A Beginner's Guide to Data Analysis Using R and Jamovi

Dr. Jimmy Moses (Ph.D.)

02 October, 2023

Contents

1 Introduction

Ecology, the study of the interactions between organisms and their environment, generates vast amounts of data. Analyzing this data is crucial for understanding ecosystems, making informed conservation decisions, and addressing environmental challenges. Statistical data analysis is the cornerstone of ecological research, enabling scientists to derive meaningful insights from complex ecological datasets.

This guide, "Exploring Ecological Data with R and Jamovi," is tailored for beginners in the field of ecology who are eager to harness the power of statistical analysis to unravel ecological mysteries. Whether you're a budding ecologist, a conservation enthusiast, or a student embarking on ecological research, this guide will serve as your compass through the intricate world of data analysis.

Why R and Jamovi?

R (R Core Team, 2023) is a popular open-source statistical programming language renowned for its versatility and power in data analysis. It has become the lingua franca of data science and is extensively used in ecological research. R provides a vast ecosystem of packages tailored to various ecological analyses, making it an invaluable tool for ecologists.

Jamovi (Şahīn & Aybek, 2020), on the other hand, is a user-friendly statistical software designed with accessibility in mind. Its intuitive graphical interface and point-and-click functionality make it an excellent choice for beginners. Jamovi seamlessly integrates with R, allowing users to transition from a simple point-and-click environment to more complex analyses in R as they gain proficiency.

Who Is This Guide For?

This guide is tailored for:

- Students and researchers entering the field of ecology or related fields.
- Conservationists and environmentalists interested in data-driven decision-making.
- Anyone curious about using R and Jamovi for data analysis, whether in ecology or any other field.

Embark on your ecological data analysis journey with confidence. This guide aims to demystify statistical analysis using R and Jamovi, providing you with the skills and knowledge to explore ecological data, ask critical questions, and contribute to the understanding and conservation of our natural world. Let's begin our exploration of ecological data with R and Jamovi.

1.1 Purpose of this Manual

This comprehensive manual is intended to serve as a detailed and user-friendly guide to the installation and competent use of core data analysis tools: R (R Core Team, 2023), RStudio (RStudio Team, 2020), and Jamovi (Şahīn & Aybek, 2020). These software programmes play critical roles in a variety of sectors, including forestry and ecology, where they are essential for conducting demanding statistical analysis, creating intelligent data visualizations, and driving significant research initiatives.

Notably, this training manual played an important role as an integral component of a workshop held at the Department of Forestry, Papua New Guinea University of Technology. The primary goal of the workshop was to empower its participants, the majority of whom were novices, by providing them with the required tools and competences for competent data analysis within the specialized environment.

This manual, which will be converted to a handbook in the near future, will be regularly expanded to cover complex data analysis techniques such as geospatial mapping and modelling. These planned upgrades will serve as the foundation for a complete manual that will take your data analysis skills to the next level. These changes are intended to address the growing expectations of forestry and ecological researchers, professionals, and students.

Furthermore, this manual will serve as a foundation for future workshops. These workshops aim to dive further into complicated data analysis, sophisticated modelling methodologies, and the use of geospatial data for improved ecological insights. We are devoted to equipping participants with cutting-edge information and

practical skills that will enable them to flourish in the dynamic fields of agriculture, environmental science, forestry and ecology.

Through these future developments and workshops, we aim to foster a community of proficient data analysts and researchers who can make significant contributions to the sustainable management of our natural ecosystems.

Key Outcomes:

- **Tool Proficiency**: Participants are equipped with the proficiency to harness the capabilities of R, RStudio, and Jamovi as powerful aids in their data analysis workflows.
- **Statistical Analysis**: Participants will obtain the knowledge and practical skills required for in-depth statistical analysis, allowing them to test hypotheses, investigate relationships, and develop data-driven conclusions.
- **Data Visualization**: The handbook instructs users on how to use these tools to generate useful and aesthetically appealing data visualizations, which are an important feature of data communication.
- **Research Empowerment**: With these essential tools and the know-how to properly use them, participants are better positioned to contribute meaningfully to research efforts, particularly in forestry and ecology.

Readers will not only obtain the skills to set up and utilize the aforementioned programs by immersing themselves in the contents of this handbook, but they will also receive vital insights into their practical applications. This newly acquired skill will not only improve their capacity to perform reliable data analysis but will also enable them to make significant contributions to the domains of forestry and ecology or any other related fields.

2 Chapter 1: Installation and Setup

2.1 Prerequisites

Before you begin the installation process, it is essential to ensure that you meet the following prerequisites:

- **Internet Connection**: You must have a stable and active internet connection to download the required software packages and updates.
- **Administrator Privileges (Windows)**: If you are using a Windows operating system, you may need administrator privileges to install software. Ensure that you have the necessary permissions.
- **Operating System Compatibility**: Verify that your operating system is compatible with the software you intend to install. Each software package has specific system requirements, which will be outlined in the installation sections.
- **Basic Computer Skills**: This manual assumes that you have basic computer skills, including the ability to navigate your operating system and use a web browser.
- **Storage Space**: Ensure that you have sufficient disk space available to accommodate the software installations. The installation sections will specify the approximate storage requirements.
- **Hardware Requirements**: Check if your computer meets the hardware requirements specified by the software developers. This information is usually available on the respective software websites.

Computers operate on three of the most common operating systems: Windows, Mac OS, and Linux. Depending on your operating system, follow the installation instructions relevant to your specific platform. The process will differ slightly for Windows, macOS, and Linux users, so ensure you select the appropriate set of instructions based on your system.

2.2 Installing R

2.2.1 Windows

2.2.1.1 Downloading R

1. Open your web browser and navigate to the R download page (download here).
2. Click on the "Download R for Windows" link.
3. Choose a CRAN mirror (usually the one geographically closest to you) and click on its link.
4. Download the base version of R for Windows by clicking on the "install R for the first time" link.

2.2.1.2 Installing R

1. Locate the downloaded R installer (an .exe file) and double-click it.
2. Follow the installation wizard's instructions:
   - Choose the language.
   - Accept the terms of the license.
   - Select the components you want to install (usually, you can leave the default settings).
   - Choose the installation location (you can leave the default).
   - Click "Next" to start the installation.

Once the installation is complete, you can now run R by searching for "R" in the Windows Start menu.

2.2.1.3 Downloading and Installing RTools in Windows RTools is a collection of tools required for building and installing R packages from source on Windows. It is essential for users who want to compile and install R packages from CRAN or other sources. Here are the steps to download and install RTools on a Windows system:

1. **Download RTools:**
   - Open your web browser and navigate to the RTools download page (download here).
   - On the RTools download page, scroll down to find the "Download Rtools" section.
   - Click on the link that corresponds to the version of RTools recommended for your version of R. It is essential to match the RTools version with the R version you have installed. For example, if you have R version 4.1.x, download the RTools version recommended for R 4.1.x.
   - You will be directed to a new page with a list of download links. Click on the link that says "install.exe" to download the RTools installer.
2. **Installing RTools:**
   - Locate the downloaded RTools installer (an .exe file) and double-click it to start the installation.
   - Follow the installation wizard's instructions:
     - Choose the language.
     - Accept the terms of the license.
     - Select the components you want to install. It is recommended to install all components, so leave the default settings selected.
     - Choose the installation location. By default, RTools will install in the "C:\Rtools" directory. You can change this location if necessary.
     - Click "Next" to start the installation.

- During the installation, you may see a message about modifying the system PATH. Make sure to select the option that adds RTools to the system PATH. This is essential for R to find and use RTools when building packages.
- Once the installation is complete, you can click "Finish" to exit the installer.

3. **Verifying the Installation:**
   - To verify that RTools has been installed correctly, open R or RStudio.
4. In the R console, you can run the following command to check if RTools is found:

```
Sys.which("make")
```

1. If the installation was successful, you should see a path to the **`make.exe`** executable associated with RTools.

2.3 Installing RStudio

2.3.1 Windows

2.3.1.1 Downloading RStudio

1. Open your web browser and navigate to the RStudio download page (download here).
2. Scroll down to the "RStudio Desktop" section.
3. Click on the "Download" button under the "RStudio Desktop (Free)" option.
4. Download the RStudio installer for Windows.

2.3.1.2 Installing RStudio

1. Locate the downloaded RStudio installer (an .exe file) and double-click it.
2. Follow the installation wizard's instructions:
   - Choose the language.
   - Accept the license agreement.
   - Choose the installation location (you can leave the default).
   - Select additional tasks (optional).
   - Click "Install" to begin the installation.
3. Once the installation is complete, you can run RStudio by searching for "RStudio" in the Windows Start menu.

2.4 Installing Jamovi

2.4.1 Windows

2.4.1.1 Downloading Jamovi

1. Open your web browser and navigate to the Jamovi download page (download here).
2. Click on the "Download for Windows" button.
3. Download the Jamovi installer for Windows.

2.4.1.2 Installing Jamovi

1. Locate the downloaded Jamovi installer (an .exe file) and double-click it.
2. Follow the installation wizard's instructions:
    - Choose the installation location (you can leave the default).
    - Select additional tasks (optional).
    - Click "Install" to begin the installation.
3. Once the installation is complete, you can run Jamovi by searching for "Jamovi" in the Windows Start menu.

That's it! You have successfully downloaded and installed R, RTools, RStudio and Jamovi on your Windows system. You are now ready to perform data analysis!

2.5 Verifying Installations

2.5.1 Checking R Installation

- **Windows**
    1. Open R by searching for "R" in the Start menu.
    2. The R console should appear. You can start using R by entering commands.

2.5.2 Checking RStudio Installation

- **Windows**
    1. Open RStudio by searching for "RStudio" in the Start menu.
    2. The RStudio IDE should open, ready for use.

2.5.3 Checking Jamovi Installation

- **Windows**
    1. Open Jamovi by searching for "Jamovi" in the Start menu.
    2. The Jamovi interface should open.

2.6 Updating and Maintaining Software

2.6.1 Updating R

- Regularly update R to ensure you have the latest features and bug fixes.

2.6.2 Updating RStudio

- For RStudio updates, download the latest version from the RStudio website and reinstall it. Your projects and settings will be preserved.

2.6.3 Updating Jamovi

- Jamovi typically updates itself automatically when you launch the software. Ensure you have an internet connection for updates to occur.

2.7 Troubleshooting

2.7.1 Common Installation Issues

2.7.1.1 Difficulty in Downloading Software

- If you encounter difficulties downloading R, RStudio, or Jamovi, ensure you have a stable internet connection.
- Try using an alternative download mirror if the default one is slow or unresponsive.
- Disable any firewall or security software temporarily, as they may block downloads.

2.7.1.2 Compatibility Issues

- Ensure that you are downloading the correct version of the software that matches your operating system (Windows, macOS, Linux) and architecture (32-bit or 64-bit).
- Verify that your operating system meets the minimum requirements for the software.

2.7.2 Package Installation Issues

2.7.2.1 R Package Installation

- If you encounter issues when installing R packages using the `install.packages()` function, ensure you have an internet connection.
- Some packages may require additional system libraries. Check the package documentation for any specific requirements.
- If you receive errors related to permissions, consider running R or RStudio with administrative privileges (e.g., "Run as administrator" on Windows).

2.7.2.2 Jamovi Module Installation

- When installing Jamovi modules (extensions), ensure that you are using a compatible version of Jamovi.
- If a module installation fails, check if there are any error messages provided. These messages often indicate the cause of the issue.

2.7.2.3 Resources for Assistance

- If you encounter installation or package-related issues that are not covered here, consider seeking help from the following resources:
  - **Online Forums:** Visit community forums or discussion boards related to R, RStudio, and Jamovi. Experienced users often provide solutions to common problems.
  - **Official Documentation:** Consult the official documentation for each software. They often include troubleshooting sections.
  - **User Communities:** Join user communities or mailing lists where you can ask questions and seek assistance from experienced users.

2.8 Testing R Package Installation

2.8.1 Introduction

To ensure that your R environment is fully functional and equipped with essential packages for data analysis, we will test the installation of key R packages. These packages include devtools, remotes, tidyverse, and rstatix. This process will help verify that the packages can be installed and loaded successfully within RStudio.

2.8.2 Installing and Loading Packages

Follow these steps to install and load the required R packages using RStudio:

1. **Open RStudio**
   - Launch RStudio by searching for "RStudio" in your computer's application menu.
2. **Installing devtools and remotes**
   - In the RStudio console, enter the following commands to install the devtools (Wickham et al., 2022) and remotes (Csárdi et al., 2023) packages:

     ```
     install.packages("devtools")
     install.packages("remotes")
     ```

   - Wait for the installations to complete. You may be prompted to select a CRAN mirror; choose a location geographically close to you for faster downloads.
3. **Loading devtools and remotes**
   - After installation, load the devtools and remotes packages by entering these commands in the console:

     ```
     library(devtools)
     library(remotes)
     ```

4. **Installing tidyverse and rstatix**
   - With devtools and remotes loaded, you can now install the tidyverse (Wickham et al., 2019) and rstatix (Kassambara, 2023) packages, which are essential for data manipulation and statistical analysis:

     ```
     install.packages("tidyverse")
     remotes::install_github("kassambara/rstatix")
     ```

   - Allow the installations to proceed. The remotes package is used to install rstatix directly from its GitHub repository.
5. **Loading tidyverse and rstatix**
   - Once installed, load the tidyverse and rstatix packages with the following commands:

     ```
     library(tidyverse)
     library(rstatix)
     ```

6. **Verifying Package Loading**
   - To confirm that the packages have been successfully loaded, you can execute a simple test. For instance, you can try running the following command, which uses a function from the tidyverse package:

     ```
     ggplot2::qplot(1:10, rnorm(10))
     ```

If the packages have been loaded correctly, you should see a basic scatterplot generated by ggplot2.

Testing the installation and loading of these packages ensures that your R environment is ready for data analysis tasks. By successfully installing and loading devtools, remotes, tidyverse, and rstatix, you have access to a powerful set of tools for data manipulation, visualization, and statistical analysis within RStudio.

You are now well-equipped to embark on data analysis projects with R, and you can confidently explore additional packages tailored to your specific needs.

2.9 Summary of Key Steps

In this manual, we have provided comprehensive guidance on the installation of essential data analysis tools: R, RStudio, and Jamovi. These tools are invaluable for conducting statistical analysis, data visualization, and research in various fields.

To recap the key steps covered in this manual:

1. **Installing R**
   - Choose the appropriate version for your operating system (Windows, macOS, or Linux).
   - Follow the step-by-step instructions provided to download and install R.
   - Verify the successful installation of R by launching the R console.
2. **Installing RStudio**
   - Select the correct version of RStudio for your operating system.
   - Follow the installation instructions to download and install RStudio.
   - Confirm the successful installation of RStudio and its integration with

     R.
3. **Installing Jamovi**
   - Download Jamovi for your operating system.
   - Execute the installation process as guided in the manual.
   - Validate the installation by launching Jamovi.
4. **Verifying Installations**
   - Ensure that R, RStudio, and Jamovi open without errors.
   - Confirm that you can access the R console and RStudio IDE smoothly.
5. **Updating and Maintaining Software**
   - Regularly check for updates to R, RStudio, and Jamovi to benefit from the latest features and bug fixes.
   - Follow the guidelines provided to update each software component.
6. **Troubleshooting**
   - Consult the troubleshooting section if you encounter common installation or package-related issues.
   - Utilize online forums, official documentation, and user communities to seek assistance for more complex problems.

2.9.1 Next Steps

Now that you have successfully installed R, RStudio, and Jamovi, you are equipped with powerful tools for data analysis, statistical modeling, and research. Your next steps might include:

- **Learning and Practicing:** Explore online tutorials and resources to enhance your skills in using R, RStudio, and Jamovi for data analysis.
- **Working on Projects:** Apply your newly acquired knowledge to real-world projects, research, or coursework.
- **Exploring Packages:** Explore and install additional R packages that cater to your specific analytical needs.

- **Collaborating:** Share your work with colleagues or collaborate on data analysis projects using these tools.
- **Staying Informed:** Stay updated with the latest developments and updates for R, RStudio, and Jamovi by subscribing to relevant newsletters or communities.
- **Supporting Others:** Share your knowledge and help others who are beginning their journey with these tools.

2.9.2 Final Remarks

I hope that this installation manual has been a valuable resource in getting you started with R, RStudio, and Jamovi. These tools offer limitless possibilities for data analysis and research. Remember that practice, exploration, and continuous learning will enhance your proficiency in using these tools effectively. Thank you for choosing this manual as your guide to installing and working with these essential data analysis tools. We wish you success in your data analysis endeavors!

2.10 Introducing R and Jamovi

We will continue our exploration of data analysis tools by introducing you to two powerful platforms: R and Jamovi. As a brief recap of our previous session, you've learned about the significance of inferential statistics and its applications in forestry and ecological research (see supplementary section for more detail information). Furthermore, this section provides an introduction to R and Jamovi, covering packages and modules, installation of R and RStudio, basic R commands, data structures, and importing data. It also introduces Jamovi's interface, data import, and basic data manipulation. Now, we will dive into the practical aspects of using R and Jamovi for data analysis.

2.10.1 R Packages and Modules in Jamovi

R Packages and Jamovi Modules are both essential components of statistical analysis and data manipulation, but they differ in several key ways:

2.10.1.1 R Packages

1. **Language Foundation**: R packages are part of the R programming language. R is a versatile, open-source scripting language and environment explicitly designed for statistical computing and data analysis.
2. **Community-Driven**: R packages are developed by a diverse community of statisticians, data scientists, and programmers from around the world. Anyone can contribute to or create R packages, leading to a vast ecosystem with thousands of packages.
3. **Functionality**: R packages provide a wide range of functions and tools for statistical analysis, data visualization, machine learning, and more. These packages can be highly specialized, focusing on specific tasks or domains.
4. **Flexibility**: R packages offer a high level of customization and flexibility. Users can write their R code, combining functions from various packages to create tailored solutions.
5. **Syntax**: R uses its syntax, which is based on function calls and assignments. Users need to learn R's specific syntax to work with R packages effectively.
6. **Integration**: R packages can be integrated with other programming languages like Python and C++, enabling users to harness the capabilities of these languages within R.
7. **Code-Based**: Using R packages often requires writing code or scripts to perform data analysis and visualization tasks. It's suitable for those comfortable with programming.

2.10.1.2 Jamovi Modules

1. **Graphical User Interface (GUI)**: Jamovi is a statistical software that provides a point-and-click graphical user interface (GUI). Jamovi modules are components within this GUI that allow users to perform specific analyses without writing code.
2. **Built-In Functionality**: Jamovi modules come pre-installed with the software and cover a wide range of statistical analyses. Users can access these modules through a user-friendly interface, eliminating the need for coding.
3. **Ease of Use**: Jamovi is designed for users who may not have programming experience. It simplifies statistical analysis by providing intuitive menus, buttons, and options.
4. **Accessibility**: Jamovi is an excellent choice for beginners and users who prefer not to write code. It offers a low learning curve and helps users quickly perform common statistical tasks.
5. **Interactivity**: Jamovi allows users to interact with their data visually. Users can load datasets, click through options, and see immediate results in the interface.
6. **Modular Design**: Jamovi follows a modular design, meaning users can combine different modules to create analysis pipelines. This modular approach promotes reusability and versatility.
7. **Scripting and R Integration**: While Jamovi emphasizes point-and-click functionality, it also includes an R Syntax mode. This mode enables users to write and execute R code within Jamovi, combining the strengths of both approaches.

R Packages are code libraries for the R programming language, offering extensive functionality and flexibility but requiring coding skills. Jamovi Modules, on the other hand, are part of a user-friendly statistical software with a GUI, designed for ease of use and accessibility, making statistical analysis more accessible to a broader audience, including those without coding experience.

2.10.2 Getting Started with R

2.10.2.1 Installing R and RStudio

2.10.2.1.1 Installing R You can install R by following these steps:

1. Visit the CRAN (Comprehensive R Archive Network) website for your operating system (Windows, macOS, or Linux): **https://cran.r-project.org/mirrors.html**
2. Download the R installer for your OS and follow the installation instructions.

2.10.2.1.2 Installing RStudio Once R is installed, you can proceed to install RStudio:

1. Visit the RStudio download page: **https://www.rstudio.com/products/rstudio/download/**
2. Download the appropriate RStudio installer for your OS (RStudio Desktop is recommended for most users).
3. Install RStudio by following the installation instructions.

2.10.2.2 Understanding R Interface and Basics

2.10.2.2.1 Launch RStudio Open RStudio by searching for "RStudio" in your computer's application menu.

2.10.2.2.2 R Interface

- The RStudio interface consists of several panels, including the script editor, console, environment, and plots.

- The script editor is where you write and execute R code.
- The console displays R's output and can be used for direct command entry.

