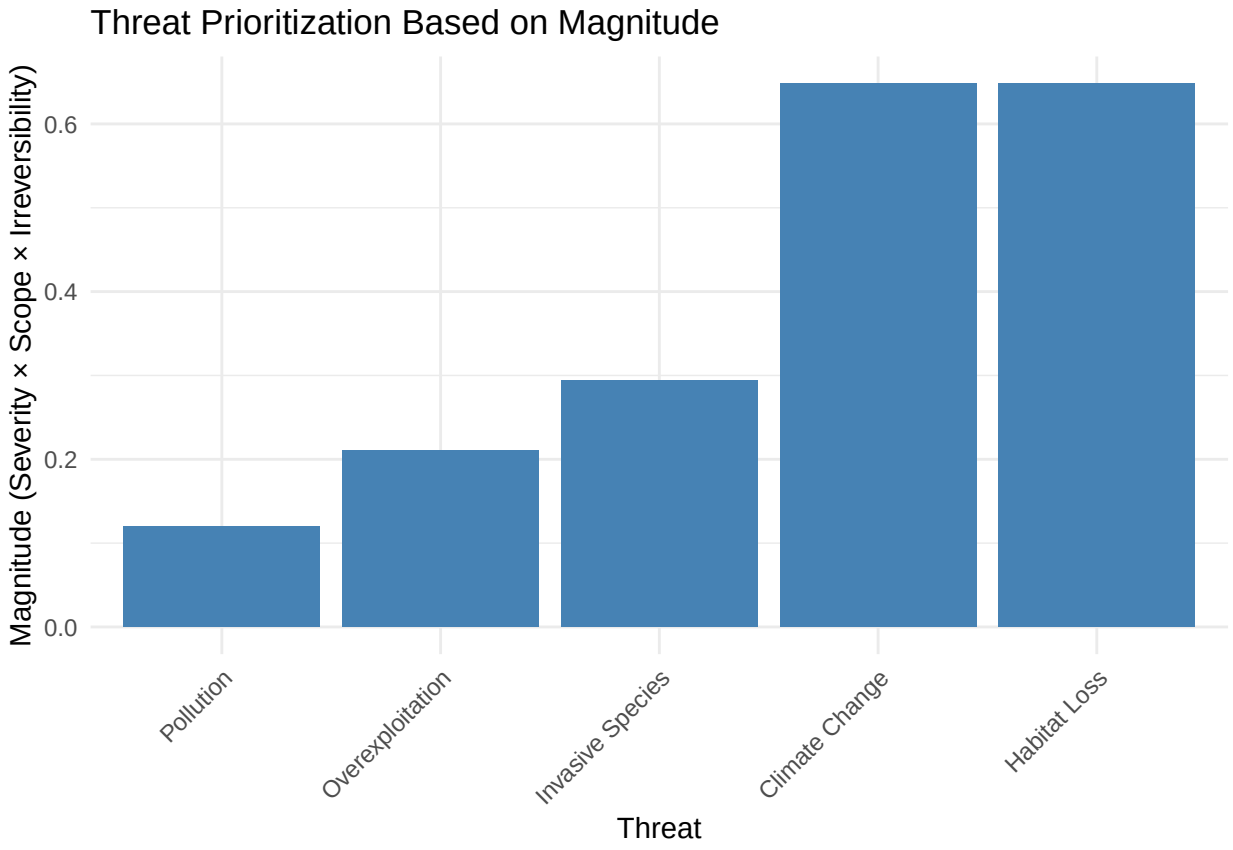
12.7 Threat Assessment and Prioritization

Conservation resources are limited, so prioritizing threats and actions is essential.