2.10.2.3 Basic Commands and Data Structures: Let's explore some basic commands and data structures:

```
# Basic Arithmetic
2 + 3 # Addition: Calculates and returns 2 plus 3, which is 5.
5 - 2 # Subtraction: Calculates and returns 5 minus 2, which is 3.
4 * 6 # Multiplication: Calculates and returns 4 times 6, which is 24.
8 / 4 # Division: Calculates and returns 8 divided by 4, which is 2.

# Assigning Values to Variables
x <-
  10 # Assign 10 to the variable x: Creates a variable 'x' and
     # assigns the value 10 to it.
y <-
  5 # Assign 5 to the variable y: Creates a variable 'y' and
    # assigns the value 5 to it.

# Vectors
my_vector <-
  c(3, 6, 9, 12) # Create a numeric vector: Constructs a vector
                 # 'my_vector' with the values 3, 6, 9, and 12.
length(my_vector) # Check the length of the vector: Returns the number of
                  # elements in 'my_vector' (4).
mean(my_vector) # Calculate the mean of the vector: Computes the average
                # of the values in 'my_vector' (7.5).

# Data Frames (a common data structure in R)
# Create a sample forestry dataset: Generates a data frame 'forest_data'
# with columns 'TreeSpecies', 'Height', and 'Diameter'.
forest_data <- data.frame(
  TreeSpecies = c("Oak", "Pine", "Maple", "Birch"),
  Height = c(25, 20, 18, 22),
  Diameter = c(10, 8, 7, 9)
)

print(forest_data)  # Print the data frame 'forest_data': Displays the
# content of the data frame.
```

R Code Explanation

- Codes above illustrate basic arithmetic operations, variable assignment, the creation and manipulation of vectors, and the construction of a data frame in R. It demonstrates fundamental operations commonly used in data analysis and manipulation in R.

2.10.2.4 Importing Data into R R allows you to import data from various file formats, such as CSV, Excel, or databases. Here's an example of importing a CSV file:

```
# Check if the "pacman" package is available; if not, install it and load it
if (!require("pacman")) {
  install.packages("pacman")
  require(pacman)
}
```

```r
# Use the "pacman" package to load multiple packages at once
pacman::p_load(
  rstatix,
  # rstatix: Provides functions for statistical analysis
  tidyverse,
  # tidyverse: A collection of packages for data manipulation and visualization
  easystats,
  # easystats: Provides easy-to-use functions for statistical analysis
  readr,
  # readr: Used for reading data from various file formats
  magrittr,
  # magrittr: Provides a pipe operator (%>%) for easier data manipulation
  knitr,
  # knitr: Used for dynamic report generation
  report,
  # report: A package for creating and formatting reports
  scatr,
  # scatr: Tools for exploratory data analysis and visualization
  jmv,
  # jmv: Tools for statistical analysis and hypothesis testing
  haven,
  # haven: A package for reading and writing data in SAS format
  foreign,
  # foreign: A package for reading and writing data in various formats
  performance,
  # performance: Provides functions for assessing model performance
  ggthemes,
  here,
  install = TRUE,
  # Install packages if not already installed
  update = TRUE   # Update packages if newer versions are available

)

# Load the iris dataset
data("iris")          # Load the built-in iris dataset into the R environment

# Uncomment the following line to view the dataset in a separate window
# View(iris)

# Display a concise summary of the iris dataset
tibble::glimpse(iris)

# Create a scatter plot using ggplot2 to visualize the relationship
# between Sepal.Length and Sepal.Width
ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width)) +
  geom_point() +                    # Add points to the plot
  geom_smooth(method = "lm")        # Add a linear regression line

# Read the iris dataset from a CSV file into a variable called iris_import
iris_import <-
  read_csv(file = here::here("docs", "data", "Filtered.csv"))
```

```r
# Uncomment the following line to view the imported dataset
# View(iris_import)

# Export the original iris dataset to a CSV file at the specified file path
utils::write.csv(
  x = iris,
  # Dataset to export (iris)
  file = "./data/iris.csv",
  # File path to save the CSV
  fileEncoding = "UTF-8",
  # Encoding of the CSV file
  row.names = FALSE                    # Exclude row names in the CSV
)

# Use getwd() to show your current working directory path
```

R Code Explanation

- The code first checks if the "pacman" package [] is available using the **`require`** function. If it's not available, it installs the "pacman" package and then loads it using **`require`**.
- The **`pacman::p_load`** function is used to load multiple packages at once. Each package listed is loaded, and if any of them are not installed, they will be installed (**`install = TRUE`**). Additionally, if newer versions of packages are available, they will be updated (**`update = TRUE`**).
- Here's a brief explanation of each package:
  - **`rstatix`**(**Kassambara, 2023**): Provides functions for statistical analysis, particularly for tidyverse users.
  - **`tidyverse`**(**Wickham et al., 2019**): A collection of packages (e.g., dplyr, ggplot2) for data manipulation and visualization.
  - **`easystats`**(**Lüdecke et al., 2022**): Provides easy-to-use functions for various statistical analyses.
  - **`readr`** (Wickham et al., 2023): Used for reading data from various file formats (e.g., CSV, Excel).
  - **`magrittr`** (Bache & Wickham, 2022)**: Provides the pipe operator (`%>%`**) for easier data manipulation.
  - **`knitr`**(**Xie, 2023**): Used for dynamic report generation, often in combination with R Markdown.
  - **`report`**(**Makowski et al., 2023**): A package for creating and formatting reports.
  - **`scatr`**(**Selker, 2017**): Provides tools for exploratory data analysis and visualization.
  - **`jmv`**(**Selker et al., 2022**): Contains tools for statistical analysis and hypothesis testing.
  - **`performance`** (Lüdecke et al., 2021): Provides functions for assessing model performance.

This code is an efficient way to ensure that all the necessary packages are available and up-to-date for your data analysis and reporting needs.

The next set of R codes above performs the following actions:

1. Loads the built-in iris dataset and optionally allows you to view its summary using **`tibble::glimpse(iris)`**.
2. You can uncomment the line **`# View(iris)`** to view the dataset in a separate window.
3. Creates a scatter plot using **`ggplot2`** to visualize the relationship between Sepal.Length and Sepal.Width.

4. Imports the iris dataset from a CSV file located at "data/iris.csv" into a variable called **`iris_import`** using the **`read_csv`** function from the **`readr`** package.
5. Exports the original iris dataset to a CSV file at the specified file path.
6. The last comment mentions that you can use **`getwd()`** to show your current working directory path.

Please note that some lines of code are commented out and can be uncommented when needed.

2.10.3 Introduction to Jamovi

2.10.3.1 Overview of Jamovi Interface and Functionalities Jamovi is a user-friendly statistical software with an intuitive interface. You can launch it by searching for "Jamovi" in your application menu.

2.10.3.1.1 Step 1: Launching Jamovi

1. Open your web browser.
2. In the address bar, type "**https://www.jamovi.org/**" and press Enter.
3. On the Jamovi website, click on the "Download" or "Try Online" button to access Jamovi.
4. If you choose to "Try Online," it will open the Jamovi online interface in your browser.

2.10.3.1.2 Step 2: Interface Overview

1. Familiarize yourself with the Jamovi interface.
   - The top menu contains options for File, Edit, View, Analysis, Data, and more.
   - The left panel displays the data set (if loaded) and variables.
   - The right panel shows the analysis output.

2.10.3.2 Importing Data and Basic Data Manipulation using Jamovi

2.10.3.2.1 Step 3: Importing Data

1. In the left panel, under the "Data" tab, click "Import Dataset."
2. Choose the data file you want to import (e.g., CSV, Excel).
3. Follow the on-screen instructions to configure data import settings and load your dataset.

2.10.3.2.2 Step 4: Basic Data Manipulation

1. After loading the data, explore the "Data" tab on the left panel.
   - You can view and manipulate variables, including renaming, re-coding, or creating new variables.
2. To perform basic data manipulation tasks:
   - Select variables by clicking on their names.
   - Use the right-click menu or the "Transform" option to apply changes.

2.10.3.2.3 Step 5: Saving Your Work

1. To save your project:
   - Click on "File" in the top menu.
   - Select "Save Project As" and choose a location to save your .omv project file.

2.10.3.3 Introduction to Statistical Analyses in Jamovi

2.10.3.3.1 Step 6: Running Basic Analyses

1. Under the "Analysis" tab in the top menu, you'll find a wide range of statistical analyses.
2. Select an analysis based on your research question (e.g., t-tests, ANOVA, regression).
3. Configure the analysis by specifying variables, settings, and options.
4. Click the "Run" button to execute the analysis.

2.10.3.3.2 Step 7: Interpreting Results

1. After running an analysis, the results will appear in the right panel.
2. Interpret the output, including statistical measures, p-values, and visualizations.
3. Jamovi often provides descriptive statistics, charts, and inferential test results.

2.10.3.4 Statistical Analyses in Jamovi Jamovi provides a wide range of statistical analyses, including t-tests, ANOVA, regression, and more. In the following sessions, we will explore these analyses in greater detail.

2.11 Software Integration

Jamovi, a user-friendly and open-source statistical analysis software, offers an interface that seamlessly integrates with **R**, a powerful programming language and environment for statistical computing. This integration combines the user-friendly features of Jamovi with the robust statistical capabilities of R, providing a flexible platform for data analysis and visualization.

Key points to consider in integrating R in Jamovi:

1. **Enhanced Statistical Power**: Jamovi's integration with R allows users to access a wide range of advanced statistical techniques and methods available in the R ecosystem. This includes specialized packages for complex data analysis and modeling.
2. **Interactive Analysis**: Users can perform analyses in Jamovi using its point-and-click interface while observing the R syntax generated in real-time. This helps users learn and understand the R code associated with their analyses.
3. **Customization and Automation**: For users familiar with R, the integration enables seamless customization and automation of analyses. R users can extend analyses conducted in Jamovi with additional scripting and package integration.
4. **Data Visualization**: R's data visualization capabilities are available within Jamovi, allowing users to create custom plots and charts for data exploration and presentation.
5. **Statistical Reporting**: Users can generate reports in Jamovi, including statistical summaries, tables, and visualizations, which can be customized and exported for publication or sharing.
6. **Collaboration**: Teams with varying levels of statistical expertise can collaborate effectively. Users can share Jamovi projects with colleagues, even if they are not R proficient, ensuring consistent analyses and results.
7. **Learning Opportunity**: For those new to R, Jamovi serves as a valuable learning tool. Users can explore the R code generated by Jamovi, helping them transition to more extensive R-based analyses.

In essence, integrating R in Jamovi offers a powerful combination of user-friendly statistical analysis and the versatility of R scripting. It caters to both beginners and experienced statisticians, making it an ideal choice for data analysis, research, and collaboration across diverse domains.

2.11.1 R as a Backend for Jamovi

1. **R as a Backend:** Jamovi, a user-friendly statistical analysis software, uses R as its computational backend. This means that when you perform statistical analyses or create plots in Jamovi, the software generates and executes corresponding R code in the background. This allows users who may not be familiar with R to take advantage of its powerful statistical capabilities.
2. **Point-and-Click Interface:** Jamovi provides a point-and-click interface for performing statistical analyses. Users can easily import datasets, perform various statistical tests, create visualizations, and generate reports without writing any R code.
3. **Real-Time Syntax Generation:** One of the key features of Jamovi is its ability to generate R syntax in real-time. As you perform actions in the Jamovi interface, such as running a t-test or creating a scatterplot, the corresponding R code is displayed. This provides users with a learning opportunity to understand the R commands associated with the analysis.
4. **Easy Transition to R:** For users who want to transition to R or have specific customization needs, Jamovi simplifies the process. Users can copy the generated R syntax from Jamovi and use it directly in their R environment for further customization or scripting.

2.11.2 Integration of Jamovi in R

1. **R Packages for Jamovi:** R users can take advantage of the "jmv" package, which allows them to interact with Jamovi directly from their R environment. This package provides functions to load Jamovi analyses, extract results, and incorporate Jamovi analyses into R scripts.
2. **Seamless Collaboration:** Researchers or data analysts who prefer R can still collaborate effectively with colleagues or team members using Jamovi. They can create their analyses in Jamovi, export the results as datasets, and then use R for advanced statistical modeling or further data manipulation.
3. **Customization:** R users can customize and extend the functionality of Jamovi analyses using R packages and scripts. This allows for greater flexibility and advanced data analysis when needed.
4. **Leveraging the Best of Both Worlds:** The integration of Jamovi in R and R in Jamovi offers the best of both worlds. Users can enjoy the simplicity and user-friendly interface of Jamovi for routine analyses while harnessing the extensive statistical and data manipulation capabilities of R when required.

The integration of R in Jamovi and Jamovi in R provides users with a flexible and powerful ecosystem for conducting statistical analyses. It accommodates users of varying skill levels, from those who prefer a graphical interface to those who are experienced R users, facilitating effective collaboration and streamlined workflows in data analysis and research.

2.11.3 Setting up R Syntax Mode in Jamovi

Setting up R Syntax mode in Jamovi allows users to harness the full power of R programming within Jamovi's user-friendly interface. This mode enables users to write, execute, and customize R code seamlessly while taking advantage of Jamovi's data visualization and analysis features. Here's how to set up R Syntax mode in Jamovi:

1. **Install Jamovi**: If you haven't already, download and install Jamovi on your computer from the official Jamovi website (**https://www.jamovi.org/download.html**).
2. **Launch Jamovi**: Open Jamovi and create or open a dataset for analysis. You'll typically start in the default point-and-click mode.
3. **Enable R Syntax Mode**
    - Navigate to the far end right corner of Jamovi window and click on the three vertical dots.

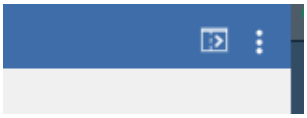


Figure 1: Accessing Jamovi setting via 3 vertical dots.

- From the drop-down menu, select "R Syntax Mode." This action switches your analysis interface to R Syntax mode.

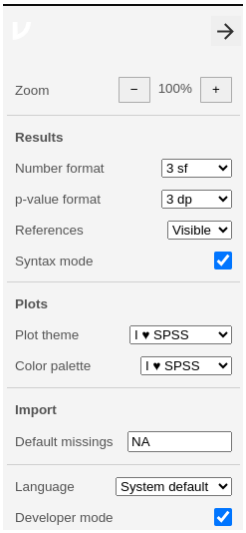


Figure 2: Syntax mode checkbox.

4. **R Syntax Mode**: Once you enable R Syntax Mode, you'll notice that for any analysis or plotting, R codes will appear next to the results.
5. **Writing R Code in Jamovi**
   - You can write and execute R codes directly from within Jamovi's R editor. This is a module which is called "Rj" and can be accessed from the modules library. It must be installed and pinned to the main menu.
   - Click on the "Rj" icon and select "Rj editor".
   - You can start writing R code directly in the console. For example, you can create variables, perform data manipulations, run statistical analyses, and create visualizations using R syntax. Examples are shown below.
6. **Running R Code**: To execute the R code you've written, simply press the "Run" button in the R Syntax Console or use the shortcut (typically Ctrl+Shift+Enter or Cmd+Shift+Enter).
7. **Output and Visualizations**: As you execute R code, any output, plots, or results will be displayed within the console. You can also create custom visualizations using R packages like ggplot2.
8. **Combining Point-and-Click and R**: Jamovi's unique feature allows you to switch between Point-and-Click mode and R Syntax mode seamlessly. You can start with a point-and-click analysis and switch to R Syntax mode to access advanced options and customization.
9. **Saving and Sharing**: You can save your Jamovi project, which includes your dataset, analyses, and R code. This makes it easy to share your work or collaborate with others.

Linear Regression

```
jmv::linReg(
    data = data,
    dep = Aboveground_Tree_Carbon_ton_per_ha,
    covs = vars(Tree_Density_per_ha, Basal_area),
    factors = Management_regime,
    blocks = list(
        list(
            "Tree_Density_per_ha",
            "Basal_area",
            "Management_regime")),
    refLevels = list(
        list(
            var="Management_regime",
            ref="Private")))
```

Model Fit Measures

Model	R	R^2
1	0.943	0.889

Model Coefficients - Aboveground_Tree_Carbon_ton_per_ha

Predictor	Estimate	SE	t	p
Intercept [a]	−1.19592	0.0984	−12.16	< .001
Tree_Density_per_ha	0.00106	8.50e−5	12.51	< .001
Basal_area	80.45590	4.3137	18.65	< .001
Management_regime:				
Department – Private	0.12192	0.0483	2.52	0.014

[a] Represents reference level

Figure 3: R syntax associated with Jamovi analytical outputs.

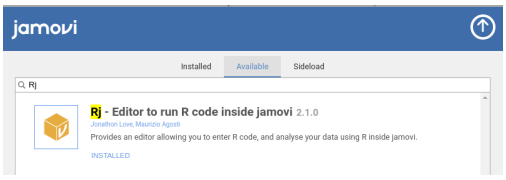


Figure 4: Example of Jamovi modules: here Rj module.

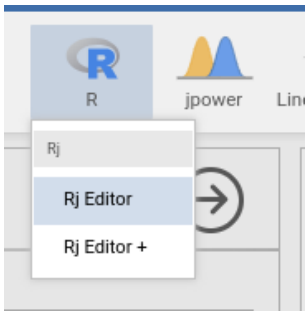


Figure 5: Accessing the Rj Editor.

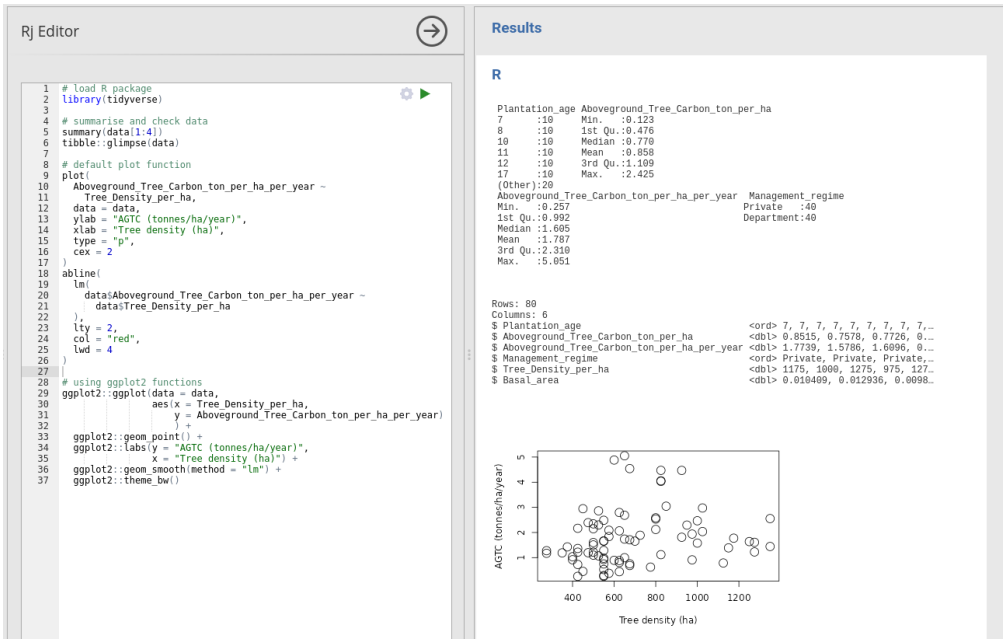


Figure 6: Examples of running R codes in Rj Editor.

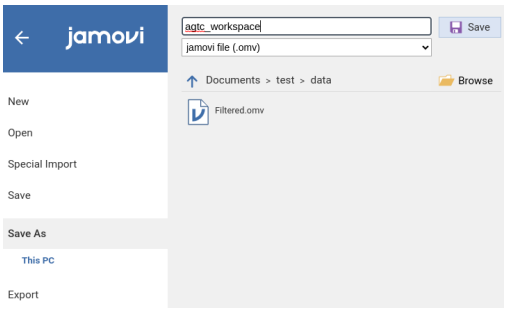


Figure 7: Saving project or workspace (.omv) to working directory.

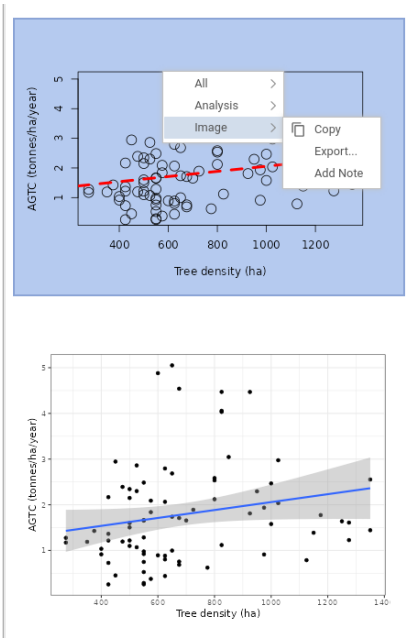


Figure 8: Copy or export jamovi results. Example shown here for plots.

10. **Learning Resources**: If you're new to R, there are plenty of online resources and tutorials available to help you learn R programming within Jamovi. Additionally, the generated R code can serve as a valuable learning tool.

Setting up R Syntax mode and Rj editor module in Jamovi empowers users to perform complex analyses and customizations, making it a versatile tool for both beginners and experienced data analysts and researchers. It combines the accessibility of Jamovi with the flexibility of R, offering the best of both worlds for data analysis and visualization.