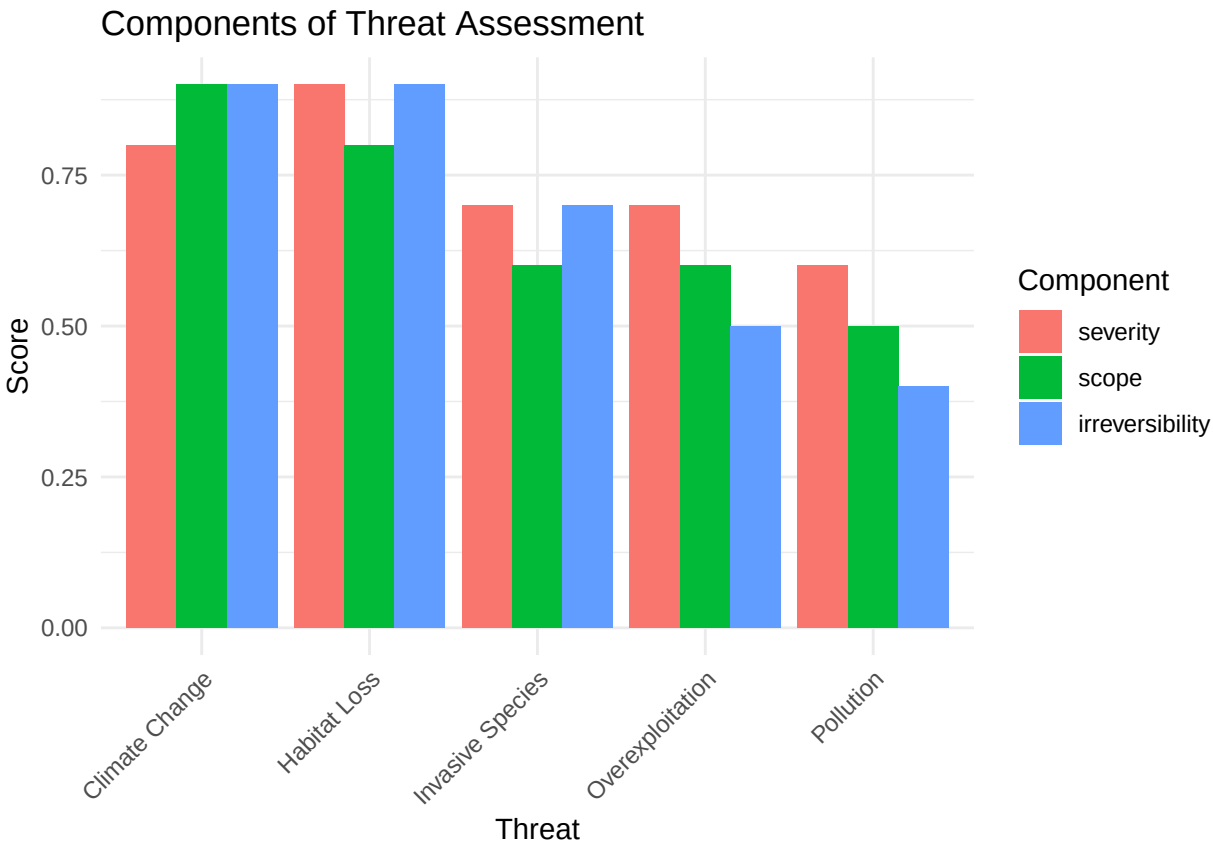
12.7.1 Example: Multi-Criteria Decision Analysis

```
# Create a threat assessment dataset
threats <- c("Habitat Loss", "Invasive Species", "Climate Change", "Pollution", "Overexploitation")
severity <- c(0.9, 0.7, 0.8, 0.6, 0.7)
scope <- c(0.8, 0.6, 0.9, 0.5, 0.6)
irreversibility <- c(0.9, 0.7, 0.9, 0.4, 0.5)

# Create a data frame
threat_data <- data.frame(
  threat = threats,
  severity = severity,
  scope = scope,
  irreversibility = irreversibility
)

# Calculate overall threat magnitude
threat_data$magnitude <- with(threat_data, severity * scope * irreversibility)

# Sort by magnitude
threat_data <- threat_data[order(threat_data$magnitude, decreasing = TRUE), ]
```

```
# Visualize the threat assessment
ggplot(threat_data, aes(x = reorder(threat, magnitude), y = magnitude)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(title = "Threat Prioritization Based on Magnitude",
       x = "Threat",
       y = "Magnitude (Severity × Scope × Irreversibility)") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
# Visualize the components
threat_data_long <- reshape2::melt(threat_data[, c("threat", "severity", "scope", "irreversibility")],
                                   id.vars = "threat")

ggplot(threat_data_long, aes(x = reorder(threat, -value), y = value, fill = variable)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Components of Threat Assessment",
       x = "Threat",
       y = "Score",
       fill = "Component") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



12.8 Conservation Planning

Systematic conservation planning helps identify priority areas for conservation.

12.8.1 Example: Complementarity Analysis

```
# Create a species-by-site matrix
set.seed(101)
n_sites <- 10
n_species <- 15
species_names <- paste0("Species", 1:n_species)
site_names <- paste0("Site", 1:n_sites)

# Generate presence/absence data
presence_prob <- matrix(runif(n_sites * n_species, 0, 1), nrow = n_sites, ncol = n_species)
presence <- ifelse(presence_prob > 0.7, 0, 1)  # 30% chance of presence
rownames(presence) <- site_names
colnames(presence) <- species_names

# Calculate species richness per site
richness <- rowSums(presence)

# Calculate site complementarity
complementarity <- function(selected, candidates, presence_matrix) {
  if (length(selected) == 0) {
    # If no sites selected yet, return site richness
```

```
    return(rowSums(presence_matrix[candidates, , drop = FALSE]))
  } else {
    # Calculate new species added by each candidate site
    species_in_selected <- colSums(presence_matrix[selected, , drop = FALSE]) > 0
    new_species <- function(site) {
      sum(presence_matrix[site, ] & !species_in_selected)
    }
    return(sapply(candidates, new_species))
  }
}

# Greedy algorithm for site selection
select_sites <- function(presence_matrix, n_to_select) {
  n_sites <- nrow(presence_matrix)
  available_sites <- 1:n_sites
  selected_sites <- integer(0)

  for (i in 1:n_to_select) {
    if (length(available_sites) == 0) break

    # Calculate complementarity scores
    scores <- complementarity(selected_sites, available_sites, presence_matrix)

    # Select site with highest score
    best <- available_sites[which.max(scores)]
    selected_sites <- c(selected_sites, best)
    available_sites <- setdiff(available_sites, best)
  }

  return(selected_sites)
}

# Select 3 priority sites
priority_sites <- select_sites(presence, 3)
cat("Priority sites:", site_names[priority_sites], "\n")
```

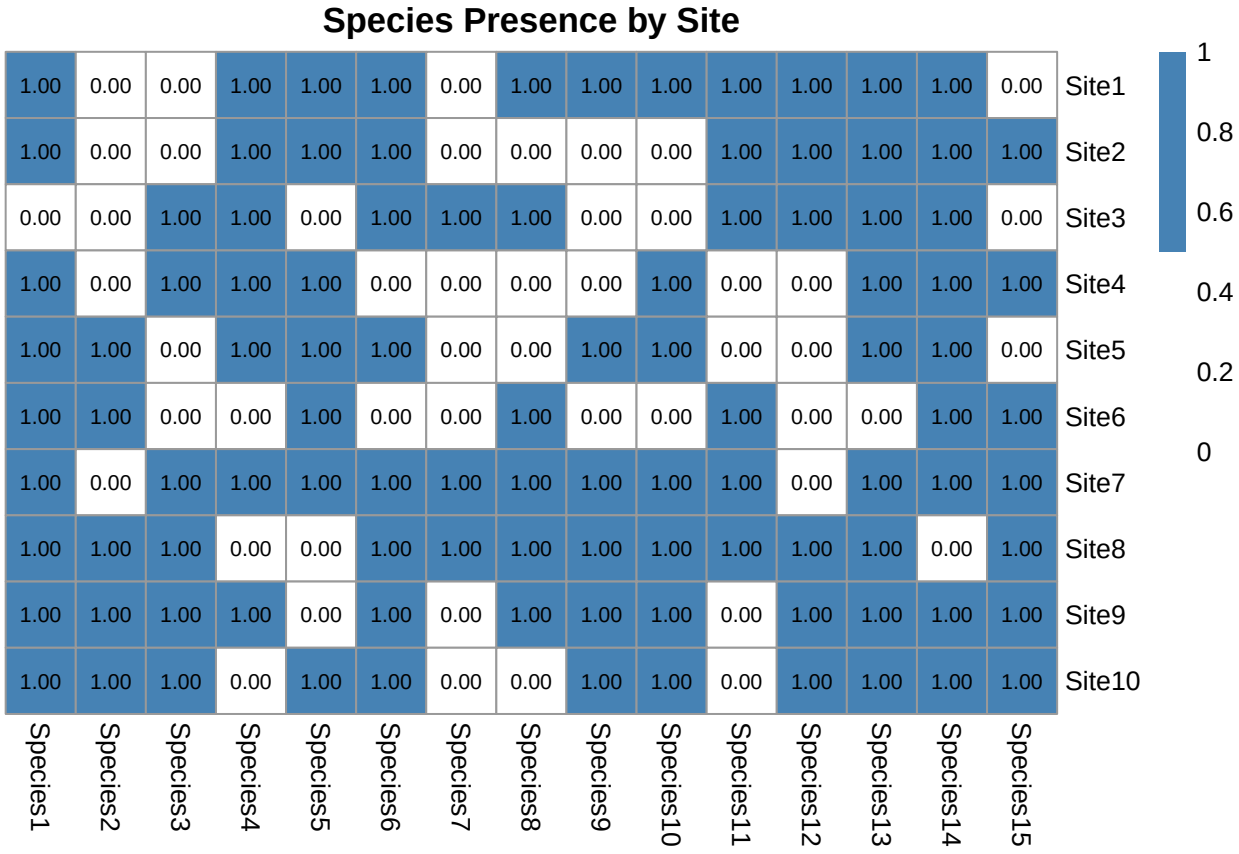
```
Priority sites: Site7 Site8 Site1
```