2.11.4 Example

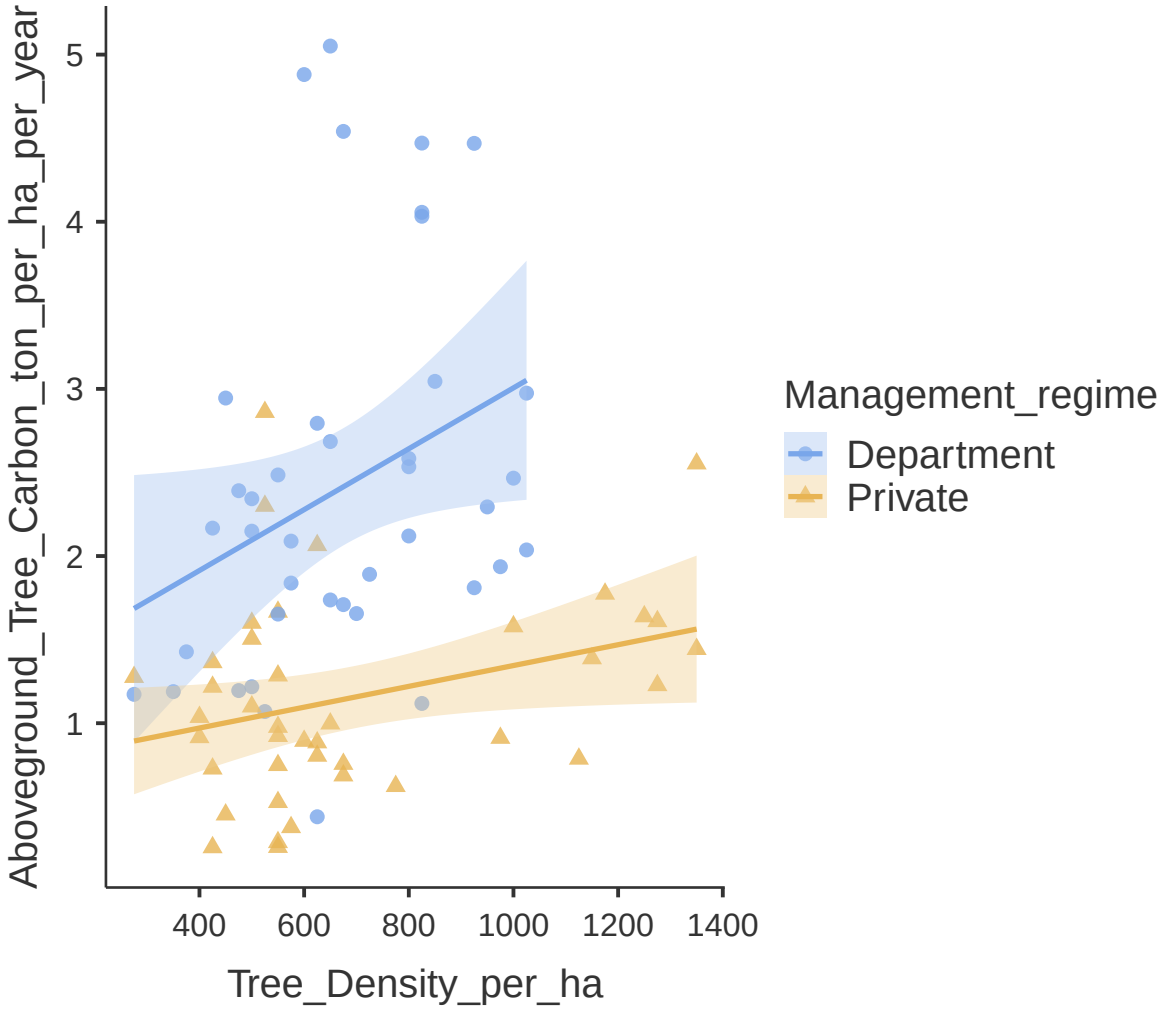
2.11.4.1 Plotting

1. Create a scatter-plot. The syntax should automatically be included in the result section along with plot. Right-click on the syntax area and select syntax to copy for use in R.

Figure 9: Copy R syntax from Jamovi to use in R.

2. In R, install and load the following packages: **jmv** (Selker et al., 2022) and **scatr** (Selker, 2017). Furthermore, import the data set into R, paste the syntax copied over from Jamovi and run the syntax as shown below.

```
# Use pacman::p_load to load packages (jmv, scatr, jmvcore)
# with optional installation and no package updates
pacman::p_load(jmv, scatr, jmvcore, install = TRUE, update = FALSE)

# Import above-ground carbon data from a CSV or the SAV file
carbon <-
  readr::read_csv(file = here::here("docs", "data", "Filtered.csv"))

carbon_spss_data <-
  haven::read_sav(file = here::here("docs", "data", "Filtered.sav"))

# Syntax from the scatr package to create a scatterplot with linear
# regression lines.
# This code visualizes the relationship between Tree Density, Aboveground
# Carbon and Management Regime.

scatr::scat(
  data = carbon,
  # Specify the dataset (carbon)
  y = "Aboveground_Tree_Carbon_ton_per_ha_per_year",
  # Choose the dependent variable (Y-axis)
  x = "Tree_Density_per_ha",
  # Choose the independent variable (X-axis)
  group = "Management_regime",
  # Group the data by Management Regime
  line = "linear",
  # Add linear regression lines
  se = TRUE
) # Display standard error bars
```

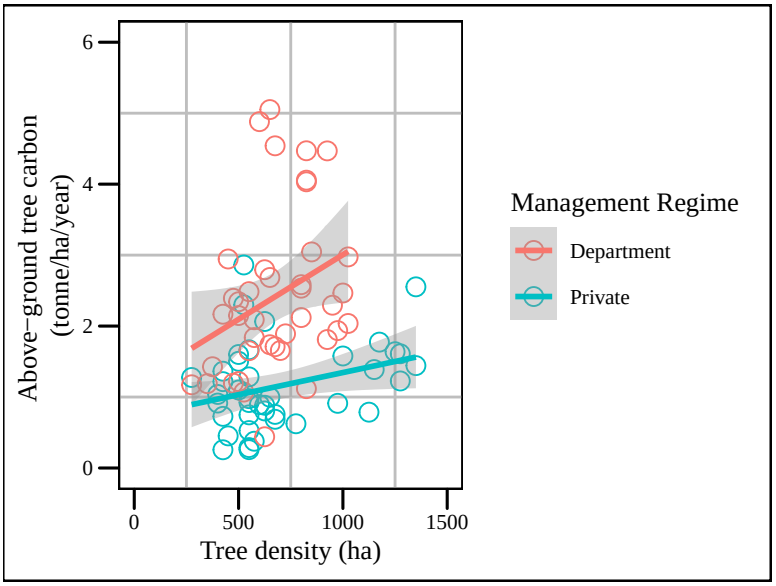


R Code Explanation

- `pacman::p_load(jmv, scatr, jmvcore, install = TRUE, update = FALSE)` is used to load three packages: `jmv`, `scatr`, and `jmvcore`. The `install = TRUE` parameter ensures that these packages are installed if they are not already installed, and `update = FALSE` prevents the packages from being updated.
- `carbon <- readr::read_csv("./data/Filtered.csv")` reads data from a CSV file named "Filtered.csv" located in the "./data" directory and stores it in a variable called "carbon." This line imports the dataset needed for further analysis.
- `scatr::scat(...)` is a function from the `scatr` package used to create a scatterplot with linear regression lines. The code specifies the dataset as "carbon," sets the Y-axis variable to "Aboveground_Tree_Carbon_ton_per_ha_per_year," sets the X-axis variable to "Tree_Density_per_ha," groups the data by "Management_regime," adds linear regression lines, and displays standard error bars. This code is used to visualize the relationship between Tree Density, Aboveground Carbon, and Management Regime in the dataset.

Note that plots created using the `scatr` package has limited functions to further tweak plots. A work around for this issue is to use the `ggplot2` package which has more customizable functions. An example is shown below.

```r
# Use pacman::p_load to load packages (ggthemes, ggplot2, patchwork)
# with optional installation and no package updates
pacman::p_load(ggthemes,
               ggplot2,
               patchwork,
               install = TRUE,
               update = FALSE)

# Import dataset from a CSV file
carbon <- readr::read_csv("./data/Filtered.csv")

# Create a customizable plot using ggplot2
plot1 <- ggplot2::ggplot(
  data = carbon,
  # Specify the dataset (carbon)
  aes(
    # Define aesthetic mappings
    x = carbon$Tree_Density_per_ha,
    # X-axis variable
    y = carbon$Aboveground_Tree_Carbon_ton_per_ha_per_year,
    # Y-axis variable
    col = carbon$Management_regime # Color by Management Regime
  )
) +
  ggplot2::geom_point(size = 3, pch = 21) + # Add points with custom size
  # and shape
  ggplot2::geom_smooth(method = "lm") + # Add a linear regression line
  ggplot2::labs(x = "Tree density (ha)", # Set X-axis label
                y = "Above-ground tree carbon \n(tonne/ha/year)", # Set Y
                #-axis label
                col = "Management Regime") + # Set legend title
  ggplot2::scale_x_continuous(limits = c(0, 1500)) + # Set X-axis limits
  ggplot2::scale_y_continuous(limits = c(0, 6)) + # Set Y-axis limits
  ggthemes::theme_base(base_size = 10, base_family = "serif") +  # Apply custom theme settings
  ggplot2::theme(
    panel.grid.minor = element_line(size = 0.5, color = "grey"),
    # Customize minor grid lines
    axis.title = element_text(size = 10) # Customize axis title text
  )

# Print the customized plot
print(plot1)
```



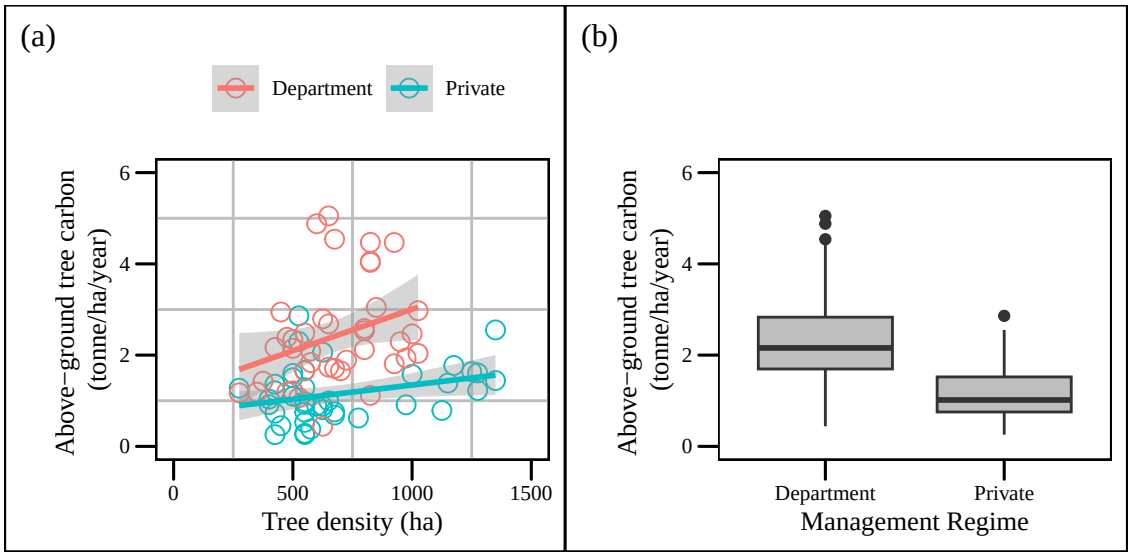
R Code Explanation

- `pacman::p_load(...)` is used to load three packages: **ggthemes**, **ggplot2**, and **patchwork**. The `install = TRUE` parameter ensures that these packages are installed if they are not already installed, and `update = FALSE` prevents the packages from being updated.
- `carbon <- readr::read_csv("./data/Filtered.csv")` reads data from a CSV file named "Filtered.csv" located in the "./data" directory and stores it in a variable called "carbon." This line imports the dataset needed for plotting.
- The `ggplot2::ggplot(...)` function is used to create a customizable plot. It specifies the dataset as "carbon" and defines aesthetic mappings, including X-axis, Y-axis, and color mapping based on the "Management_regime" column.
- Various `ggplot2::geom_` functions are used to add geometrical elements to the plot, such as points and a linear regression line.
- `ggplot2::labs(...)`, `ggplot2::scale_...`, and `ggthemes::theme_...` functions are used to customize plot labels, axis limits, and theme settings.
- Finally, `print(plot1)` prints the customized plot to the output.

2.11.5 Combine multiple plots

```
# Modify the first plot (plot1)
p1 <- plot1 +
  ggplot2::labs(x = "Tree density (ha)", # Change the X-axis label
                y = "Above-ground tree carbon \n(tonne/ha/year)", # Change the Y-axis label
                col = "") # Remove the color legend title

# Create another customizable plot using ggplot2
plot2 <- ggplot2::ggplot(
  data = carbon, # Specify the dataset (carbon)
  aes(
    x = carbon$Management_regime, # X-axis variable
    y = carbon$Aboveground_Tree_Carbon_ton_per_ha_per_year # Y-axis
# variable
  )
```

```r
) +
  ggplot2::geom_boxplot(fill = "grey") + # Add a boxplot with grey fill
  ggplot2::labs(x = "Management Regime", # Set X-axis label
                y = "Above-ground tree carbon \n(tonne/ha/year)") +  # Set Y
  # -axis label
  ggplot2::scale_y_continuous(limits = c(0, 6)) + # Set Y-axis limits
  ggthemes::theme_base(base_size = 10, base_family = "serif") + # Apply custom theme settings
  ggplot2::theme(axis.title = element_text(size = 10)) # Customize axis title text

# Print the second plot (plot2)
# print(plot2) #uncomment the code to run

# Modify the second plot (plot2)
p2 <-
  plot2 + ggplot2::labs(x = "Management Regime", # Change the X-axis label
                        y = "Above-ground tree carbon \n(tonne/ha/year)")

# Combine both modified plots (p1 and p2) using the "&" operator
# Also, customize the legend position and add plot annotations
combined_plots <- p1 + p2 +
  patchwork::plot_annotation(tag_levels = "a",
                             tag_prefix = "(",
                             tag_suffix = ")") &
  theme(legend.position = "top")

# print/ display combined plots
print(combined_plots)
```



R Code Explanation

- `p1` is created by modifying `plot1`. The X-axis label is changed, and the color legend title is removed.
- `plot2` is created as a new customizable plot using `ggplot2`. It specifies the dataset and aesthetic mappings, creates a box-plot, and customizes the plot labels, axis limits, and theme settings.
- `p2` is created by modifying `plot2`. The X-axis label is changed, and the Y-axis label is removed.

- `p1 + p2 & ...` combines both modified plots `p1` and `p2` using the `&` operator. Additional customizations are applied to the combined plot, including changing the legend position and adding plot annotations with tags.

2.11.6 Analyses Syntax

The following syntax and results (image) below were generated in Jamovi. Syntax is further modified in R.

One-Way ANOVA (Non-parametric)

```
jmv::anovaNP(
    formula = Aboveground_Tree_Carbon_ton_per_ha_per_year ~ Plantation_age,
    data = data,
    pairs = TRUE)
```

Kruskal-Wallis

	χ²	df	p
Aboveground_Tree_Carbon_ton_per_ha_per_year	48.8	7	< .001

Dwass-Steel-Critchlow-Fligner pairwise comparisons

Any p value < 0.05 indicates significant differences.

Pairwise comparisons - Aboveground_Tree_Carbon_ton_per_ha_per_year

		W	p
7	8	−4.850	0.014
7	10	−0.428	1.000
7	11	1.710	0.930
7	12	−2.780	0.506
7	17	1.283	0.986
7	18	4.811	0.015
7	21	4.276	0.051
8	10	4.388	0.040
8	11	4.619	0.024
8	12	4.157	0.065
8	17	5.196	0.006
8	18	5.196	0.006
8	21	5.196	0.006
10	11	1.390	0.977
10	12	−0.855	0.999
10	17	2.138	0.802
10	18	4.276	0.051
10	21	4.490	0.032
11	12	−2.886	0.454
11	17	0.535	1.000
11	18	4.383	0.041
11	21	4.169	0.064
12	17	3.314	0.270
12	18	5.345	0.004

Figure 10: Analysis syntax and Kruskal-Wallis test results from Jamovi.

```r
# Load necessary R packages using pacman
pacman::p_load(jmv, # Load the jmv package for statistical analysis
               magrittr, # Load the magrittr package for data manipulation
               install = TRUE, # Install packages if not already installed
               update = FALSE # Do not update packages if newer versions are available
)

# Perform Kruskal-Wallis test with pairwise comparisons
kruskal_pairwise <- jmv::anovaNP(
    formula = Aboveground_Tree_Carbon_ton_per_ha_per_year ~ Plantation_age,  # Specify the formula
    data = carbon, # Specify the dataset (carbon)
    pairs = TRUE   # Perform pairwise comparisons
)

# Display the Kruskal-Wallis significance test results
kruskal_pairwise %>% print()
```

```
##
##  ONE-WAY ANOVA (NON-PARAMETRIC)
##
##  Kruskal-Wallis
##
##                                                    ²           df    p
##
##    Aboveground_Tree_Carbon_ton_per_ha_per_year    50.05889     7    < .0000001
##
##
##
##  DWASS-STEEL-CRITCHLOW-FLIGNER PAIRWISE COMPARISONS
##
##  Pairwise comparisons - Aboveground_Tree_Carbon_ton_per_ha_per_year
##
##                W            p
##
##    7     8     -5.0245113    0.0091010
##    7     10    -0.4276180    0.9999889
##    7     11     1.7104719    0.9296347
##    7     12    -2.7795169    0.5055258
##    7     17     1.2828540    0.9855155
##    7     18     4.8107024    0.0154435
##    7     21     4.2761799    0.0511307
##    8     10     4.5968934    0.0254800
##    8     11     4.7037979    0.0199077
##    8     12     4.0623709    0.0783341
##    8     17     5.3452248    0.0039229
##    8     18     5.3452248    0.0039229
##    8     21     5.3452248    0.0039229
##    10    11     1.3897585    0.9769915
##    10    12    -0.8552360    0.9988368
##    10    17     2.1380899    0.8016074
##    10    18     4.2761799    0.0511307
##    10    21     4.4899889    0.0323768
##    11    12    -2.8864214    0.4543784
##    11    17     0.5345225    0.9999489
##    11    18     4.3830844    0.0408393
##    11    21     4.1692754    0.0635316
##    12    17     3.3140394    0.2701716
##    12    18     5.3452248    0.0039229
##    12    21     4.7037979    0.0199077
##    17    18     4.8107024    0.0154435
##    17    21     4.0623709    0.0783341
##    18    21     1.6035675    0.9494883
##
```

R Code Explanation

- **`pacman::p_load(...)`** is used to load the required R packages, including **`jmv`** for statistical analysis and **`magrittr`** for data manipulation. The **`install`** parameter is set to **`TRUE`** to install the packages if they are not already installed, and **`update`** is set to **`FALSE`** to prevent updating packages.
- **`kruskal_pairwise`** stores the result of the Kruskal-Wallis test with pairwise comparisons. It calculates the Kruskal-Wallis test for the specified formula and dataset, and the **`pairs = TRUE`** argument indicates

that pairwise comparisons should be performed.

- `kruskal_pairwise %>% print()` is used to display the Kruskal-Wallis significance test results directly without the need to extract them into a data frame.

3 Chapter 2: Data Import and Cleaning

3.1 Introduction

Chapter 2 focuses on the critical aspects of data preparation for ecological data analysis. Here, you will learn how to efficiently import ecological datasets into R and Jamovi. Additionally, you'll explore techniques for cleaning and preprocessing data, ensuring that your analyses are based on high-quality, error-free data. By the end of this chapter, you will have:

- Acquired the skills to import data from various file formats.
- Understood the importance of data cleaning and preprocessing.
- Applied techniques to handle missing data, outliers, and data transformations.

3.2 Importing Data into R

3.2.1 Data Import Overview

Data import is a critical initial step in ecological research, as it involves bringing external data into your analysis environment (typically R or a similar software) for examination, manipulation, and analysis. The significance of data import in ecological research is multifaceted:

1. **Data Collection and Compilation**: Ecological research often requires gathering data from various sources, such as field surveys, remote sensing, weather stations, or pre-existing databases. Data import is the process of bringing these diverse datasets together for a comprehensive analysis.
2. **Data Integration**: Ecologists work with heterogeneous datasets, including numeric measurements, spatial coordinates, categorical variables, and textual descriptions. Data import allows you to integrate these diverse data types into a unified dataset, making it ready for analysis.
3. **Quality Assurance**: Imported data may contain errors, missing values, outliers, or inconsistencies. During the import process, researchers can identify and address data quality issues, ensuring the integrity and reliability of subsequent analyses.
4. **Data Exploration**: Once imported, data can be visualized and explored to gain insights into patterns, trends, and relationships. Effective data import facilitates initial exploratory data analysis (EDA), helping researchers decide on suitable statistical methods.
5. **Statistical Analysis**: Ecological research often involves a wide range of statistical techniques, from simple descriptive statistics to advanced modeling. To apply these methods, data must be in a format that allows for statistical analysis, which data import achieves.
6. **Communication and Reporting**: Accurate and well-organized data is crucial for communicating research findings to peers, policymakers, or the public. Data import ensures that data is in a format conducive to creating meaningful charts, graphs, and reports.