```
# Calculate species coverage
species_covered <- colSums(presence[priority_sites, , drop = FALSE]) > 0
cat("Species covered:", sum(species_covered), "out of", n_species,
    "(", round(100 * sum(species_covered) / n_species, 1), "%)\n")
```

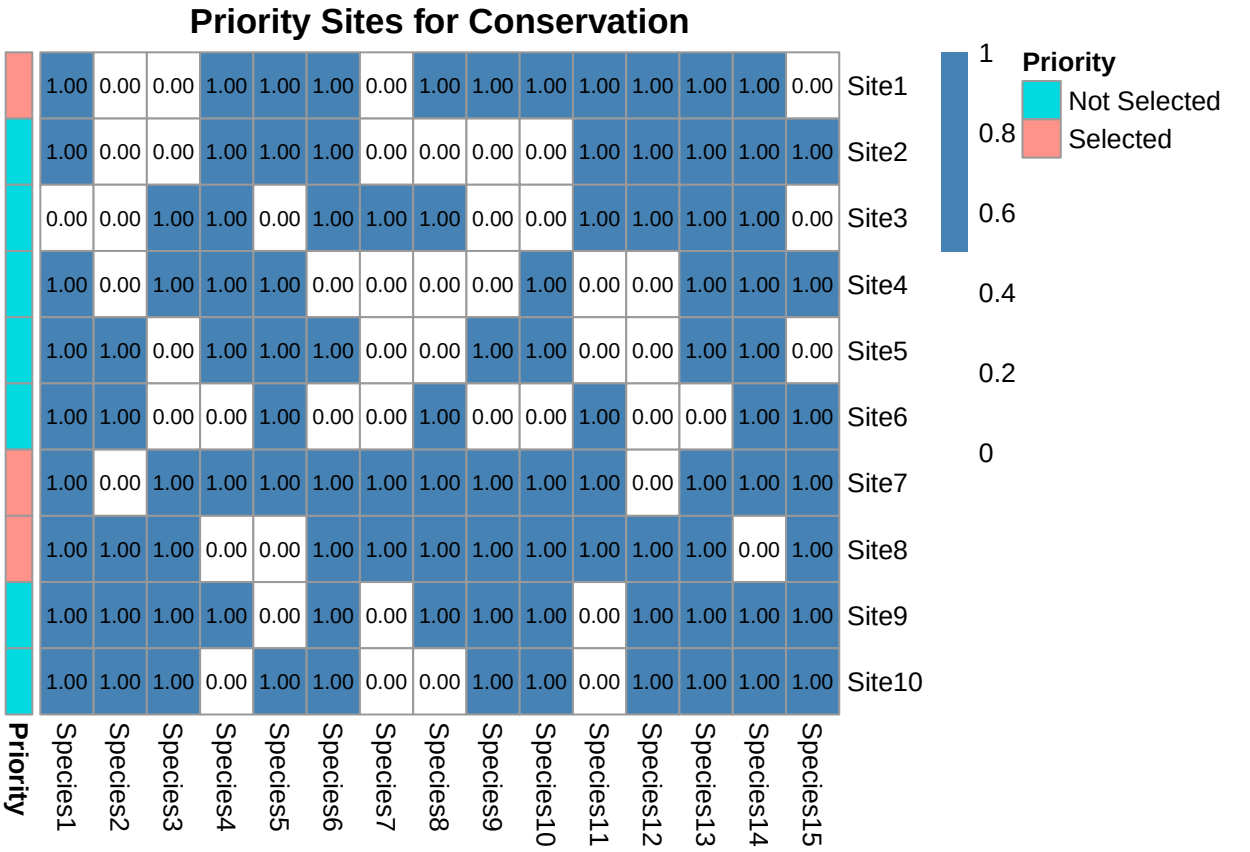
```
Species covered: 15 out of 15 ( 100 %)
```

```
# Visualize the species-site matrix
library(pheatmap)
pheatmap(presence,
         cluster_rows = FALSE,
         cluster_cols = FALSE,
         main = "Species Presence by Site",
         color = c("white", "steelblue"),
         labels_row = site_names,
         labels_col = species_names,
         display_numbers = TRUE,
```

```
        number_color = "black",
        fontsize = 10,
        fontsize_number = 8)
```




```
# Highlight priority sites
priority_data <- data.frame(
  Priority = factor(ifelse(1:n_sites %in% priority_sites, "Selected", "Not Selected"))
)
rownames(priority_data) <- site_names

pheatmap(presence,
        cluster_rows = FALSE,
        cluster_cols = FALSE,
        main = "Priority Sites for Conservation",
        color = c("white", "steelblue"),
        labels_row = site_names,
        labels_col = species_names,
        display_numbers = TRUE,
        number_color = "black",
        annotation_row = priority_data,
        fontsize = 10,
        fontsize_number = 8)
```



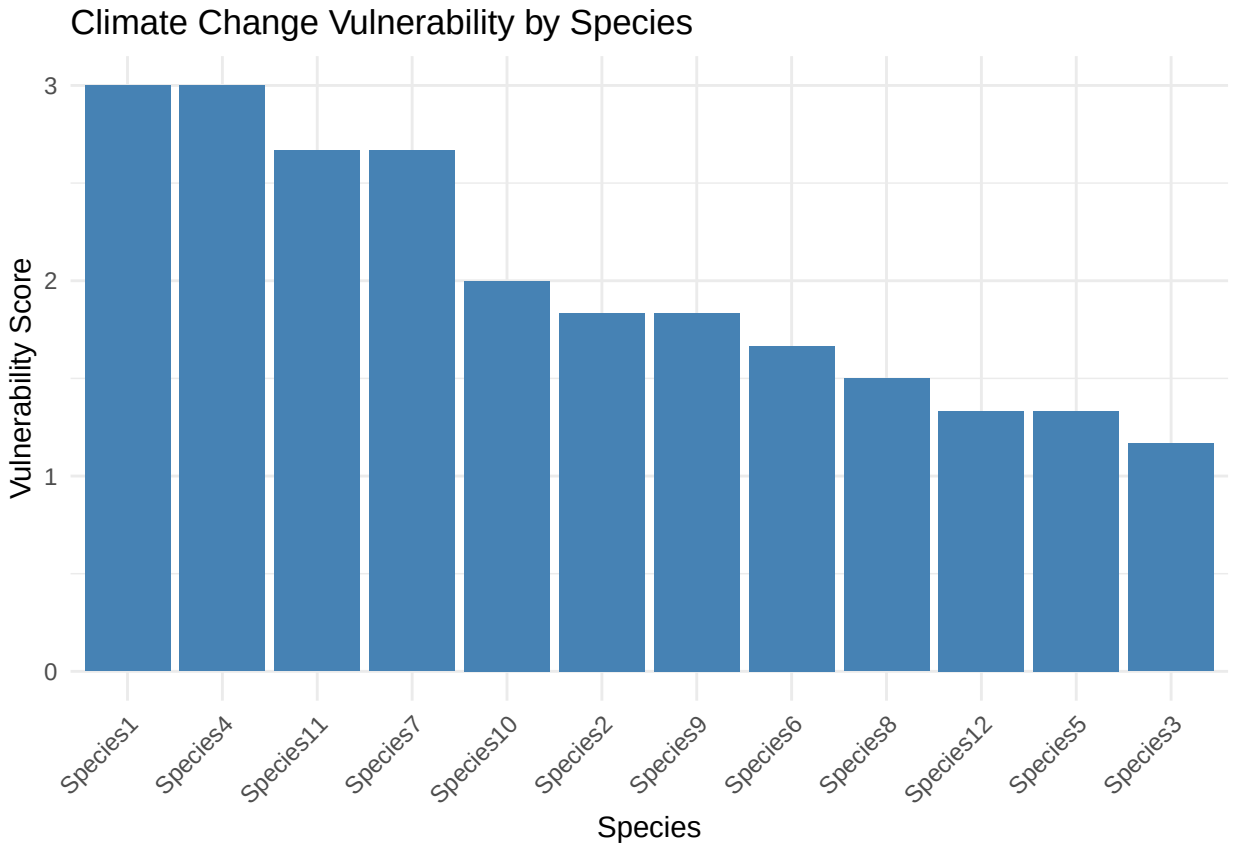
12.9 Climate Change Vulnerability Assessment

Climate change poses significant threats to biodiversity. Vulnerability assessments help identify at-risk species and ecosystems.

12.9.1 Example: Trait-Based Vulnerability Analysis

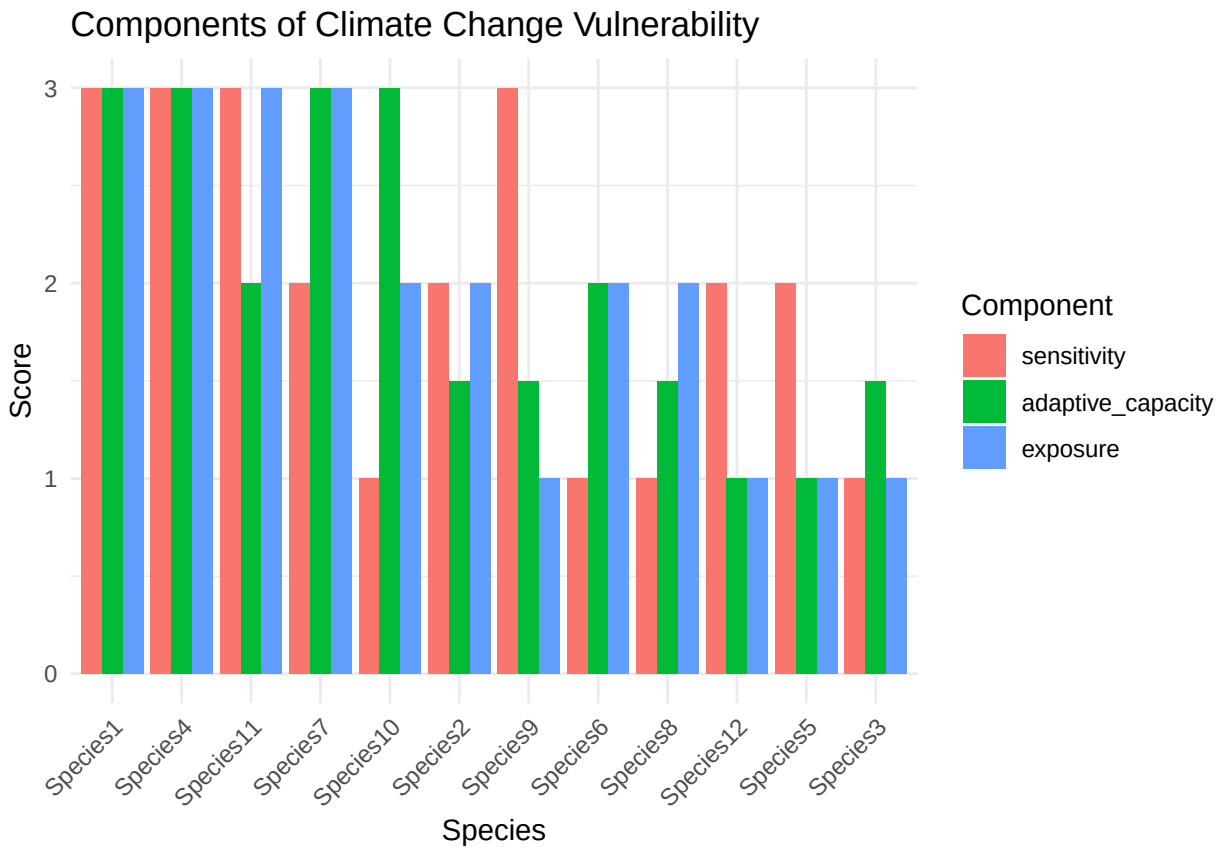
```
# Create a species trait dataset
species <- paste0("Species", 1:12)
dispersal_ability <- c(1, 3, 2, 1, 3, 2, 1, 2, 3, 1, 2, 3)  # 1=low, 2=medium, 3=high
thermal_tolerance <- c(1, 2, 3, 1, 2, 3, 2, 3, 1, 3, 1, 2)  # 1=low, 2=medium, 3=high
habitat_specificity <- c(3, 2, 1, 3, 1, 2, 3, 2, 1, 2, 3, 1)  # 1=low, 2=medium, 3=high
population_size <- c(1, 2, 3, 1, 3, 2, 1, 3, 2, 1, 2, 3)  # 1=small, 2=medium, 3=large