3.2.2 Common Data Sources in Ecological Research

1. **Field Surveys**: Researchers collect primary data through field surveys, which can include measurements of species abundance, biodiversity, habitat characteristics, and environmental variables like temperature and precipitation.

2. **Remote Sensing**: Satellite and aerial imagery, LiDAR (Light Detection and Ranging), and drones provide remote sensing data that can be used to monitor land cover, vegetation health, and changes in the environment over time.
3. **Climate and Weather Data**: Climate and weather data, obtained from weather stations, are essential for studying the effects of climate change on ecosystems and wildlife behavior.
4. **Geospatial Data**: Geographic Information Systems (GIS) data, including spatial coordinates, topography, and land use, are often used to study spatial patterns and relationships within ecosystems.
5. **Government Databases**: Government agencies and organizations maintain extensive ecological datasets, including data on wildlife populations, conservation areas, and environmental regulations.
6. **Scientific Literature**: Ecologists may extract data from published research papers or online databases, such as GenBank for genetic data or Global Biodiversity Information Facility (GBIF) for species occurrence records.
7. **Social Surveys**: Ecological research may incorporate social surveys to assess human interactions with ecosystems, such as visitor behavior in national parks or community perceptions of environmental issues.
8. **Laboratory Experiments**: Experimental data from controlled laboratory studies are used to investigate specific ecological hypotheses.

In summary, data import in ecological research serves as the gateway to the analysis and interpretation of complex ecological systems. It allows researchers to leverage diverse data sources, address data quality concerns, and ultimately gain a deeper understanding of the natural world.

3.2.3 Importing Flat Files (CSV and Excel)

```r
# Load the readr package (if not already loaded)
library(readr)

# Set the file path to your CSV file
file_path <- "path/to/your/data.csv"

# Import the CSV file into a data frame
csv_data <- readr::read_csv(file_path)

# View the imported data
head(csv_data)

# Load the readxl package (if not already loaded)
library(readxl)

# Set the file path to your Excel file
file_path <- "path/to/your/data.xlsx"

# Import the Excel file into a data frame (assuming the data is in the first sheet)
xlsx_data <- readxl::read_excel(file_path, sheet = 1)

# View the imported data
head(xlsx_data)
```

R Code Explanation

For Importing a CSV File:

1. **Load the `readr` Package:** This line (line 2) loads the **`readr`** package, which is used for reading CSV files.
2. **Set the File Path:** In line 5, you need to set the file path to your CSV data file. Replace **`"path/to/your/data.csv"`** with your actual file path.
3. **Import the CSV File:** Line 8 uses the **`read_csv`** function from the **`readr`** package to read the CSV file specified by **`file_path`** and stores it in the **`csv_data`** data frame.
4. **View the Imported Data:** Line 11 displays the first few rows of the imported CSV data using the **`head`** function, allowing you to inspect the data.

For Importing an Excel File:

5. **Load the `readxl` Package:** This line (line 14) loads the **`readxl`** package, which is used for reading Excel files.
6. **Set the File Path:** In line 17, you should set the file path to your Excel data file. Replace **`"path/to/your/data.xlsx"`** with your actual file path.
7. **Import the Excel File:** Line 20 uses the **`read_excel`** function from the **`readxl`** package to read the Excel file specified by **`file_path`**. The imported data is stored in the **`xlsx_data`** data frame.
8. **View the Imported Data:** Line 23 displays the first few rows of the imported Excel data using the **`head`** function for initial data exploration.

These steps guide you through the process of importing data from CSV and Excel files into R, allowing you to work with your ecological datasets. Remember to replace the file paths with your actual file paths when applying these steps in your R script or environment.

3.2.4 Importing Data from Databases (e.g., SQL Databases)

```r
# Load the DBI and odbc packages (if not already loaded)
library(DBI)
library(odbc)

# Define the database connection details
db_connection <- dbConnect(
  odbc::odbc(),
  Driver = "Your_Database_Driver",
  # e.g., "SQL Server" or "PostgreSQL"
  Server = "Your_Server_Address",
  Database = "Your_Database_Name",
  UID = "Your_Username",
  PWD = "Your_Password"
)

# Check the database connection
dbIsValid(db_connection)

# Specify the SQL query to retrieve data from a table
sql_query <- "SELECT * FROM Your_Table_Name"

# Execute the SQL query and import the data into a data frame
db_data <- dbGetQuery(db_connection, sql_query)

# View the imported data
head(db_data)
```

R Code Explanation

1. **Load DBI and odbc Packages**
   - These lines load the necessary R packages for working with databases. `DBI` is a database interface, and `odbc` is a package for database connectivity.
2. **Define the Database Connection Details**
   - `db_connection <- dbConnect(odbc::odbc(), ...)` sets up a connection to a database. You need to specify details such as the database driver, server address, database name, username, and password.
   - Replace `"Your_Database_Driver"`, `"Your_Server_Address"`, `"Your_Database_Name"`, `"Your_Username"`, and `"Your_Password"` with your actual database information.
3. **Check the Database Connection**
   - `dbIsValid(db_connection)` verifies if the database connection is valid. It will return `TRUE` if the connection is successful.
4. **Specify the SQL Query:**
   - `sql_query <- "SELECT * FROM Your_Table_Name"` sets up an SQL query to retrieve data from a specific table in your database.
   - Replace `"Your_Table_Name"` with the actual name of the table you want to query.
5. **Execute the SQL Query and Import Data**
   - `db_data <- dbGetQuery(db_connection, sql_query)` executes the SQL query on the database server and imports the result into an R data frame called `db_data`.
6. **View the Imported Data**
   - `head(db_data)` displays the first few rows of the imported data frame, allowing you to inspect the data.

These codes are used to establish a connection to a database, retrieve data from it using an SQL query, and bring that data into R for further analysis. Remember to replace the placeholders with your actual database information.

Note that structured data management is of paramount importance in ecological research, and here's why it deserves special attention:

1. **Data Quality Assurance**: Structured data management ensures that the data you work with is accurate, reliable, and consistent. This includes addressing issues like missing values, outliers, duplicates, and data entry errors. High-quality data is essential for making sound ecological inferences and drawing reliable conclusions.
2. **Data Integrity**: Managing data in a structured manner preserves its integrity throughout the research process. It reduces the risk of inadvertent changes, deletions, or overwrites that can compromise the reliability of your findings.
3. **Reproducibility**: Structured data management facilitates research reproducibility. When others attempt to replicate your research or when you revisit your own work after some time, well-structured data ensures that you can easily understand, replicate, and build upon your previous analyses.
4. **Data Traceability**: Structured data management often involves proper documentation of data sources, collection methods, and transformations. This traceability is critical for establishing the credibility and transparency of your research.

5. **Efficient Analysis**: Structured data is easier to work with, reducing the time and effort required for data cleaning, preprocessing, and analysis. Researchers can focus on the ecological questions and insights rather than wrestling with messy data.
6. **Collaboration**: If your research involves collaboration with other researchers or teams, structured data management becomes indispensable. It ensures that everyone is on the same page regarding data formats, variable definitions, and data handling protocols.
7. **Long-Term Data Preservation**: Ecological research often spans long periods. Properly structured data ensures that data can be preserved and reused over time, even as technology and personnel change.
8. **Data Sharing and Accessibility**: In many cases, ecological research data is of broader interest and value to the scientific community, policymakers, and conservationists. Well-structured data can be more easily shared, reused, and made accessible to a wider audience.
9. **Statistical Analysis**: Structured data is a prerequisite for conducting meaningful statistical analyses. Many statistical tools and software packages require data to be organized in a particular format. Structured data management ensures that your data is analysis-ready.
10. **Ethical Considerations**: Ethical research practices often include data protection and privacy considerations. Structured data management helps in anonymizing and securing sensitive data as needed, ensuring compliance with ethical guidelines.

In summary, structured data management is the foundation upon which ecological research is built. It ensures data quality, facilitates analysis, promotes transparency, and enhances the overall credibility and impact of your research. Researchers who invest in structured data management are better equipped to make significant contributions to the understanding and conservation of ecosystems.

3.3 Importing Data into Jamovi

3.3.1 Jamovi Data Import

Jamovi simplifies data import with its user-friendly interface, making it accessible to users who may not be familiar with coding or complex data manipulation.

- **Importing CSV and Excel Files in Jamovi**

3.4 Data Cleaning

3.4.1 Data Cleaning Overview

Data cleaning is of paramount significance in ecological analysis due to its substantial impact on research outcomes. Here's why data cleaning is essential in ecological research:

1. **Data Quality Assurance:** Raw ecological data often contain errors, inaccuracies, or inconsistencies due to various factors, such as measurement errors, sensor malfunctions, or human errors during data collection. Data cleaning is the process of identifying and rectifying these issues, ensuring that the data accurately represent the ecological phenomena under study.
2. **Accurate Analyses:** Cleaned data provide a reliable foundation for statistical analyses and modeling. Without data cleaning, the results of analyses may be misleading or erroneous, potentially leading to incorrect conclusions about ecological relationships or trends.
3. **Reducing Bias:** Incomplete or erroneous data can introduce bias into research outcomes. Data cleaning helps reduce this bias, ensuring that the results are more representative of the true ecological conditions.
4. **Enhanced Interpretability:** Cleaned data are easier to interpret and visualize. Researchers can trust that patterns and relationships observed in the data are genuine and not artifacts of data errors or anomalies.

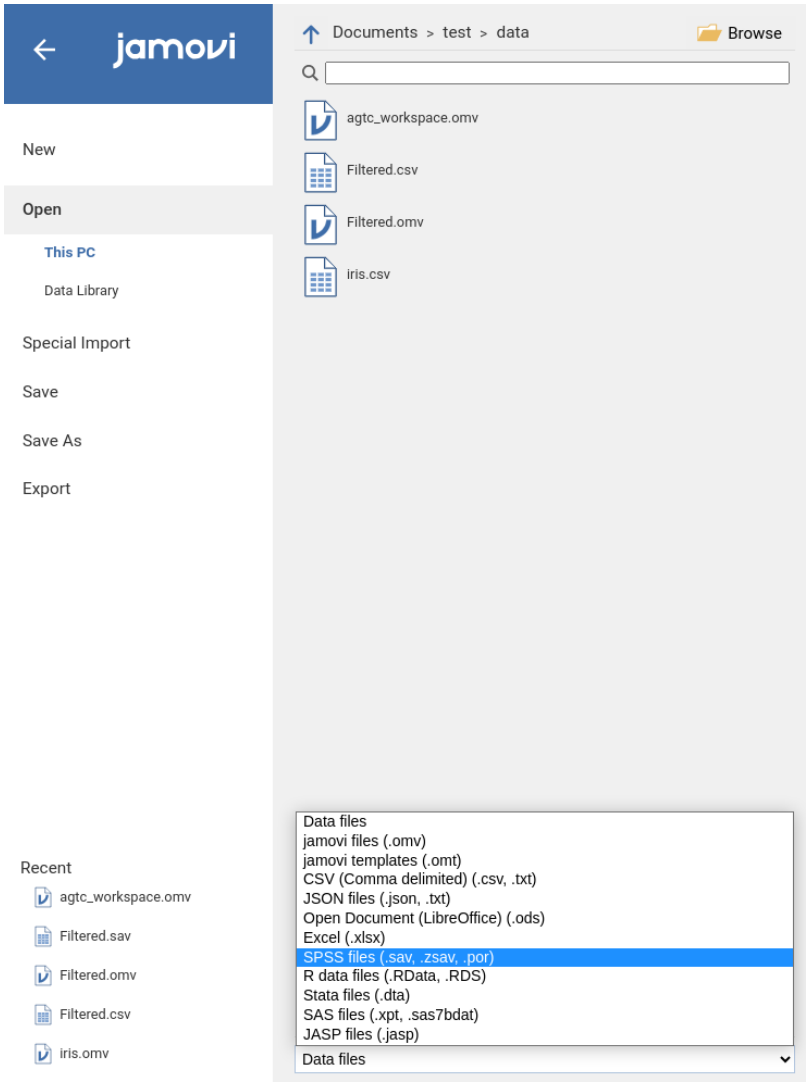


Figure 11: Example of importing data into Jamovi.

5. **Effective Data Visualization:** Data cleaning improves the quality of data visualization. Visualizations, such as graphs and charts, are crucial for conveying ecological findings. Cleaned data enable researchers to create informative and accurate visual representations.
6. **Comparability:** Cleaned data allow for meaningful comparisons within and between ecological studies. Researchers can confidently compare data from different sources or time periods, knowing that data quality issues have been addressed.
7. **Scientific Credibility:** Ecological research relies on the credibility of findings. Cleaned data enhance the scientific rigor of a study, increasing confidence in its results and conclusions.
8. **Effective Decision-Making:** Ecological research often informs conservation efforts, policy decisions, and resource management. Cleaned data ensure that decisions made based on research outcomes are well-founded and have a positive impact on ecosystems and biodiversity.
9. **Data Archiving:** High-quality, cleaned data are more suitable for long-term archiving and sharing with the scientific community. Properly cleaned and documented data sets can contribute to broader ecological knowledge and support future research.
10. **Time and Resource Efficiency:** Although data cleaning requires effort, it ultimately saves time and resources during analysis. Cleaned data lead to more efficient and accurate statistical procedures.

In ecological analysis, where data are collected in diverse and often challenging environments, data cleaning is a critical step in the research process. It transforms raw, potentially problematic data into a reliable foundation for meaningful analysis, interpretation, and the generation of ecological insights that contribute to our understanding of the natural world.

3.4.2 Handling Missing Data

Handling missing data in ecological research is crucial to ensure the integrity and validity of your analyses. Here are techniques for identifying and handling missing data:

1. **Identifying Missing Data**
   - **Summary Statistics:** Calculate summary statistics such as mean, median, and standard deviation for each variable. Missing data will be indicated by "NA" or "NaN" values in R or blank cells in spreadsheets.
   - **Visualization:** Create visualizations like histograms or bar plots to visualize the distribution of missing data for each variable. This can reveal patterns of missingness.
   - **Missing Data Packages:** R packages like `naniar`, `skimr` or `VIM` provide functions and plots specifically designed for visualizing and understanding missing data patterns.
2. **Handling Missing Data**
   - **Removal:** Sometimes, the simplest approach is to remove observations (rows) or variables (columns) with missing data. However, this should be done judiciously, as it can lead to loss of valuable information.
   - **Imputation:** Imputation involves estimating missing values based on observed data. Common imputation methods include mean imputation (replacing missing values with the mean of the variable), median imputation, or regression imputation (predicting missing values based on other variables). R packages like `mice` and `missForest` provide powerful imputation tools.
   - **Data Augmentation:** In Bayesian statistics, data augmentation techniques can be used to account for missing data by treating them as additional parameters to be estimated.
   - **Interpolation/Extrapolation:** In time series or spatial data, missing values can often be estimated through interpolation (estimating values within the range of observed data) or extrapolation (estimating values beyond the range of observed data).

- **Multiple Imputation:** This advanced technique involves creating multiple datasets with different imputed values for missing data and analyzing each dataset separately. Results are then combined to account for uncertainty due to missing data. The `mice` package in R is commonly used for multiple imputation.

3. **Exploring Patterns of Missing Data**
    - **Missing Data Heatmaps:** Use heat-maps to visualize the patterns of missing data in your dataset. This helps identify if missingness is random or if there are systematic patterns related to specific variables or time periods.
    - **Missing Data by Subgroup:** Explore if missingness varies by subgroups within your data (e.g., by location, species, or time). Understanding these patterns can inform imputation strategies.
    - **Missing Data Mechanisms:** Consider the mechanism behind missing data. Is it missing completely at random (MCAR), missing at random (MAR), or not missing at random (NMAR)? This information can guide imputation methods.
    - **Sensitivity Analysis:** Perform sensitivity analyses to assess how different imputation methods or missing data assumptions impact your results. This helps quantify the uncertainty associated with missing data.
    - **Collect More Data:** In some cases, the best solution is to collect more data to reduce missingness in critical variables.

Remember that there is no one-size-fits-all approach to handling missing data in ecological research. The choice of method should depend on the nature of the data, the extent of missingness, and the goals of your analysis. Transparently report your methods for handling missing data in research publications to ensure the reproducibility of your findings.

3.4.3 Outlier Detection and Treatment

Outliers are data points that significantly deviate from the rest of the data in a dataset. Detecting and addressing outliers is essential in ecological research for several reasons:

Importance of Addressing Outliers

1. **Influence on Statistics:** Outliers can strongly influence summary statistics like the mean and standard deviation, leading to biased estimates. This can affect the interpretation of ecological patterns and relationships.
2. **Model Assumptions:** Many statistical models assume that the data follow a certain distribution or have homogeneous variances. Outliers violate these assumptions, potentially leading to incorrect model inferences.
3. **Ecological Significance:** Outliers may represent rare or unusual ecological events that are of particular interest or concern. Identifying and understanding these outliers can be critical for ecological research.

Methods for Detecting Outliers

1. **Visual Inspection:** The simplest method is to create visualizations like scatter plots, box plots, or histograms. Outliers often appear as data points far from the main cluster or as individual points outside the whiskers of a box plot.
2. **Z-Scores:** Calculate the Z-score for each data point, which measures how many standard deviations a data point is from the mean. Data points with Z-scores beyond a certain threshold (e.g., $|Z| > 2$ or 3) are considered outliers.
3. **Tukey's Method:** Tukey's method uses the Interquartile Range (IQR) to detect outliers. Data points outside the range defined by Q1 - 1.5 * IQR and Q3 + 1.5 * IQR are considered outliers, where Q1 and Q3 are the first and third quartiles, respectively.

4. **Modified Z-Scores:** In cases where data are not normally distributed, modified Z-scores like the Median Absolute Deviation (MAD) can be more robust for outlier detection.

Techniques for Outlier Treatment

1. **Removal:** The simplest approach is to remove outliers from the dataset. However, this should be done cautiously and with justification, as removing data points can lead to information loss.
2. **Transformation:** Transforming the data using mathematical functions (e.g., logarithm) can reduce the impact of outliers and make the data more amenable to analysis.
3. **Winsorization:** Winsorization replaces extreme values with values closer to the rest of the data (e.g., setting all values above a certain threshold to that threshold). This approach preserves the data distribution while mitigating the influence of outliers.
4. **Robust Statistical Methods:** Robust statistical methods, such as robust regression or robust estimation of central tendency and variance, are less influenced by outliers and provide more reliable estimates.
5. **Modeling Approaches:** In some cases, it may be appropriate to model outliers explicitly as a separate group or to use models that are robust to outliers.
6. **Reporting:** Regardless of the approach chosen, it is essential to transparently report how outliers were handled in research publications to ensure the reproducibility and credibility of the analysis.

The choice of outlier detection and treatment methods should depend on the nature of the data and the research objectives. It is advisable to perform sensitivity analyses to assess how different outlier strategies impact research findings.

3.5 Data Preprocessing

3.5.1 Data Preprocessing Overview

Data preprocessing refers to a set of procedures and techniques used to clean, transform, and prepare raw data for analysis. It plays a pivotal role in the data analysis process, as the quality and structure of the data significantly influence the outcomes of statistical analyses and machine learning models.

The key steps in data preprocessing include data cleaning (dealing with missing values and outliers), data transformation (changing the format or distribution of data), and data reduction (reducing the volume but preserving the key information). Data preprocessing aims to ensure that the data is in a suitable form for analysis, making it more interpretable and increasing the accuracy and reliability of analytical results.

3.5.1.1 Data Transformation Data transformation involves altering the data values to meet the assumptions of statistical analysis. Various data transformation techniques can be applied in ecological research:

1. **Log Transformation:** Logarithmic transformation is commonly used to stabilize variance, make the data more symmetric, and approximate a normal distribution. It is particularly useful when dealing with data that exhibit exponential growth or decay, such as species abundance data or tree growth rates.
2. **Square Root Transformation:** Similar to log transformation, square root transformation can be used to stabilize variance and approximate normality. It is effective when dealing with count data or data with non-constant variance.
3. **Box-Cox Transformation:** The Box-Cox transformation is a family of power transformations that can be applied to make data conform to normality assumptions. It includes both logarithmic and square root transformations as special cases. The optimal transformation is selected based on maximum likelihood estimation.

4. **Arcsine Transformation:** Arcsine square root transformation is used for proportional data or data with bounded values (e.g., percentage data). It can make the data more symmetric and suitable for parametric tests.
5. **Exponential Transformation:** When dealing with data that follows a decay process, an exponential transformation can be applied to linearize the relationship.