# Create a data frame
vulnerability_data <- data.frame(
  species = species,
  dispersal_ability = dispersal_ability,
  thermal_tolerance = thermal_tolerance,
  habitat_specificity = habitat_specificity,
  population_size = population_size
)

# Calculate vulnerability scores (higher = more vulnerable)
vulnerability_data$sensitivity <- 4 - thermal_tolerance
```

```
vulnerability_data$adaptive_capacity <- 4 - (dispersal_ability + population_size) / 2
vulnerability_data$exposure <- habitat_specificity
vulnerability_data$vulnerability <- with(vulnerability_data,
                                         (sensitivity + adaptive_capacity + exposure) / 3)

# Sort by vulnerability
vulnerability_data <- vulnerability_data[order(vulnerability_data$vulnerability, decreasing = TRUE), ]

# Visualize vulnerability scores
ggplot(vulnerability_data, aes(x = reorder(species, -vulnerability), y = vulnerability)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(title = "Climate Change Vulnerability by Species",
       x = "Species",
       y = "Vulnerability Score") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
# Visualize components
vulnerability_components <- vulnerability_data[, c("species", "sensitivity", "adaptive_capacity", "expos
vulnerability_long <- reshape2::melt(vulnerability_components, id.vars = "species")

ggplot(vulnerability_long, aes(x = reorder(species, -value), y = value, fill = variable)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Components of Climate Change Vulnerability",
       x = "Species",
       y = "Score",
       fill = "Component") +
```