3.5.1.1.1 When to Apply Data Transformations Data transformations should be applied when:

- Data violate assumptions of normality or homoscedasticity required for parametric tests.
- Data exhibit heteroscedasticity (variance changes with the level of a variable).
- The research question or theoretical considerations suggest that a particular transformation is appropriate (e.g., log-transforming biomass data).

3.5.1.1.2 Improvement of Normality and Homoscedasticity Data transformations can improve normality by reducing skewness and kurtosis in the data distribution. Normality is essential for parametric tests like t-tests and ANOVA, which assume that data follow a normal distribution.

Transformations can also help in achieving homoscedasticity, where the variance of the residuals is constant across levels of an independent variable. This is crucial for linear regression and ANOVA, as violations of homoscedasticity can lead to incorrect inferences.

3.5.2 Scaling and Centering

- **Scaling:** Scaling variables involves transforming them to have a common scale or range, typically between 0 and 1 or with a mean of 0 and a standard deviation of 1. Scaling is essential when variables have different units or scales because it ensures that all variables contribute equally to analyses like clustering or principal component analysis.
- **Centering:** Centering involves subtracting the mean of a variable from each data point. Centering is useful when interpreting regression coefficients because it makes the intercept more interpretable. In the context of multiple regression, centering can reduce multicollinearity between predictor variables.

3.5.2.1 Why Scaling Is Essential for Variables with Different Units When variables have different units or scales, their magnitudes can dominate the results of certain analyses. Scaling ensures that all variables are treated equally, preventing larger variables from unduly influencing the outcomes. It also facilitates the interpretation of coefficients in regression models because the coefficients represent the effect of a one-unit change in the predictor variable while holding other variables constant. Scaling ensures that this one-unit change is consistent across all predictors, regardless of their units or scales.

3.6 Conclusion

Chapter 2 of "Exploring Ecological Data with R and Jamovi" emphasizes the critical importance of successful data import, cleaning, and preprocessing in ecological data analysis. Here are the key takeaways:

1. **Data Import Significance:** Data import is the initial step in any data analysis project. Ecological researchers often deal with diverse data sources, including flat files (e.g., CSV, Excel) and databases. Accurate data import ensures that you have access to the necessary information for analysis.
2. **Common Data Sources:** Ecological research commonly involves data from various sources, including field observations, experiments, and remote sensing. Understanding how to import data from these sources is essential for ecologists.
3. **R and Jamovi Integration:** Both R and Jamovi offer user-friendly approaches to data import. Jamovi provides a point-and-click interface, while R offers versatile functions like `readr` and `readxl` for importing data from flat files.

4. **Database Connection:** For larger datasets stored in relational databases, knowing how to connect to and import data from databases is crucial. R provides packages like DBI and odbc for this purpose.
5. **Structured Data Management:** Structured data management ensures that your data is organized, consistent, and error-free. This process involves tasks such as handling missing data, identifying and treating outliers, and transforming data when necessary.
6. **Missing Data Handling:** Missing data can impact the validity of your analyses. Techniques like data imputation, removal of missing values, and exploring patterns of missingness are essential for handling missing data effectively.
7. **Outlier Detection and Treatment:** Outliers can distort statistical analyses and lead to inaccurate conclusions. Visual inspection, Z-scores, and Tukey's method are valuable tools for identifying and addressing outliers.
8. **Data Transformation:** Data transformation techniques like log transformation, square root transformation, and Box-Cox transformation can help meet assumptions of normality and homoscedasticity, improving the reliability of statistical analyses.
9. **Scaling and Centering:** Scaling variables to a common range and centering variables around their mean are important for ensuring that variables with different units or scales are treated equally in analyses.
10. **Key Emphasis:** Successful data import, cleaning, and preprocessing are essential steps to ensure that ecological analyses are based on accurate and reliable data. These steps lay the foundation for meaningful and trustworthy research outcomes.

By mastering these data management techniques, you are now well-prepared to explore and analyze ecological datasets with confidence, setting the stage for robust ecological research.

4 Chapter 3: Exploratory Data Analysis (EDA)

4.1 Introduction

In Chapter 3, you will embark on a journey into the world of Exploratory Data Analysis (EDA) for ecological data. EDA is a crucial step that allows you to understand and gain insights from your datasets before diving into formal statistical analyses. By the end of this chapter, you will have:

- Mastered various visualization techniques for ecological data.
- Learned how to summarize and describe your data effectively.
- Discovered patterns, relationships, and outliers within your datasets.

4.2 The Importance of EDA

4.2.1 Understanding EDA

Exploratory Data Analysis (EDA) is a critical phase in the data analysis process that involves the initial examination, visualization, and summary of data. It serves as a fundamental tool in ecological research and data analysis. Here's an explanation of the concept and its significance in ecological research:

4.2.1.1 Concept of EDA

- EDA is an approach used to understand the main characteristics of a dataset before applying more complex statistical methods.
- It aims to uncover patterns, relationships, anomalies, and other insights within the data.
- EDA employs a combination of graphical and numerical techniques to achieve these objectives.

- It is a crucial step in the data analysis pipeline, allowing researchers to formulate hypotheses and make informed decisions about subsequent analyses.

4.2.1.2 Significance of EDA in Ecological Research

- Ecological datasets are often complex and multidimensional, containing numerous variables and data points.
- EDA helps researchers gain an initial understanding of the dataset's structure and content.
- It aids in identifying potential data quality issues such as outliers, missing values, or data inconsistencies.
- EDA enables the discovery of trends and patterns within ecological data, which can guide further analysis.
- Through visualization, EDA helps in the communication of results to both scientific and non-scientific audiences.
- EDA can highlight relationships between ecological variables, supporting the formulation of research questions and hypotheses.
- In ecological research, where the influence of environmental factors on ecosystems is studied, EDA is crucial for uncovering insights that can drive conservation efforts and environmental management decisions.

4.2.2 EDA Workflow

The typical workflow of an EDA process in ecological research involves the following steps:

1. **Data Collection:** Gather ecological data from various sources, such as field surveys, experiments, or remote sensing.
2. **Data Cleaning and Preprocessing:** As discussed in Chapter 2, prepare the data by handling missing values, identifying and treating outliers, and performing necessary data transformations.
3. **Univariate Analysis:** Begin with a univariate analysis, which involves exploring each variable individually. Compute summary statistics, generate histograms, density plots, and box plots to understand the distribution and central tendency of each variable.
4. **Bivariate Analysis:** Move on to bivariate analysis to examine relationships between pairs of variables. Scatter plots, correlation matrices, and cross-tabulations can reveal associations between ecological factors.
5. **Multivariate Analysis:** Explore relationships involving multiple variables simultaneously. Techniques like principal component analysis (PCA) or multidimensional scaling (MDS) can provide insights into complex data structures.
6. **Visualization:** Utilize data visualization tools, such as scatter plots, bar charts, heat-maps, and spatial maps, to create visual representations of ecological patterns and trends.
7. **Hypothesis Generation:** Based on insights gained from EDA, generate hypotheses about ecological processes, interactions, or correlations that warrant further investigation.
8. **Summary and Reporting:** Summarize key findings and insights from the EDA process. Create reports, presentations, or visuals to communicate the results to stakeholders, colleagues, or the broader scientific community.
9. **Iterative Process:** EDA is often iterative, as insights from initial analysis may lead to further questions or refinements in subsequent analyses.

In ecological research, EDA serves as a powerful tool for uncovering hidden insights within complex datasets, guiding subsequent analyses, and informing ecological decision-making processes. It enables researchers to make data-driven conclusions and contributes to a deeper understanding of ecological systems and environmental dynamics.

4.3 Data Visualization

4.3.1 Data Visualization Overview

Data visualization is a cornerstone of Exploratory Data Analysis (EDA) in ecological research. It is a powerful technique that allows researchers to represent complex data visually, making it easier to understand and interpret. Here's an overview of the significance of data visualization in EDA:

- **Understanding Complex Data:** Ecological datasets often contain numerous variables and data points. Visualization provides a means to simplify complex data structures and reveal patterns, trends, and relationships that might be hidden in raw numbers.
- **Quality Assessment:** Visualization aids in identifying data quality issues, such as outliers, missing values, and anomalies. Visual cues can highlight problematic data points for further investigation.
- **Hypothesis Generation:** Visual exploration of data can spark hypotheses and research questions. Researchers can form initial insights into ecological processes and phenomena, guiding subsequent analyses.
- **Effective Communication:** Visual representations of data are powerful tools for communicating research findings to both scientific and non-scientific audiences. Clear and compelling visuals enhance the impact of ecological research.

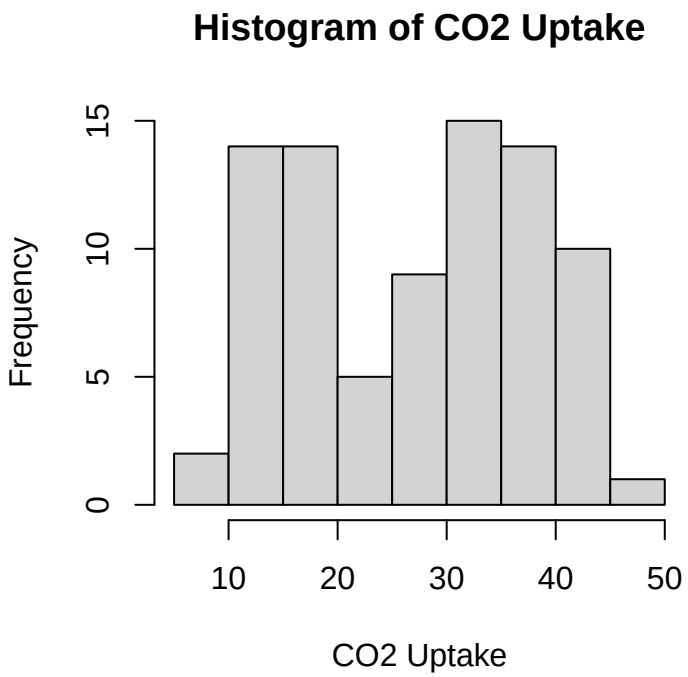
Now, let's delve into various aspects of data visualization, including univariate, bivariate, and multivariate visualization techniques, using the CO2 dataset in R.

4.3.2 Univariate Visualization

Univariate visualization focuses on visualizing single variables to understand their distributions and characteristics. Here are some common techniques and interpretations:

- **Histograms:** Histograms display the frequency distribution of a single variable. They help visualize the shape (e.g., normal, skewed) and central tendency (e.g., mean, median) of the data.

```r
# Load the CO2 dataset
data("CO2")

# Create a histogram of CO2 uptake
hist(CO2$uptake, main = "Histogram of CO2 Uptake", xlab = "CO2 Uptake")
```



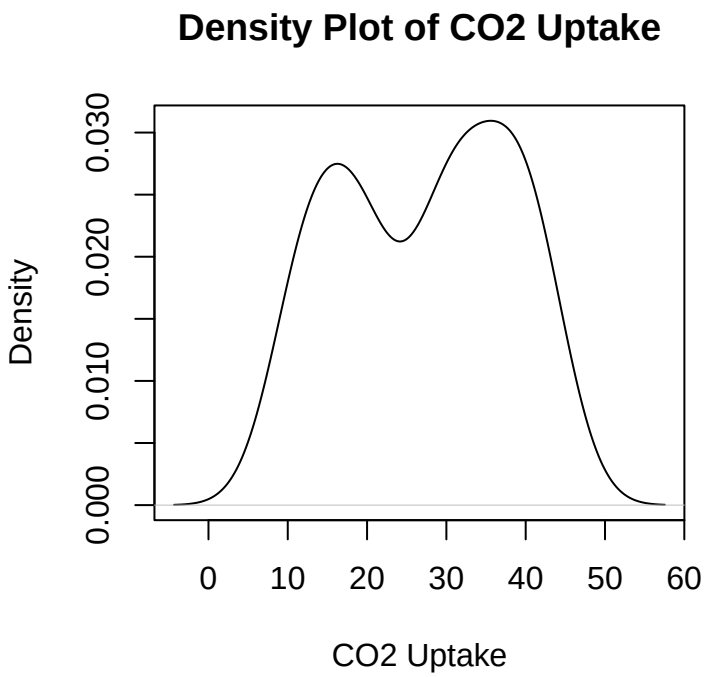
R Code Explanation

- `data("CO2")` loads the built-in CO2 dataset into your R environment. This dataset contains measurements related to the uptake of carbon dioxide (CO2) by different plants under varying conditions.
- `hist(CO2$uptake, ...)` creates a histogram of the "uptake" variable within the CO2 dataset. Specifically:
  - `CO2$uptake` extracts the "uptake" column (variable) from the CO2 dataset, which represents the CO2 uptake measurements.
  - `hist(...)` is the function used to create histograms in R.
- `main = "Histogram of CO2 Uptake"` specifies the main title of the histogram, which is displayed at the top of the plot. In this case, it's titled "Histogram of CO2 Uptake."
- `xlab = "CO2 Uptake"` labels the x-axis of the histogram, providing a description of what the x-axis represents. Here, it indicates that the x-axis represents CO2 uptake.

This code, when executed, will load the CO2 dataset and then generate a histogram showing the distribution of CO2 uptake measurements. The histogram's title will be "Histogram of CO2 Uptake," and the x-axis will be labeled "CO2 Uptake," making the plot informative and easy to understand.

- **Density Plots:** Density plots provide smoothed representations of data distributions. They are useful for visualizing data density and identifying modes (peaks) in the distribution.

```
# Create a density plot of CO2 uptake
plot(density(CO2$uptake), main = "Density Plot of CO2 Uptake", xlab = "CO2 Uptake")
```



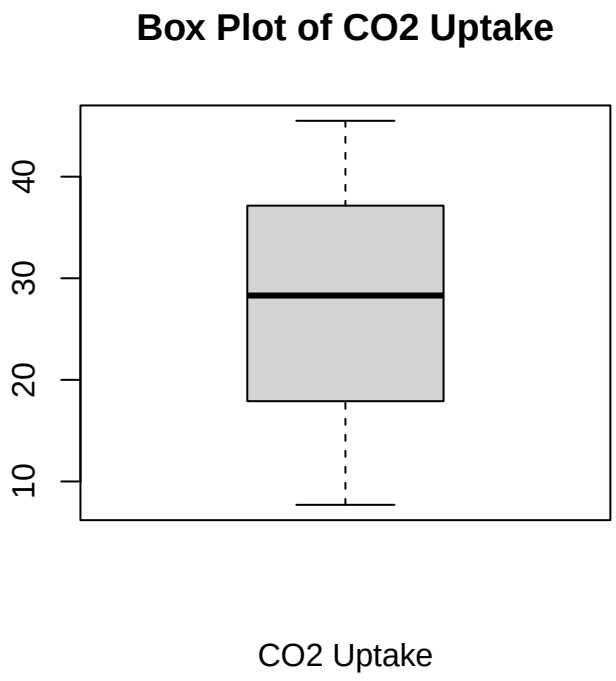
R Code Explanation

- `plot(density(CO2$uptake), ...)` creates a density plot (also known as a kernel density plot) of the "uptake" variable within the CO2 dataset. Specifically:
  - `density(CO2$uptake)` calculates the density estimate for the "uptake" variable, representing the distribution of CO2 uptake measurements. This estimate is what the density plot will be based on.
  - `plot(...)` is used to create plots in R.
- `main = "Density Plot of CO2 Uptake"` specifies the main title of the density plot, which is displayed at the top of the plot. In this case, it's titled "Density Plot of CO2 Uptake."
- `xlab = "CO2 Uptake"` labels the x-axis of the density plot, providing a description of what the x-axis represents. Here, it indicates that the x-axis represents CO2 uptake.

When this code is executed, it will calculate the density estimate of CO2 uptake measurements and create a density plot to visualize the distribution. The main title will be "Density Plot of CO2 Uptake," and the x-axis will be labeled "CO2 Uptake," making the plot informative and easy to interpret.

- **Box Plots:** Box plots summarize the distribution of a variable, showing its median, quartiles, and potential outliers. They are effective for identifying skewness and outlier presence.

```
# Create a box plot of CO2 uptake
boxplot(CO2$uptake, main = "Box Plot of CO2 Uptake", xlab = "CO2 Uptake")
```



R Code Explanation

- `boxplot(CO2$uptake, ...)` creates a box plot of the "uptake" variable within the CO2 dataset. Specifically:
  - `CO2$uptake` specifies the variable to be plotted, which is CO2 uptake in this case.
- `main = "Box Plot of CO2 Uptake"` specifies the main title of the box plot, which is displayed at the top of the plot. In this case, it's titled "Box Plot of CO2 Uptake."
- `xlab = "CO2 Uptake"` labels the x-axis of the box plot, providing a description of what the x-axis represents. Here, it indicates that the x-axis represents CO2 uptake.

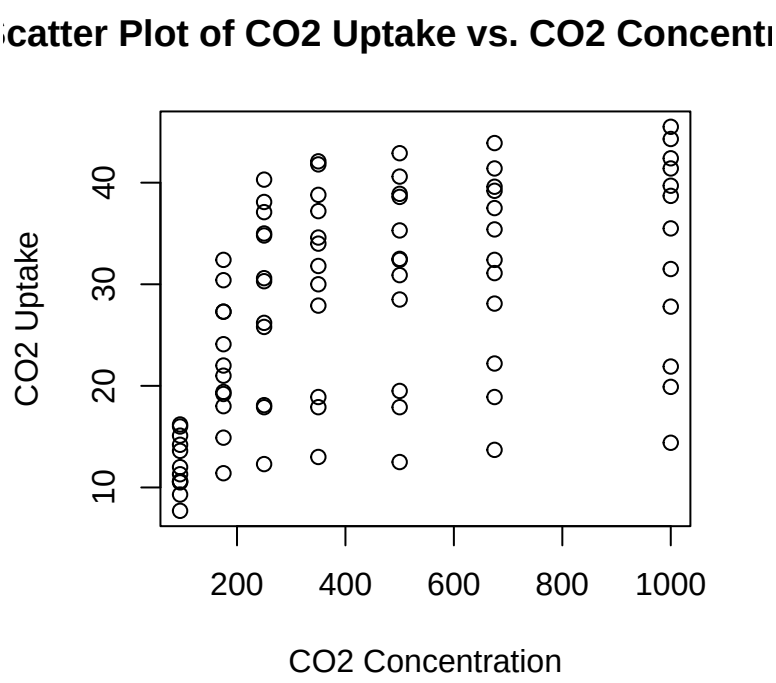
When this code is executed, it will create a box plot of the CO2 uptake variable, allowing you to visualize the distribution of CO2 uptake measurements. The main title will be "Box Plot of CO2 Uptake," and the x-axis will be labeled "CO2 Uptake," making the plot informative and easy to interpret. Box plots are useful for visualizing the spread and central tendency of a dataset, as well as identifying potential outliers.

4.3.3 Bivariate Visualization

Bivariate visualization involves exploring relationships between two variables. Here are some techniques and their interpretations:

- **Scatter Plots:** Scatter plots display the relationship between two continuous variables. They can reveal patterns, correlations, and potential outliers.

```
# Create a scatter plot of CO2 uptake vs. CO2 concentration
plot(CO2$conc,
     CO2$uptake,
     main = "Scatter Plot of CO2 Uptake vs. CO2 Concentration",
     xlab = "CO2 Concentration",
     ylab = "CO2 Uptake")
```



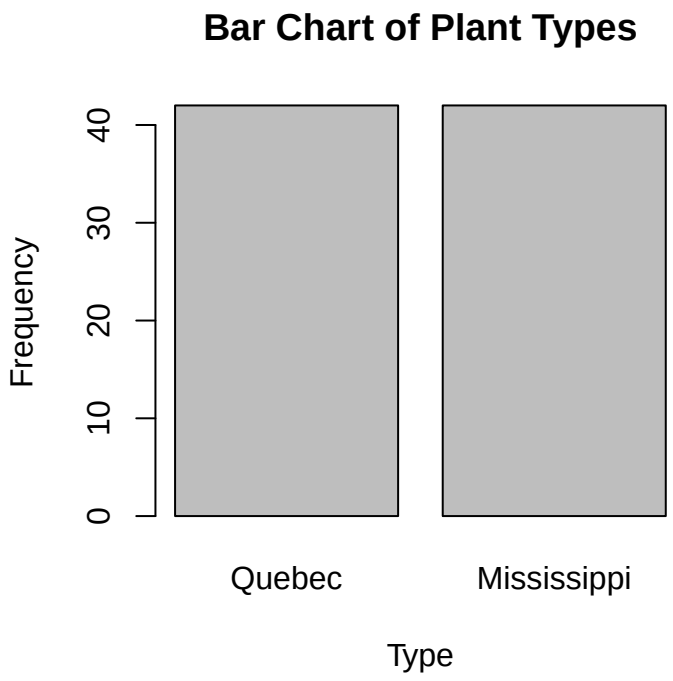
R Code Explanation

- `plot(CO2$conc, CO2$uptake, ...)` creates a scatter plot with CO2 concentration (x-axis) on one axis and CO2 uptake (y-axis) on the other. Specifically:
  - `CO2$conc` specifies the variable to be plotted on the x-axis, which is CO2 concentration.
  - `CO2$uptake` specifies the variable to be plotted on the y-axis, which is CO2 uptake.
- `main = "Scatter Plot of CO2 Uptake vs. CO2 Concentration"` specifies the main title of the scatter plot. In this case, it's titled "Scatter Plot of CO2 Uptake vs. CO2 Concentration."
- `xlab = "CO2 Concentration"` labels the x-axis of the scatter plot, providing a description of what the x-axis represents. Here, it indicates that the x-axis represents CO2 concentration.
- `ylab = "CO2 Uptake"` labels the y-axis of the scatter plot, providing a description of what the y-axis represents. Here, it indicates that the y-axis represents CO2 uptake.