```
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



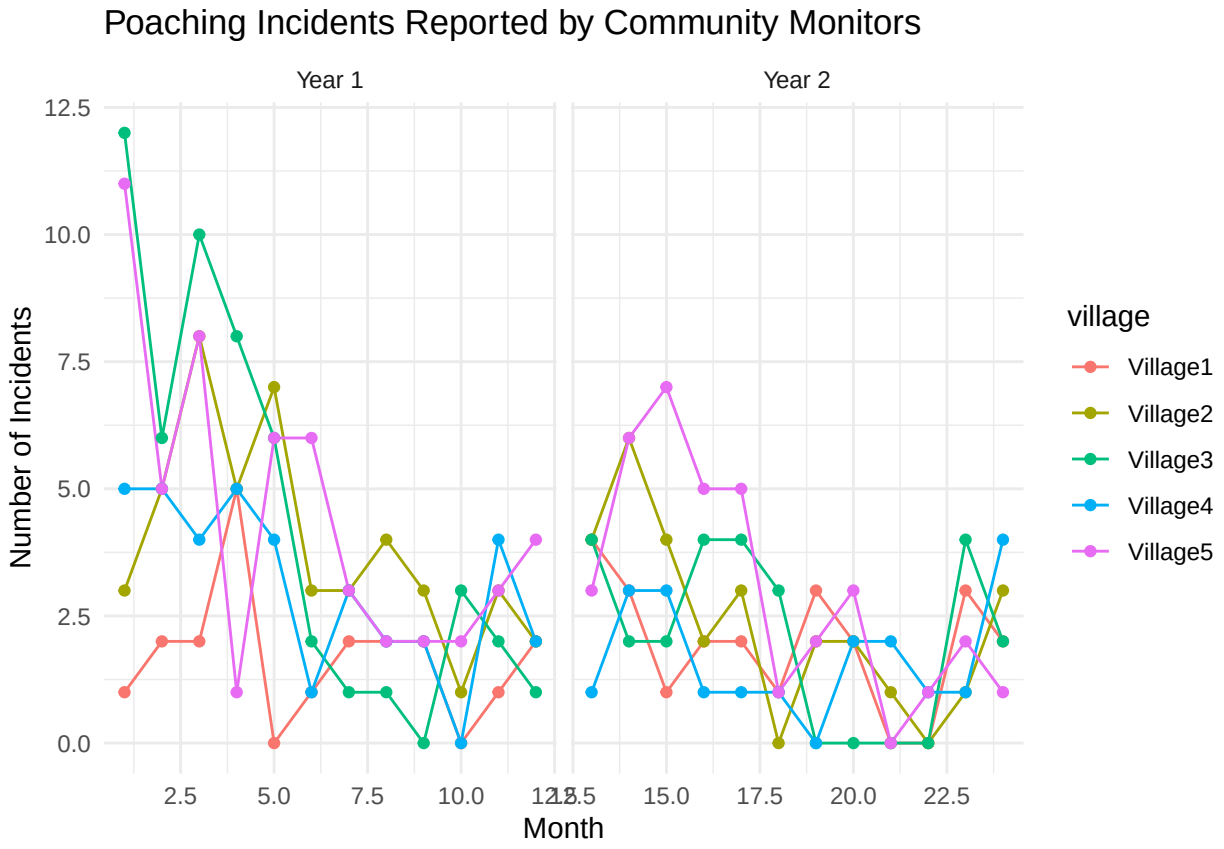
12.10 Community-Based Conservation Monitoring

Involving local communities in conservation monitoring can improve data collection and conservation outcomes.