When this code is executed, it will create a scatter plot that allows you to visualize the relationship between CO2 concentration and CO2 uptake. The main title will be "Scatter Plot of CO2 Uptake vs. CO2 Concentration," and both the x-axis and y-axis will be appropriately labeled, making the plot informative and easy to interpret. Scatter plots are useful for identifying patterns and relationships between two continuous variables.

- **Bar Charts:** Bar charts are useful for visualizing the distribution of a categorical variable or the relationship between a categorical variable and a continuous variable.

```
# Create a bar chart of Plant types
barplot(table(CO2$Type),
        main = "Bar Chart of Plant Types",
        xlab = "Type",
        ylab = "Frequency")
```



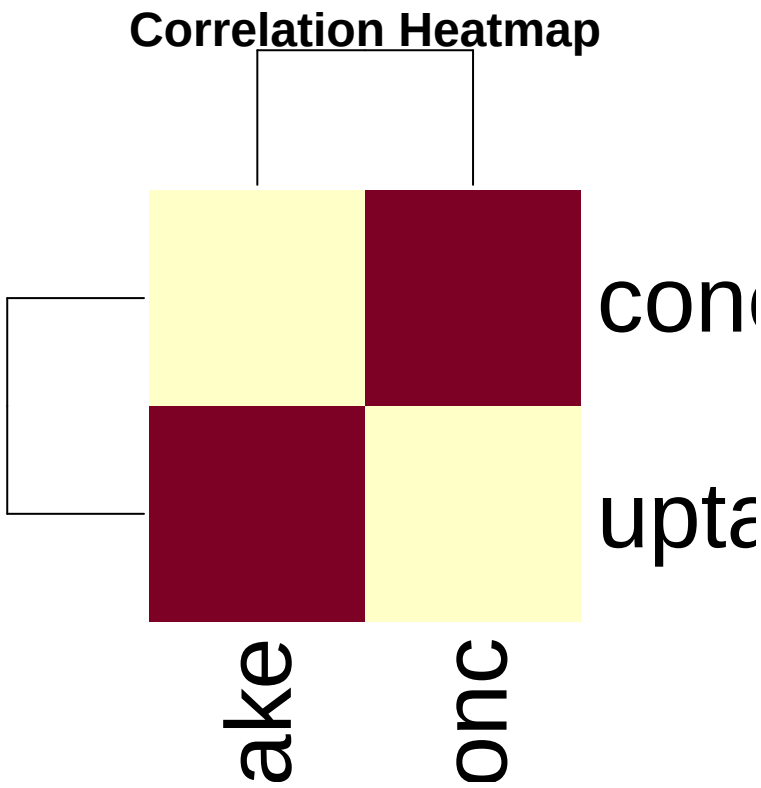
R Code Explanation

- **`barplot(table(CO2$Type), ...)`** creates a bar chart of plant types based on the **`CO2$Type`** variable. Specifically:
  - **`table(CO2$Type)`** computes a frequency table of the plant types in the CO2 dataset. It counts how many times each unique plant type appears.
  - **`barplot(...)`** takes the frequency table as input and generates a bar chart from it.
- **`main = "Bar Chart of Plant Types"`** specifies the main title of the bar chart. In this case, it's titled "Bar Chart of Plant Types."
- **`xlab = "Type"`** labels the x-axis of the bar chart, providing a description of what the x-axis represents. Here, it indicates that the x-axis represents different plant types.
- **`ylab = "Frequency"`** labels the y-axis of the bar chart, providing a description of what the y-axis represents. Here, it indicates that the y-axis represents the frequency (count) of each plant type.

When this code is executed, it will create a bar chart that visually displays the frequency of each plant type in the CO2 dataset. The main title will be "Bar Chart of Plant Types," and both the x-axis and y-axis will be appropriately labeled, making the chart informative and easy to interpret. Bar charts are useful for comparing categories or groups by showing the frequency or count of each category.

- **Correlation Plots:** Correlation plots, such as heatmaps or scatterplot matrices, show the pairwise relationships between multiple continuous variables. They help identify correlations and dependencies among variables.

```r
# Compute the correlation matrix
cor_matrix <- cor(CO2[, c("uptake", "conc")])

# Create a correlation heatmap
heatmap(cor_matrix, main = "Correlation Heatmap")
```



R Code Explanation

- **`heatmap(...)`** generates a heatmap (a graphical representation of data in which values are depicted as colors) based on the correlation matrix provided as input. Here's how it's used in this code:
  - **`cor_matrix`** is the correlation matrix we computed earlier, containing the correlation coefficients between "uptake" and "conc."
  - **`main = "Correlation Heatmap"`** specifies the main title of the heatmap. In this case, it's titled "Correlation Heatmap."

When this code is executed, it will create a heatmap that visually represents the correlations between the "uptake" and "conc" variables from the CO2 dataset. The heatmap's colors and intensity will indicate the strength and direction of the correlations between these variables. It provides a quick and informative way to assess the relationships between variables in a dataset.

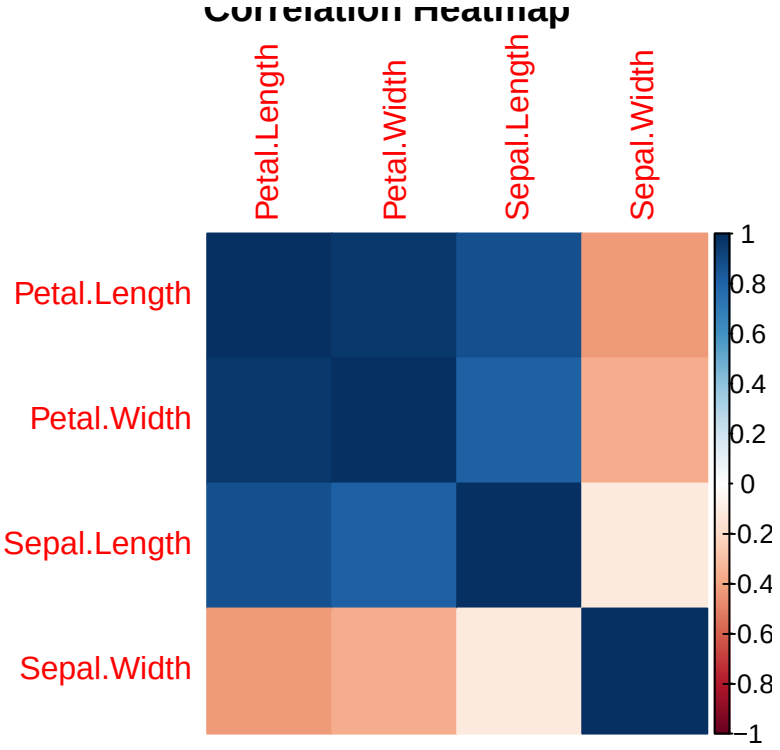
4.3.4 Multivariate Visualization

Multivariate visualization techniques allow researchers to analyze interactions among multiple variables. Here's an example:

- **Heatmaps:** Heatmaps are effective for visualizing relationships between multiple variables, especially in ecological studies where datasets are multidimensional. They provide a comprehensive view of correlations or patterns.

```
# Load the corrplot package for enhanced heatmap visualization
library(corrplot)

# Load data
data("iris")

# Compute the correlation matrix
```

```r
cor_matrix2 <-
  cor(iris[, c("Petal.Length", "Petal.Width", "Sepal.Length", "Sepal.Width")])

# Create a correlation heatmap for multiple variables
corrplot(cor_matrix2, method = "color", title = "Correlation Heatmap")
```



R Code Explanation

- **`library(corrplot)`** loads the **`corrplot`** package, which provides enhanced capabilities for visualizing correlation matrices.
- **`data("iris")`** loads the famous "iris" dataset, which contains measurements of sepal and petal length and width for different species of iris flowers. This dataset is used for the correlation analysis.
- **`cor(...)`** calculates the correlation coefficients between variables. In this case, it calculates the correlation coefficients between four variables: "Petal.Length," "Petal.Width," "Sepal.Length," and "Sepal.Width." The resulting **`cor_matrix2`** is a 4x4 correlation matrix, where each entry represents the correlation between two variables.
- **`corrplot(...)`** from the **`corrplot`** package generates a correlation heatmap based on the correlation matrix provided as input (**`cor_matrix2`**). Here's how it's used in this code:
  - **`method = "color"`** specifies the method for displaying the correlations using colors. This method colors the cells of the heatmap based on the correlation values.
  - **`title = "Correlation Heatmap"`** sets the title of the heatmap to "Correlation Heatmap."

When this code is executed, it will create an enhanced correlation heatmap that visually represents the correlations between the specified variables (Petal.Length, Petal.Width, Sepal.Length, Sepal.Width) in the iris dataset. The colors and intensity in the heatmap indicate the strength and direction of the correlations between these variables. This visualization is helpful for understanding the relationships between multiple variables in a dataset.

In ecological research, these visualization techniques aid in understanding the data, generating hypotheses, and communicating findings effectively. They are essential tools for any ecologist seeking to explore and analyze ecological datasets.

4.4 Summary Statistics

4.4.1 Summary Statistics Overview

Summary statistics are essential in ecological research for several reasons:

1. **Data Summarization:** They provide a concise summary of large datasets, making it easier to grasp the dataset's characteristics.
2. **Data Exploration:** Summary statistics help ecologists understand the distribution, central tendency, and variability of ecological measurements.
3. **Comparison:** They enable researchers to compare different datasets or subsets within a dataset.

4.4.2 Measures of Central Tendency

These measures describe the center or average of a dataset:

1. **Mean:** The mean, also known as the average, is calculated by summing all values in a dataset and dividing by the number of values. It's appropriate for normally distributed data.
2. **Median:** The median is the middle value when data is sorted in ascending order. It's robust to outliers and appropriate for skewed data.
3. **Mode:** The mode is the most frequent value(s) in a dataset. It's suitable for categorical or discrete data.

4.4.3 Measures of Variability

Measures of variability quantify the spread or dispersion of data:

1. **Variance:** Variance measures how much each data point deviates from the mean. It's calculated as the average of the squared differences between each data point and the mean.
2. **Standard Deviation:** The standard deviation is the square root of the variance. It provides a more interpretable measure of dispersion in the same units as the data.

4.4.4 Quantiles and Percentiles

1. **Quantiles:** Quantiles divide a dataset into equal portions. For instance, the median is the 50th percentile, dividing data into two equal halves. Quantiles can reveal data distribution and detect outliers.
2. **Percentiles:** Percentiles are specific quantiles that indicate the relative standing of a value within a dataset. For example, the 25th percentile is the value below which 25% of the data falls.

The following code chunk below provides summary of all important descriptive stats outlined above (Codes shown only - not executed).

```
# Load necessary packages
library(skimr)      # For data summary statistics
library(dplyr)      # For data manipulation
library(flextable)  # For creating formatted tables
library(elucidate)  # For quick data visualization

# Define formatting properties of tables using flextable
flextable::set_flextable_defaults(
```

```
  font.size = 12, # Set font size
  theme_fun = "theme_apa", # Apply APA-style theme
  font.family = "serif", # Set font family
  digits = 3 # Number of decimal places
)

# Display data properties for the CO2 dataset using skimr and create a flextable
CO2 %>% skimr::skim_without_charts() %>% flextable::flextable()

# For only one variable (e.g., 'conc' column) in the CO2 dataset
CO2$conc %>% skimr::skim_without_charts() %>% flextable::flextable()

# Group data by a factor variable ('Type' column) and create flextables for
# selected columns
CO2[, c(2, 4:5)] %>%
  dplyr::group_by(Type) %>% # Group data by 'Type' column
  skimr::skim_without_charts() %>% # Calculate summary statistics
  flextable::qflextable() # Create a flextable with APA-style formatting

# Quick plot/visual display of data for variables 'conc' and 'uptake'
CO2 %>% elucidate::plot_var_all(cols = c("conc", "uptake"))
```

R Code Explanation

1. The code loads the necessary packages (**skimr**, **dplyr**), including **elucidate** for quick data visualization.
2. Formatting properties for tables using **flextable** are defined. These properties include font size, theme, font family, number of decimal places, and font color, ensuring that all subsequent flextables adhere to these formatting settings.
3. The **CO2** dataset is summarized using **skimr::skim_without_charts()**, which provides summary statistics without charts. The result is then formatted into a table using **flextable::flextable()**.
4. To summarize a single variable (in this case, 'conc'), the same process is repeated, but only the 'conc' column is selected for summary statistics.
5. The script groups the data by the 'Type' column (renamed from 'Plant') and calculates summary statistics for the selected columns (columns 1, 4, and 5) within each group. The result is formatted into a table using **flextable::qflextable()**. This demonstrates how to perform group-wise summaries and format the results into an APA-style table.
6. Finally, a quick visual display of the data for the 'conc' and 'uptake' variables is generated using **elucidate::plot_var_all()**, providing a convenient way to visualize these variables. Note that dashed lines on the density plots are theoretical normal distribution curves.

These codes enhance the data summarization and visualization capabilities for the CO2 dataset, making it easier to analyze and present the data.

5 Chapter 4: Statistical Tests

5.1 Introduction

Chapter 4 delves into the core of statistical data analysis for ecological research. You will gain a comprehensive understanding of statistical hypothesis testing and learn to perform a variety of tests commonly used in ecology. By the end of this chapter, you will have:

- Acquired knowledge of fundamental statistical concepts.
- Explored various statistical tests relevant to ecological research.
- Gained hands-on experience in conducting these tests using R and Jamovi.

5.2 Hypothesis Testing Fundamentals

In ecological research, hypothesis testing plays a crucial role in making data-driven decisions and drawing valid conclusions from data. It allows researchers to systematically evaluate whether there is enough evidence to support a particular claim or hypothesis about ecological phenomena. Hypothesis testing helps distinguish between random variation in data and meaningful patterns or effects.

5.2.1 Null hypothesis (H0) and alternative hypothesis (Ha)

- **Null Hypothesis (H0):** The null hypothesis is a statement that suggests there is no significant effect, relationship, or difference between groups or variables in the population being studied. It serves as a default assumption or starting point for hypothesis testing.
- **Alternative Hypothesis (Ha):** The alternative hypothesis is a statement that contradicts the null hypothesis. It suggests that there is a significant effect, relationship, or difference in the population. Researchers typically design experiments or analyses with the hope of finding evidence to support the alternative hypothesis.

5.2.2 Significance Level (Alpha)

- The significance level, denoted as alpha ( ), is a critical parameter in hypothesis testing. It represents the threshold for determining statistical significance. In other words, it sets the standard for how strong the evidence must be to reject the null hypothesis.
- Commonly used alpha values include 0.05 (5%) and 0.01 (1%). A significance level of 0.05 means that the researcher is willing to accept a 5% chance of making a Type I error (rejecting the null hypothesis when it's true). A lower alpha (e.g., 0.01) requires stronger evidence to reject the null hypothesis but increases the risk of Type II errors (failing to reject the null hypothesis when it's false). The choice of alpha depends on the research question, the consequences of Type I and Type II errors, and prevailing scientific standards.

Understanding these fundamental concepts of hypothesis testing is essential for conducting meaningful ecological research and making valid inferences from data. Researchers design experiments, collect data, and perform statistical tests with the aim of either supporting the alternative hypothesis or failing to reject the null hypothesis, based on the evidence provided by the data. The significance level alpha serves as a critical tool for controlling the balance between making Type I and Type II errors, ensuring that research findings are robust and reliable.

5.3 Parametric vs. Non-Parametric Tests

5.3.1 Distinguishing Parametric and Non-Parametric Tests

- **Parametric Tests:** Parametric tests are statistical tests that make specific assumptions about the population distribution, such as normality and homogeneity of variances. These tests rely on the estimation of population parameters (e.g., means and variances) and often provide more statistical power when the assumptions are met.
- **Non-Parametric Tests:** Non-parametric tests are statistical tests that do not rely on assumptions about the population distribution. They are distribution-free tests that use ranking and order statistics to make inferences about the population. Non-parametric tests are robust to violations of distributional assumptions.

5.3.2 When to use each type of test based on data characteristics

- **Parametric tests** are appropriate when data meet the assumptions of normality and homogeneity of variances. They are more powerful than non-parametric tests when these assumptions are met.
- **Non-parametric tests** are suitable when data do not meet the assumptions of normality and homogeneity of variances or when dealing with ordinal or categorical data. They are also preferred when researchers want to make minimal distributional assumptions.

5.3.3 How to use parametric tests like t-tests, ANOVA, and linear regression

- **T-Tests:** T-tests are used to compare the means of two groups or conditions. For example, in ecology, a t-test can be used to compare the mean tree height between two different treatment groups.
- **ANOVA (Analysis of Variance):** ANOVA is used to compare the means of three or more groups or conditions. In ecology, it can be applied to compare the mean biomass across multiple vegetation types.
- **Linear Regression:** Linear regression is used to model the relationship between a dependent variable and one or more independent variables. Ecological examples include modeling the relationship between temperature and species diversity.

5.3.4 How to use non-parametric tests like Mann-Whitney U test and Kruskal-Wallis test

- **Mann-Whitney U Test:** The Mann-Whitney U test compares the distributions of two independent groups when the assumptions for a t-test are not met. For example, it can be used to compare the abundance of a species in two different habitats.
- **Kruskal-Wallis Test:** The Kruskal-Wallis test extends the Mann-Whitney U test to three or more independent groups. It is used when comparing medians across multiple groups, such as testing the effect of different soil types on plant growth.

Understanding when to use parametric and non-parametric tests is crucial for ecological research. Parametric tests are powerful when assumptions are met, while non-parametric tests provide robust alternatives when assumptions are violated or when dealing with non-normally distributed data. The choice between these two types of tests should be based on the nature of the data and the specific research question at hand.

5.4 Performing Statistical Tests in R and Jamovi

5.4.1 Overview of R and Jamovi for Statistical Tests

- **R:** R is a versatile and powerful statistical programming language. It offers a wide range of packages and libraries specifically tailored for ecological data analysis. R provides the flexibility to perform basic to advanced statistical tests, hypothesis testing, regression analysis, multivariate analysis, spatial analysis, and more. Its extensive graphical capabilities also enable the creation of informative data visualizations, which are crucial for ecological research.
- **Jamovi:** Jamovi is a user-friendly statistical software that simplifies data analysis. It is particularly suitable for beginners in ecological research due to its intuitive graphical interface and point-and-click functionality. Jamovi seamlessly integrates with R, allowing users to transition from simple analyses in Jamovi to more complex statistical tests in R as they gain proficiency. Jamovi's ecosystem includes a range of statistical tests commonly used in ecological research.

5.4.2 Benefits of using these tools for data analysis

1. **Versatility:** Both R and Jamovi offer a wide array of statistical tests commonly used in ecological research. Researchers can perform t-tests, ANOVA, regression analysis, non-parametric tests, and advanced multivariate analyses using these tools.

2. **Flexibility:** R, in particular, provides unlimited flexibility. Users can customize analyses, create bespoke statistical models, and develop complex ecological workflows to suit their research needs.
3. **Visualization:** Both R and Jamovi excel in data visualization. Researchers can create publication-quality graphs, plots, and charts to present their ecological findings effectively.
4. **Integration:** Jamovi's integration with R is a valuable feature. Users can start with simple analyses in Jamovi and gradually transition to more advanced analyses in R as their skills grow.
5. **Community Support:** R benefits from a large and active user community. Researchers can find extensive resources, tutorials, and forums to seek help and share knowledge. Jamovi also has a growing community and offers user support.
6. **Open Source:** Both R and Jamovi are open-source software, making them accessible and cost-effective tools for ecological research.
7. **Reproducibility:** Using R or Jamovi for data analysis enhances the reproducibility of ecological research. Researchers can document their analyses, share code, and ensure transparency in their work.
8. **Teaching and Learning:** Jamovi's user-friendly interface makes it an excellent tool for teaching ecological data analysis to students and beginners. R, with its extensive capabilities, serves as a powerful teaching tool for advanced statistical concepts.

In summary, R and Jamovi offer a robust and accessible environment for ecological data analysis. They empower researchers, from beginners to experts, to conduct a wide range of statistical tests, explore data visually, and enhance the rigor and reproducibility of ecological research.

5.4.3 Step-by-Step Test Procedures

5.4.3.1 R: Performing t-Test, Mann-Whitney U test, Anova and Kruskal-Wallis tests

1. Set Up Your Environment and perform t-test
2. First do normality test to decide whether to use a parametric or non-parametric test.

- **Normality test**

```
# Load necessary libraries (if not already loaded)
library(tidyverse)  # Loads the tidyverse package for data manipulation and
# visualization.
library(janitor)  # Loads the janitor package for data cleaning.
library(report) #  Load the report package, for generating summary reports.

# Load the InsectSprays dataset
data("InsectSprays")

# Perform the Shapiro-Wilk normality test on the 'count' variable
shapiro_test_result <- stats::shapiro.test(InsectSprays$count)

# View the normality test result
print(shapiro_test_result)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  InsectSprays$count
## W = 0.9216, p-value = 0.0002525
```

```
# create a function to interpret the result
if (shapiro_test_result$p.value < 0.05) {
```

```r
  cat(
    "The data is not normally distributed (p < 0.05). Select a non-parametric counterpart test."
  )
} else {
  cat("The data is normally distributed (p >= 0.05). Proceed with parametric test.")
}
```

```
## The data is not normally distributed (p < 0.05). Select a non-parametric counterpart test.
```

R Code Explanation

This code loads necessary libraries, loads the **InsectSprays** dataset, performs the Shapiro-Wilk normality test on the **count** variable, and prints the test result. The interpretation is provided using a function based on the p-value, indicating whether the data is normally distributed or not, and suggesting whether to proceed with a parametric or non-parametric test accordingly.

- **Mann-Whitney U Test (non-parametric test)**

```r
# Load necessary libraries (if not already loaded)
library(tidyverse)  # Loads the tidyverse package for data manipulation and
# visualization.
library(janitor) # Loads the janitor package for data cleaning.
library(report)  #  Load the report package, for generating summary reports.

# Use the InsectSprays data to test for the effectiveness of Insect Sprays.The
# dataset contains counts of insects in agricultural experimental units treated # with different insect
data("InsectSprays")

# Subset the dataset to use only two types of insecticides (spray A and spray B
# ). The dplyr::filter() function is used to filter rows where spray is either
# "A" or "B".
insectides <- InsectSprays %>% dplyr::filter(spray %in% c("A", "B"))
# Perform the Mann-Whitney U Test on the 'insectides' dataset
mann_whitney_result <-
  stats::wilcox.test(count ~ spray, data = insectides, alternative = "two.sided")

# View the Mann-Whitney U Test result
print(mann_whitney_result)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  count by spray
## W = 62, p-value = 0.5812
## alternative hypothesis: true location shift is not equal to 0
```

```r
# create a function to interpret the result
if (mann_whitney_result$p.value < 0.05) {
  cat("There is a significant difference between spray 'A' and spray 'B' (p < 0.05).")
} else {
  cat("There is no significant difference between spray 'A' and spray 'B' (p >= 0.05).")
}
```

```
## There is no significant difference between spray 'A' and spray 'B' (p >= 0.05).
```

R Code Explanation

1. **Loading Necessary Libraries:** The code begins by loading several R packages using the **library()**

function. Each package serves a specific purpose in the analysis:

- **`tidyverse`**: This package is loaded for data manipulation and visualization. The "tidyverse" collection includes a set of packages that make data analysis and visualization more efficient and consistent.
- **`janitor`**: The "janitor" package is loaded for data cleaning tasks. It provides functions for cleaning and tidying data, which is an essential step in data analysis.
- **`report`**: The "report" package is loaded to facilitate the generation of summary reports based on statistical analysis results. It automates the report creation process.

2. **Loading the InsectSprays Dataset:** The code loads the "InsectSprays" dataset using the **`data()`** function. This dataset contains information about counts of insects in agricultural experimental units treated with different insecticides. It's a common dataset used for statistical testing and analysis.
3. **Subsetting the Dataset:** The **`insectides`** variable is created by subsetting the original dataset. It selects only the rows where the "spray" column has values "A" or "B." This subset of data will be used for the Mann-Whitney U Test, focusing on comparing the effectiveness of these two insecticide sprays.
4. **Performing the Mann-Whitney U Test:** The Mann-Whitney U Test is conducted using the **`wilcox.test()`** function. It assesses whether there is a significant difference in the distribution of insect counts between spray "A" and spray "B." The **`count ~ spray`** formula specifies that the "count" variable is being compared across the different "spray" groups.
5. **Viewing the Test Result:** The result of the Mann-Whitney U Test is printed to the console using the **`print()`** function. This result includes statistics such as the U statistic and the p-value, which are crucial for interpreting the test outcome.
6. **Interpreting the Result:** A conditional statement is used to interpret the test result. If the p-value is less than 0.05 (typically chosen as the significance level), it suggests that there is a significant difference between spray "A" and spray "B" regarding their effectiveness in controlling insects. If the p-value is greater than or equal to 0.05, it suggests that there is no significant difference between the two sprays.

In summary, this code demonstrates how to perform and interpret a Mann-Whitney U Test using R. It focuses on comparing the effectiveness of two insecticide sprays ("A" and "B") in controlling insect populations based on count data. The "report" package is used to facilitate report generation, which can be helpful for documenting and communicating the results of statistical tests.

- **t-Test (parametric test; assuming normality test above is *p >= 0.05*)**

```
# Load necessary libraries (if not already loaded)
library(tidyverse) # Loads the tidyverse package for data manipulation and
# visualization.
library(janitor) # Loads the janitor package for data cleaning.
library(report) #  Load the report package, for generating summary reports.

# Use the InsectSprays data to test for the effectiveness of Insect Sprays.The
# dataset contains counts of insects in agricultural experimental units treated
# with different insecticides.
data("InsectSprays")

#Subset the dataset to use only two types of insecticides (spray A and spray B). # The dplyr::filter()
# or "B".
insectides <- InsectSprays %>% dplyr::filter(spray %in% c("A", "B"))

# Assuming you want to compare two groups (spray A and spray B).
# Perform a t-Test for independent samples using the stats::t.test() function.
# The formula count ~ spray specifies that you want to compare the 'count'
```

```r
# variable
# across different 'spray' groups. 'data = insectides' specifies the dataset to
# use.
t_test_result <- stats::t.test(count ~ spray, data = insectides)

# View the t-test result using the print() function.
t_test_result %>% print()
```

```
##
##  Welch Two Sample t-test
##
## data:  count by spray
## t = -0.45352, df = 21.784, p-value = 0.6547
## alternative hypothesis: true difference in means between group A and group B is not equal to 0
## 95 percent confidence interval:
##  -4.646182  2.979515
## sample estimates:
## mean in group A mean in group B
##        14.50000        15.33333
```

```r
# autogenerate a report. Ingnore the warning.
t_test_result %>% report::report()
```

```
## Effect sizes were labelled following Cohen's (1988) recommendations.
##
## The Welch Two Sample t-test testing the difference of count by spray (mean in
## group A = 14.50, mean in group B = 15.33) suggests that the effect is negative,
## statistically not significant, and very small (difference = -0.83, 95% CI
## [-4.65, 2.98], t(21.78) = -0.45, p = 0.655; Cohen's d = -0.19, 95% CI [-1.03,
## 0.65])
```

R Code Explanation

1. Libraries are loaded to make the necessary functions available for data manipulation (**tidyverse**) and cleaning (**janitor**).
2. The **InsectSprays** dataset is loaded, which contains information about the effectiveness of various insect sprays in controlling insect populations.
3. The dataset is subsetted to include only two types of insecticides, "A" and "B," using the **dplyr::filter()** function.
4. The t-test is performed using **stats::t.test()**. It compares the '**count**' of insects between the two groups defined by '**spray**' (A and B).
5. The t-test result is stored in the variable '**t_test_result**.'
6. The t-test result is printed to the console using the **print()** function.
7. Finally, the script generates an automatic report using the **report::report()** function, providing a summary of the t-test results.

This script helps you assess whether there is a significant difference in the effectiveness of insect sprays A and B in controlling insect populations. The t-test compares the means of the 'count' variable between the two groups and provides information about the statistical significance of any observed differences.

Check the p-value in the t-test result. If $p < 0.05$ (assuming a significance level of 0.05), you can reject the null hypothesis (H0) and conclude that there is a significant difference between the groups.

Below are other tests conducted in R.

One-Way ANOVA (parametric)

```
# Load necessary libraries (if not already loaded)
library(tidyverse)

# Load the InsectSprays dataset
data("InsectSprays")

# Subset the dataset to use only two types of insecticides (spray A and spray B)
insecticides <-
  InsectSprays %>% dplyr::filter(spray %in% c("A", "B"))

# Perform a one-way ANOVA
anova_result <- stats::aov(count ~ spray, data = insecticides)

# View the ANOVA result
summary(anova_result)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## spray        1    4.2   4.167   0.206  0.655
## Residuals   22  445.7  20.258
```

Two-Way ANOVA (parametric)

```
# Perform a two-way ANOVA
two_way_anova_result <-
  stats::aov(count ~ spray, data = InsectSprays)

# View the two-way ANOVA result
summary(two_way_anova_result)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## spray        5   2669   533.8    34.7 <2e-16 ***
## Residuals   66   1015    15.4
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Kruskal-Wallis Test (non-parametric)

```
# Load necessary libraries (if not already loaded)
library(tidyverse)

# Perform a Kruskal-Wallis test
kruskal_wallis_result <-
  stats::kruskal.test(count ~ spray, data = InsectSprays)

# View the Kruskal-Wallis test result
kruskal_wallis_result
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  count by spray
## Kruskal-Wallis chi-squared = 54.691, df = 5, p-value = 1.511e-10
```

Post-Hoc Pairwise Comparison Test

To perform a post hoc test with Bonferroni correction using the Agricolae R package on the InsectSprays

dataset, you can follow these steps:

1. Install and load the Agricolae package (if not already installed).
2. Perform a Kruskal-Wallis test on the dataset.
3. Conduct a post hoc test with Bonferroni correction to compare groups.

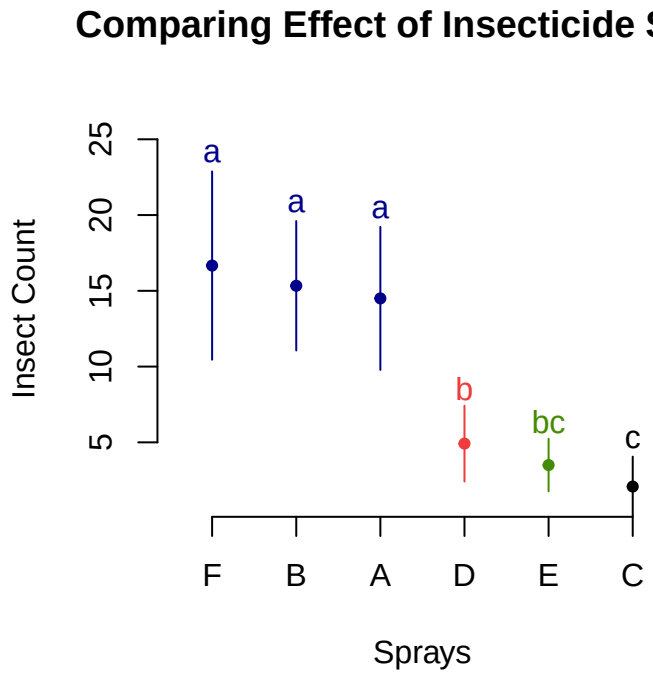
Here's the R code to achieve this:

```
# Install and load the required packages
# We use 'pacman' to install and load packages in a single step.
# If not already installed, 'pacman' will install the packages.
# 'agricolae' for conducting Kruskal-Wallis tests and post hoc analysis.
# 'install = TRUE' specifies to install the packages if not present.
# 'update = FALSE' prevents updating existing packages.
pacman::p_load(agricolae, install = TRUE, update = FALSE)

# Load the InsectSprays dataset
# The 'data()' function loads the InsectSprays dataset, which contains insect
# count data.
data("InsectSprays")

# Perform Kruskal-Wallis test without grouping
# 'agricolae::kruskal()' conducts the Kruskal-Wallis test.
# We specify the 'count' variable as the dependent variable and 'spray' as the
# independent variable.
# 'group = FALSE' indicates that we don't want to group the results.
# 'p.adj = "bon"' specifies Bonferroni correction for post hoc tests.
comparison_stats <- with(InsectSprays,
                         agricolae::kruskal(count, spray,
                                            group = FALSE,
                                            p.adj = "bon"))

# Display selected statistical results
# We extract specific results from the 'comparison_stats' object.
# In this case, we're interested in columns 1, 2, and 4.
comparison_stats[c(1:2, 4)]
```

```
## $statistics
##      Chisq Df      p.chisq  t.value      MSD
##   54.69134  5 1.510845e-10 3.045792 12.91015
##
## $parameters
##             test  p.ajusted name.t ntr alpha
##   Kruskal-Wallis bonferroni  spray   6  0.05
##
## $comparison
##        Difference pvalue Signif.        LCL        UCL
## A - B  -2.6666667 1.0000         -15.576816  10.243483
## A - C  40.7083333 0.0000     ***  27.798184  53.618483
## A - D  26.5833333 0.0000     ***  13.673184  39.493483
## A - E  32.8333333 0.0000     ***  19.923184  45.743483
## A - F  -3.4583333 1.0000         -16.368483   9.451816
## B - C  43.3750000 0.0000     ***  30.464851  56.285149
## B - D  29.2500000 0.0000     ***  16.339851  42.160149
## B - E  35.5000000 0.0000     ***  22.589851  48.410149
```

```
## B - F  -0.7916667 1.0000         -13.701816  12.118483
## C - D -14.1250000 0.0212       * -27.035149  -1.214851
## C - E  -7.8750000 1.0000         -20.785149   5.035149
## C - F -44.1666667 0.0000     *** -57.076816 -31.256517
## D - E   6.2500000 1.0000          -6.660149  19.160149
## D - F -30.0416667 0.0000     *** -42.951816 -17.131517
## E - F -36.2916667 0.0000     *** -49.201816 -23.381517
```

```r
# Perform Kruskal-Wallis test with grouping
# This time, we set 'group = TRUE' to group the results for plotting.
comparison_grp <- with(InsectSprays,
                       agricolae::kruskal(count, spray,
                                          group = TRUE,
                                          p.adj = "bon"))

# Plot group comparison
# 'agricolae::plot.group()' is used to create a bar chart of group comparisons.
# 'variation = "SE"' specifies to display standard errors.
# 'decreasing = TRUE' orders the bars in descending order of means.
# 'main = "Insecticide Sprays"' sets the chart's title.
agricolae::plot.group(
  comparison_grp,
  variation = "SD",
  decreasing = TRUE,
  main = "Comparing Effect of Insecticide Sprays",
  xlab = "Sprays",
  ylab = "Insect Count"
)
```



R Code Explanation

- The code begins by installing and loading the necessary R packages using the '**pacman**' package manager.
- The '**InsectSprays**' dataset is loaded using the 'data()' function. This dataset contains insect count data.
- Two Kruskal-Wallis tests are performed, one without grouping and one with grouping. The '**agricolae::kruskal()**' function is used for this purpose.
- Bonferroni correction ('**p.adj = "bon"**') is applied to adjust p-values in post hoc tests.
- Selected statistical results are displayed for the first Kruskal-Wallis test.
- A bar chart of group comparisons is created for the second Kruskal-Wallis test using '**agricolae::plot.group()**'. This chart displays standard errors, orders bars by decreasing means, and sets the title.

Overall, this code demonstrates the use of the '**agricolae**' package for Kruskal-Wallis tests and post hoc analysis, along with visualizing group comparisons.

5.4.3.2 Jamovi: Performing t-Test, Mann-Whitney U test, Anova and Kruskal-Wallis tests Step 1: Set Up Your Environment

Open Jamovi and load your ecological dataset. We'll use the same dataset "**InsectSprays**". First in R, save this data as a flat csv file to your data directory. Here the dataset in called "**insecticide**".

```
library(readr)# Load the readr package
library(here)# Load the here package for managing file paths