12.10.1 Example: Analyzing Community Monitoring Data

```
# Simulate community monitoring data
set.seed(202)
n_villages <- 5
n_months <- 24

# Create variables
village <- rep(paste0("Village", 1:n_villages), each = n_months)
month <- rep(1:n_months, times = n_villages)
year <- rep(rep(c(1, 2), each = 12), times = n_villages)

# Generate poaching incidents with seasonal pattern and declining trend
season <- sin(month * pi / 6) + 1  # Seasonal pattern
trend <- -0.03 * (month - 1)  # Declining trend
village_effect <- rep(rnorm(n_villages, 0, 0.5), each = n_months)
lambda <- exp(1 + 0.5 * season + trend + village_effect)
```

```
poaching <- rpois(n_villages * n_months, lambda)

# Create a data frame
monitoring_data <- data.frame(
  village = factor(village),
  month = month,
  year = factor(year),
  poaching = poaching
)

# Visualize the data
ggplot(monitoring_data, aes(x = month, y = poaching, color = village, group = village)) +
  geom_line() +
  geom_point() +
  facet_wrap(~year, scales = "free_x", labeller = labeller(year = c("1" = "Year 1", "2" = "Year 2"))) +
  labs(title = "Poaching Incidents Reported by Community Monitors",
       x = "Month",
       y = "Number of Incidents") +
  theme_minimal()
```




```
# Analyze trends
library(MASS)
trend_model <- glm.nb(poaching ~ month + village, data = monitoring_data)
summary(trend_model)
```

```
Call:
```

```
glm.nb(formula = poaching ~ month + village, data = monitoring_data,
    init.theta = 21.97524464, link = log)

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)      1.229650   0.181336   6.781 1.19e-11 ***
month           -0.057015   0.008742  -6.522 6.94e-11 ***
villageVillage2  0.545243   0.201645   2.704 0.006852 **
villageVillage3  0.558968   0.201193   2.778 0.005465 **
villageVillage4  0.270966   0.211843   1.279 0.200866
villageVillage5  0.716424   0.196360   3.649 0.000264 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Negative Binomial(21.9752) family taken to be 1)

    Null deviance: 199.13  on 119  degrees of freedom
Residual deviance: 136.01  on 114  degrees of freedom
AIC: 468.75

Number of Fisher Scoring iterations: 1

              Theta:  22.0
          Std. Err.:  23.9

 2 x log-likelihood:  -454.752
```

```
# Calculate overall trend
trend_coef <- coef(trend_model)["month"]
monthly_change <- (exp(trend_coef) - 1) * 100
cat("Monthly change in poaching incidents:", round(monthly_change, 2), "%\n")
```

```
Monthly change in poaching incidents: -5.54 %
```

```
# Analyze seasonal patterns
season_model <- glm.nb(poaching ~ sin(2 * pi * month / 12) + cos(2 * pi * month / 12) + village,
                       data = monitoring_data)
summary(season_model)
```

```
Call:
glm.nb(formula = poaching ~ sin(2 * pi * month/12) + cos(2 *
    pi * month/12) + village, data = monitoring_data, init.theta = 40.9900692,
    link = log)

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)            0.48468    0.15815   3.065 0.002180 **
sin(2 * pi * month/12) 0.64312    0.08539   7.531 5.03e-14 ***
cos(2 * pi * month/12) 0.08170    0.08155   1.002 0.316459
villageVillage2        0.55310    0.19722   2.805 0.005039 **
villageVillage3        0.57202    0.19658   2.910 0.003617 **
villageVillage4        0.28057    0.20754   1.352 0.176412
villageVillage5        0.72313    0.19186   3.769 0.000164 ***
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Negative Binomial(40.9901) family taken to be 1)

    Null deviance: 209.78  on 119  degrees of freedom
Residual deviance: 129.01  on 113  degrees of freedom
AIC: 457.46

Number of Fisher Scoring iterations: 1

              Theta:  41.0
          Std. Err.:  70.8

 2 x log-likelihood:  -441.462
```

```
# Compare models
anova(trend_model, season_model)
```

```
Likelihood ratio tests of Negative Binomial Models

Response: poaching
                                                          Model    theta Resid. df
1                                                 month + village 21.97524       114
2 sin(2 * pi * month/12) + cos(2 * pi * month/12) + village 40.99007       113
     2 x log-lik.   Test    df LR stat.      Pr(Chi)
1       -454.7522
2       -441.4616 1 vs 2     1  13.2906 0.0002667407
```

12.11 Summary

In this chapter, we've explored how data analysis techniques can be applied to conservation challenges:

- Species distribution modeling to predict habitat suitability
- Population trend analysis to monitor species status
- Habitat fragmentation analysis to assess landscape connectivity
- Protected area effectiveness evaluation using BACI designs
- Threat assessment and prioritization for conservation planning
- Systematic conservation planning using complementarity analysis
- Climate change vulnerability assessment based on species traits
- Community-based conservation monitoring to track threats

These applications demonstrate how the statistical methods covered throughout this book can help address real-world conservation problems, inform management decisions, and ultimately contribute to biodiversity conservation.

12.12 Exercises

1. Import a dataset on species occurrences and environmental variables, then build a simple species distribution model.
2. Analyze population monitoring data to detect trends and assess conservation status.
3. Calculate basic landscape metrics for a land cover map to quantify habitat fragmentation.
4. Design and analyze a BACI study to evaluate the effectiveness of a conservation intervention.
5. Conduct a threat assessment for a species or ecosystem of your choice.
6. Use complementarity analysis to identify priority sites for conservation.
7. Perform a climate change vulnerability assessment for a group of species.

8. Analyze community monitoring data to detect trends in threats or biodiversity.

References

Bolker, B. et al. (2009). *Generalized linear mixed models: A practical guide.* Trends in Ecology & Evolution.

Elith, J., Leathwick, J. R., & Hastie, T. (2009). Species distribution models: Ecological explanation and prediction across space and time. *Annual Review of Ecology, Evolution, and Systematics*, *40*, 677–697.

Gotelli, N. J., & Ellison, A. M. (2004). *Null model analysis of species co-occurrence patterns.* Sinauer Associates.

Wickham, H., & Grolemund, G. (2016). *R for data science: Import, tidy, transform, visualize, and model data.* O'Reilly Media, Inc.

Zuur, A., Ieno, E. N., & Smith, G. M. (2007). *Analyzing ecological data.* Springer.

Zuur, A., Ieno, E. N., Walker, N., Saveliev, A. A., & Smith, G. M. (2009). *Mixed effects models and extensions in ecology with r.* Springer Science & Business Media.