# Write the InsectSprays dataset to a CSV file
# Specify the dataset to be written (InsectSprays)
# Specify the file path where the CSV file will be saved (./docs/data
# /insecticide.csv)
#Specify that column names should be included in the CSV file (col_names = TRUE)
readr::write_csv(
  InsectSprays, # Dataset to be written
  file = here::here("docs", "data", "insecticide.csv"), # File path and name
  col_names = TRUE, # Include column names in the CSV file
  append = FALSE # Don't append to an existing file, create a new one
)
```

R Code Explanation

- This section of the code is responsible for writing the **InsectSprays** dataset to a CSV file.
- **readr::write_csv()** is used to write the CSV file. Here's what each argument does:
  - **InsectSprays**: The first argument is the dataset you want to write to the CSV file, in this case, it's the **InsectSprays** dataset.
  - **file**: This argument specifies the file path and name for the CSV file. The **here::here()** function is used to create a file path that is relative to the project's root directory. It specifies that the file should be saved in the "docs/data" directory with the name "insecticide.csv."
  - **col_names**: This argument specifies whether column names should be included in the CSV file. Setting it to **TRUE** means that the first row of the CSV file will contain the column names.
- **append = FALSE** ensures that if a file with the same name already exists at the specified path, it won't be appended to. Instead, a new file will be created, potentially overwriting the existing one.

In summary, this code loads the necessary packages, specifies the dataset to be written, defines the file path and name, specifies that column names should be included in the CSV file, and ensures that a new CSV file is created at the specified location.

Step 2: Perform the t-Test, Mann-Whitney, One-Way and Two-Way Anova; and Kruskal-Wallis tests.

1. Open the file in Jamovi.

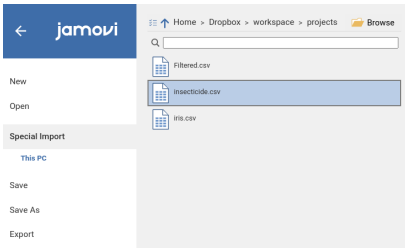


Figure 12: Import insecticide data.

2. Remove unnecessary columns/ variables

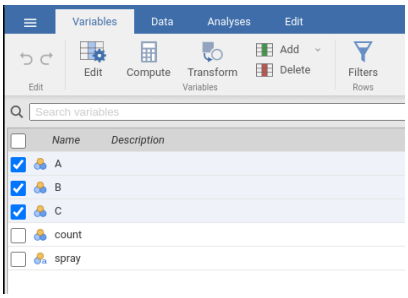


Figure 13: Remove unwanted columns.

3. Append new columns to the original dataset. Note that we would need these variables for the t-test, mann-whitney and one-way anova tests. These tests require the data to have a grouping variable with 2 levels. Other variables with > 2 variables can used in a two-way anova etc.
4. Add two new columns and named as "`count2`" and "`spray2`". change their measure types for count2 as "`continuous`" and spray2 as "`nominal`". Measure/ data types should be the same as the original variables (i.e., "`count`" and "`spray`").
5. Click on "Analyses" in the top menu.
6. Select "T-Tests".
7. Choose "Independent Samples T-Test" if comparing two groups or "One Sample T-Test" if comparing against a fixed value.
8. Drag and drop your ecological variable of interest into the "Test Variables" box.
9. Drag and drop your grouping variable into the "Grouping Variable" box (for independent samples t-test).
10. Customize options and click "OK."
11. Now perform the one-way anova test.
12. Selected appropriate statistics to appear in the results.
13. Do the same for the two-way anova. Use the original variables with > 2 spray levels. You can perform a post-hoc comparison if anova test is significant ($p < 0.05$). Note that our data as previously diagnosed does not conform to the normality assumption hence a non-parametric counterpart test in advisable.

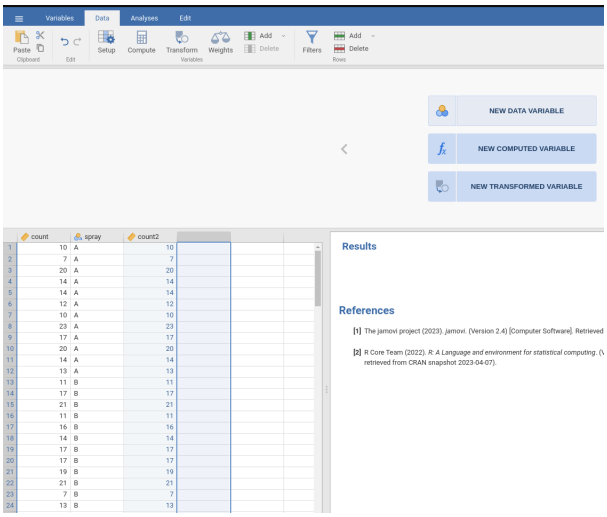


Figure 14: Add columns and change data/measurement types.

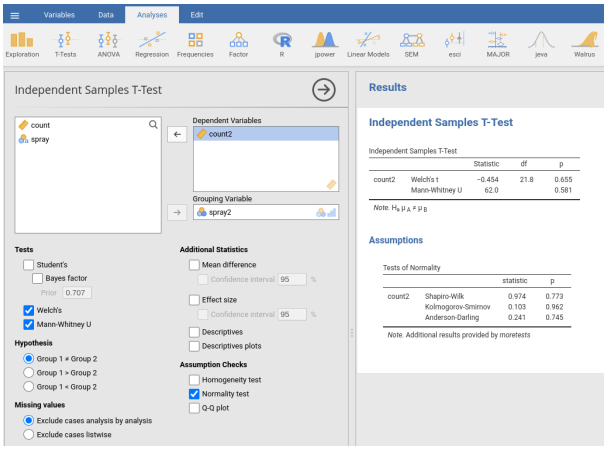


Figure 15: Perform Independent sample t-test.

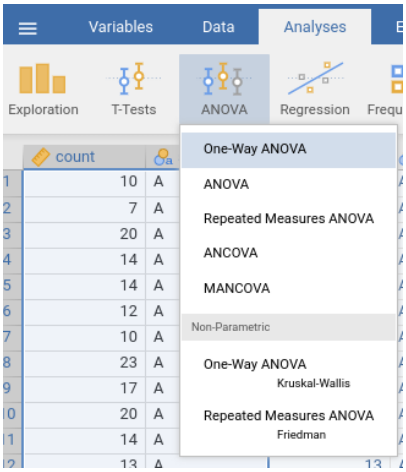


Figure 16: Perform One-way ANOVA (Parametric test).

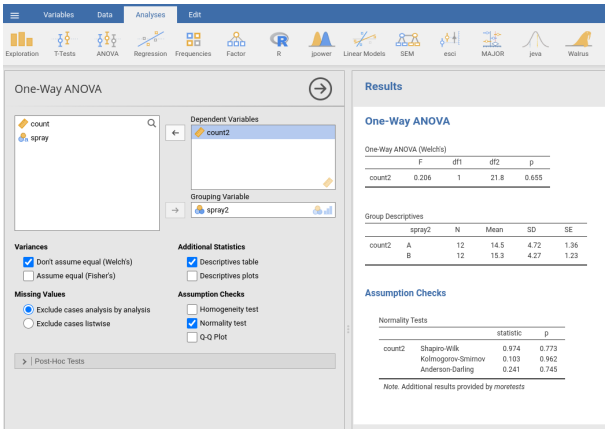


Figure 17: One-way anova test results.

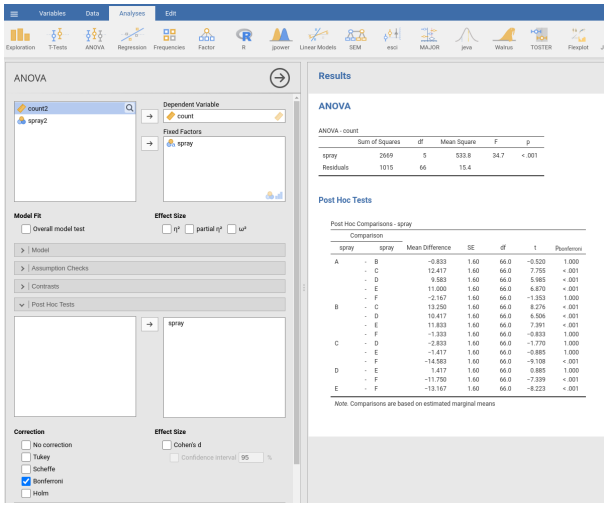


Figure 18: Perform Two-way anova with results.

14. Next, perform Kruskal-Wallis test. Notice normality test is p $<$ 0.05 from the two-way anova test. Notice p $<$ 0.05 so a further pairwise comparison is appropriate.

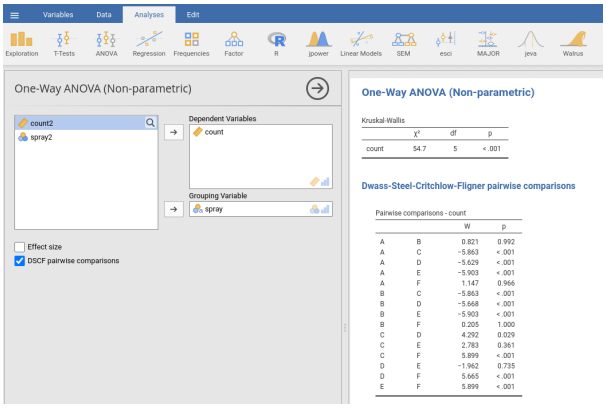


Figure 19: Performing Kruskal_Wallis test.

Step 3: Interpret the Results

- Examine the output for the t-test. Similar to R, look for the p-value. If p $<$ 0.05 (assuming a significance level of 0.05), you can reject the null hypothesis (H0) and conclude that there is a significant difference between the groups.

5.5 Interpreting Test Results

Interpreting the results of statistical tests is a critical aspect of ecological research. Here's a general guideline on how to interpret the results, with a focus on understanding p-values and effect sizes:

1. **Understand the Null Hypothesis (H0) and Alternative Hypothesis (Ha):** Before interpreting the results, it's essential to recall the null hypothesis (H0) and alternative hypothesis (Ha) that you formulated for your test. The null hypothesis typically represents the absence of an effect or relationship, while the alternative hypothesis states the expected outcome.
2. **Examine the Test Statistic:** Most statistical tests generate a test statistic (e.g., t-statistic, F-statistic, chi-square statistic) that quantifies the difference or relationship observed in the data. Larger test statistics often indicate stronger evidence against the null hypothesis.
3. **Check the p-value:** The p-value measures the strength of evidence against the null hypothesis. It represents the probability of obtaining results as extreme as, or more extreme than, the observed results if the null hypothesis is true. A small p-value (typically less than the chosen significance level, alpha) suggests strong evidence against the null hypothesis. Conversely, a large p-value suggests weak evidence against it.
    - **Interpretation of p-values:**
        - p $<$ alpha (e.g., p $<$ 0.05): Strong evidence against H0; you may reject the null hypothesis.
        - p >= alpha: Weak evidence against H0; you fail to reject the null hypothesis.
4. **Consider the Effect Size:** While p-values tell you whether there is a significant difference or relationship, effect sizes provide information about the practical or clinical significance of the result. Effect sizes quantify the strength or magnitude of the observed effect. In ecological research, understanding the biological or ecological significance of an effect is often more important than its statistical significance.
    - **Common effect size measures:** Cohen's d (for t-tests), eta-squared (for ANOVA), correlation coefficients, odds ratios (for logistic regression), etc.

5. **Look at Confidence Intervals:** Confidence intervals provide a range of values within which the true population parameter (e.g., mean, proportion) is likely to fall. They complement p-values and offer additional insights into the precision of your estimates.
6. **Consider Ecological Relevance:** In ecological research, it's crucial to consider whether the results have practical significance. Statistical significance may not always translate to ecological significance. Evaluate the results in the context of your research question and the potential impact on the ecosystem or species you are studying.
7. **Replication and Consistency:** Consider whether the results are consistent with previous research or if they need to be replicated in other studies or under different conditions to strengthen their validity.
8. **Beware of Multiple Comparisons:** If you are conducting multiple tests on the same dataset, be cautious about the issue of multiple comparisons. Adjusting alpha (e.g., Bonferroni correction) can help control the family-wise error rate.
9. **Consult Experts:** If you are unsure about the interpretation of your results, consider seeking guidance from statistical or ecological experts. Collaborating with colleagues who have expertise in the field can enhance the quality of your interpretation.

This is a general overview of performing a t-test, Mann-Whitney u test, anova and Kruskall-Wallis tests in both R and Jamovi. The specific steps may vary depending on your dataset and research question. In ecological research, it's not only important to detect statistically significant results but also to understand their ecological implications. A strong understanding of p-values, effect sizes, and their ecological relevance will contribute to more meaningful and robust ecological research outcomes.

5.6 Conclusion

Chapter 4 provides comprehensive insights into statistical hypothesis testing, which is a fundamental aspect of ecological research.

- You have learned about the significance of hypothesis testing in ecological research, where it helps you make data-driven decisions and draw conclusions about ecological phenomena.
- Key terms like null hypothesis (H0), alternative hypothesis (Ha), and significance level (alpha) have been defined and their roles in hypothesis testing explained.
- You now understand the distinction between parametric and non-parametric tests, as well as when to use each type based on data characteristics.
- Parametric tests such as t-tests, ANOVA, and linear regression have been introduced, along with practical ecological examples for each.
- Non-parametric tests like Mann-Whitney U and Kruskal-Wallis have also been explained, along with ecological examples.
- You've gained practical skills through step-by-step instructions for performing these tests in both R and Jamovi.
- The importance of interpreting results, understanding p-values and effect sizes, and considering ecological relevance has been emphasized.

Overall, Chapter 4 equips you with a strong foundation in statistical hypothesis testing, empowering you to conduct a wide range of tests essential for ecological data analysis and make informed ecological conclusions.

6 Chapter 5: Regression Analysis

6.1 Introduction

Chapter 5 is a deep dive into regression analysis, a powerful tool for modeling ecological relationships. In this chapter, you will learn about two essential types of regression: linear and logistic. By the end of this chapter, you will have:

- A solid understanding of regression analysis and its relevance in ecological research.
- Proficiency in performing linear and logistic regression in both R and Jamovi.
- The ability to interpret regression outputs and draw meaningful ecological insights.

6.2 Understanding Regression Analysis

Regression analysis is a powerful statistical method used to model relationships between variables. It helps us understand how one or more independent variables are related to a dependent variable and how changes in the independent variables impact the dependent variable. In ecological research, regression analysis plays a crucial role in modeling ecological relationships, making predictions, and understanding the impact of environmental factors on biological phenomena.

Key Concepts:

1. **Dependent Variable (Response Variable):** This is the variable we want to predict or explain. In ecological research, it could be the population of a species, the growth rate of a plant, or any other measurable ecological outcome.
2. **Independent Variables (Predictors or Explanatory Variables):** These are the variables that we believe influence or explain changes in the dependent variable. Independent variables can be continuous (e.g., temperature, rainfall) or categorical (e.g., habitat type, presence/absence of a predator).
3. **Regression Equation:** The mathematical formula that represents the relationship between the dependent and independent variables. It allows us to make predictions based on the values of the independent variables.
4. **Types of Regression:** There are different types of regression analysis, including linear regression (for continuous dependent variables), logistic regression (for binary outcomes), and more complex forms like polynomial regression and mixed-effects models.

Applications in Ecological Research

1. **Species-Habitat Relationships:** Ecologists often use regression analysis to model how the abundance or presence of a species is related to habitat variables such as vegetation type, temperature, or elevation.
2. **Climate Change Impact:** Regression models can help assess the impact of climate change variables (e.g., temperature, precipitation) on ecological systems, predicting how ecosystems may respond to future climate scenarios.
3. **Population Dynamics:** Ecological researchers use regression to model population growth, decline, or other changes over time. For example, how does temperature affect the growth rate of a plant species?
4. **Community Ecology:** Regression can be applied to understand the relationships between species richness, diversity, and various environmental factors, shedding light on the mechanisms driving community structure.
5. **Ecosystem Functioning:** Researchers explore how changes in ecological variables (e.g., nutrient availability) impact ecosystem functions (e.g., carbon cycling) using regression modeling.

Regression analysis is a fundamental tool in ecological research that allows researchers to quantify and understand the relationships between ecological variables. It helps in making predictions, testing hypotheses, and gaining insights into the complex dynamics of ecological systems.

6.3 Linear Regression

Definition: Linear regression is a statistical method used to model the relationship between a dependent variable (DV) and one or more independent variables (IVs) by fitting a linear equation to observed data. It assumes a linear relationship between the IVs and the DV, where changes in the IVs lead to a proportional change in the DV.

6.3.1 Applications in Ecological Research

1. **Species Abundance:** Linear regression can be used to understand how environmental factors like temperature, precipitation, or habitat type influence the abundance of a particular species.
2. **Growth Rates:** Ecologists often use linear regression to model the growth rates of plants or animal populations as a function of variables like temperature or nutrient availability.
3. **Biodiversity:** Researchers can examine how habitat diversity, fragmentation, or disturbance affect species richness using linear regression.
4. **Carbon Sequestration:** Linear regression can be applied to study the relationship between forest characteristics (e.g., tree density, age) and carbon sequestration rates in ecosystems.

6.3.2 Performing Linear Regression

6.3.2.1 Performing linear regression in R

1. **Load Required Packages:** Begin by loading necessary packages like `tidyverse` for data manipulation and `lm()` for linear modeling.
2. **Load Data:** Import your ecological dataset into R.
3. **Fit the Model:** Use the `lm()` function to fit a linear regression model. For example: `model <- lm(DV ~ IV1 + IV2, data = dataset)`, where `DV` is the dependent variable, `IV1` and `IV2` are independent variables, and `dataset` is your data.
4. **View Model Summary:** Use `summary(model)` to view the regression model's summary, including coefficients, R-squared, and p-values.
5. **Example:** The `ToothGrowth` dataset in R is a built-in dataset that comes with the base R installation. It provides data on the effect of vitamin C on tooth growth in guinea pigs. This dataset is often used for teaching and learning purposes and is useful for practicing various statistical analyses.

- Here's some information about the `ToothGrowth` dataset:
  - **Description:** The dataset contains observations on the length of guinea pig teeth (tooth growth) under different dosage levels of vitamin C and two delivery methods.
  - **Variables:**
    - `len`: The length of tooth growth (in millimeters).
    - `supp`: The supplement type, either "VC" (vitamin C) or "OJ" (orange juice).
    - `dose`: The dosage of the supplement in milligrams per day, which can be 0.5, 1.0, or 2.0.
  - **Data Structure:** The dataset consists of 60 observations.

You can load and access the `ToothGrowth` dataset in R by simply typing:

```r
# load tooth growth dataset
data("ToothGrowth")
```

Once loaded, you can explore the dataset using functions like **`head(ToothGrowth)`**, **`summary(ToothGrowth)`**, or by creating visualizations and conducting statistical analyses.

This dataset is often used to demonstrate concepts like hypothesis testing, analysis of variance (ANOVA), and regression analysis in introductory statistics and data analysis courses.

```r
# Load necessary R packages using the 'pacman' package
# 'pacman' is a package management tool that makes it easy to load and manage
# multiple packages at once.
# It installs the packages if they are not already installed
# and loads them into the R session.
# The 'tidyverse' package includes a collection of packages for data
# manipulation and visualization. The 'report' package is used for generating
# summary reports.
pacman::p_load(
  tidyverse,# Load the tidyverse package for data manipulation and visualization
  report,# Load the report package for generating summary reports.
  install = TRUE,# Install the packages if not already installed.
  update = FALSE # Do not update already installed packages.
)

# Load the ToothGrowth dataset
# The 'ToothGrowth' dataset is included in R and contains data related to the
# effect of vitamin C on tooth growth in guinea pigs.
data("ToothGrowth")

# Define a linear regression model
# Create a linear regression model using the 'lm' function.
# The model predicts the 'len' (tooth length) variable based on 'supp'
# (supplement
# type) and 'dose' (dose level) predictors.
lm_mod1 <- lm(
  len ~ supp + dose, # Model formula specifying the response variable and
  # predictor variables.
  data = ToothGrowth # Specify the dataset in which to find the variables.
)

# Show the summary of the linear regression model
# The 'summary' function provides detailed information about the linear
# regression model, including coefficients, standard errors, t-values, and p
# -values.
summary(lm_mod1)
```

```
##
## Call:
## lm(formula = len ~ supp + dose, data = ToothGrowth)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.600 -3.700  0.373  2.116  8.800
##
## Coefficients:
```

```
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.2725000  1.2823649   7.231 1.31e-09 ***
## suppVC      -3.7000000  1.0936045  -3.383   0.0013 **
## dose         0.0097636  0.0008768  11.135 6.31e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.236 on 57 degrees of freedom
## Multiple R-squared:  0.7038, Adjusted R-squared:  0.6934
## F-statistic: 67.72 on 2 and 57 DF,  p-value: 8.716e-16
```

```
# Extract and print a model output report
# The 'report' package is used here to generate a summary report
```

R Code Explanation

The codes above load necessary packages and the ToothGrowth dataset, defines a linear regression model, displays a summary of the model using the **`summary`** function, and generates a detailed model output report using the **`report`** package. The report includes information about the model's coefficients and statistics related to its fit.

Result/ Report Interpretation

The provided report output contains detailed information about the linear regression model's performance and parameter estimates. Let's break down the key interpretations:

1. **Model Explanation**
   - A linear regression model was fitted using Ordinary Least Squares (OLS) to predict tooth length (**`len`**) based on two predictors: **`supp`** (supplement type) and **`dose`** (dose level). The formula used for the model is **`len ~ supp + dose`**.
2. **Model Fit**
   - The model is statistically significant and explains a substantial proportion of variance. The key statistics include:
     - R-squared (R^2) value of 0.70: This indicates that approximately 70% of the variability in tooth length is explained by the model.
     - F-statistic ($F(2, 57)$) of 67.72: This tests the overall significance of the model, and the low p-value (< 0.001) suggests that the model is statistically significant.
     - Adjusted R-squared (adj. R^2) of 0.69: This adjusts R^2 for the number of predictors in the model, providing a measure of model fit.
3. **Model Intercept**
   - The model's intercept corresponds to **`supp = OJ`** and **`dose = 0`**.
   - The intercept value is 9.27 with a 95% confidence interval (CI) of [6.70, 11.84].
   - The t-statistic ($t(57)$) is 7.23, and the p-value is < 0.001.
   - This indicates that when **`supp`** is OJ, and **`dose`** is 0, the estimated average tooth length is 9.27.
4. **Parameter Effects**
   - The report provides information about the effects of individual predictors within the model.
   - The effect of **`supp`** (supplement type) with the level **`[VC]`** is statistically significant and negative.
     - The beta coefficient is -3.70 with a 95% CI of [-5.89, -1.51].
     - The t-statistic is -3.38, and the p-value is 0.001.

  - The standardized beta (Std. beta) is -0.48.

- The effect of **dose** is statistically significant and positive.
  - The beta coefficient is 9.76 with a 95% CI of [8.01, 11.52].
  - The t-statistic is 11.14, and the p-value is < 0.001.
  - The standardized beta (Std. beta) is 0.80.

5. **Standardized Parameters**
   - The standardized parameters were obtained by fitting the model on a standardized version of the dataset. Standardized parameters allow for comparing the relative importance of predictors in different units.
6. **Confidence Intervals and p-values**
   - 95% Confidence Intervals (CIs) and p-values were computed using a Wald t-distribution approximation. These values help assess the precision and statistical significance of parameter estimates.

In summary, the report indicates that the linear regression model is a good fit for explaining tooth length (**len**) based on the predictors **supp** and **dose**. The model's parameters, including intercept and effects of predictors, are statistically significant and provide valuable insights into the relationship between these variables and tooth length.

A better approach for linear modelling in R is shown as:

```
# a better stats output summary can be done
lm_mod1 %>% anova(test = "F") %>% report::report()
```

```
## The ANOVA suggests that:
##
##   - The main effect of supp is statistically significant and large (F(1, 57) =
## 11.45, p = 0.001; Eta2 (partial) = 0.17, 95% CI [0.05, 1.00])
##   - The main effect of dose is statistically significant and large (F(1, 57) =
## 123.99, p < .001; Eta2 (partial) = 0.69, 95% CI [0.57, 1.00])
##
## Effect sizes were labelled following Field's (2013) recommendations.
```

In the context of linear regression modeling in R, using **anova(lm_mod1, test = "F-test")** is more appropriate than using **summary(lm_mod1)** when you want to compare the fit of nested models or assess the overall significance of a group of predictors. Here's why:

1. **Comparison of Nested Models:** **anova(lm_mod1, test = "F-test")** is particularly useful when you want to compare two or more nested linear regression models. Nested models are those where one model is a subset of the other, typically achieved by adding or removing predictor variables. The F-test provided by **anova** helps you determine whether the inclusion of additional predictors significantly improves the model fit. This is essential for model selection and assessing the relevance of specific predictors.
2. **Hypothesis Testing for Groups of Predictors:** Sometimes, you may want to test the overall significance of a group of predictors rather than examining each predictor individually. The F-test allows you to test the null hypothesis that all the coefficients associated with a specific group of predictors are equal to zero simultaneously. This is useful in scenarios where you have multiple predictors with a similar theoretical basis (e.g., multiple related ecological variables) and want to determine if, collectively, they contribute significantly to explaining the response variable.
3. **Model Comparison:** The **anova** function provides a way to perform statistical tests for model comparison. By comparing nested models or models with different sets of predictors, you can make informed decisions about which variables are essential for explaining the variance in the response variable and which can be omitted. This helps in simplifying models and avoiding overfitting.

In contrast, **`summary(lm_mod1)`** typically provides detailed information about the coefficients of the linear regression model, including the estimated coefficients, standard errors, t-values, and p-values for each predictor. While this is valuable for understanding the individual effects of predictors, it doesn't directly address the questions related to model comparison or the overall significance of groups of predictors.

In summary, **`anova(lm_mod1, test = "F-test")`** is a valuable tool for model comparison, assessing the significance of groups of predictors, and making informed decisions about model complexity. It complements the **`summary`** function, which is more focused on providing detailed information about individual predictor coefficients.

6.3.2.2 Performing linear regression in Jamovi

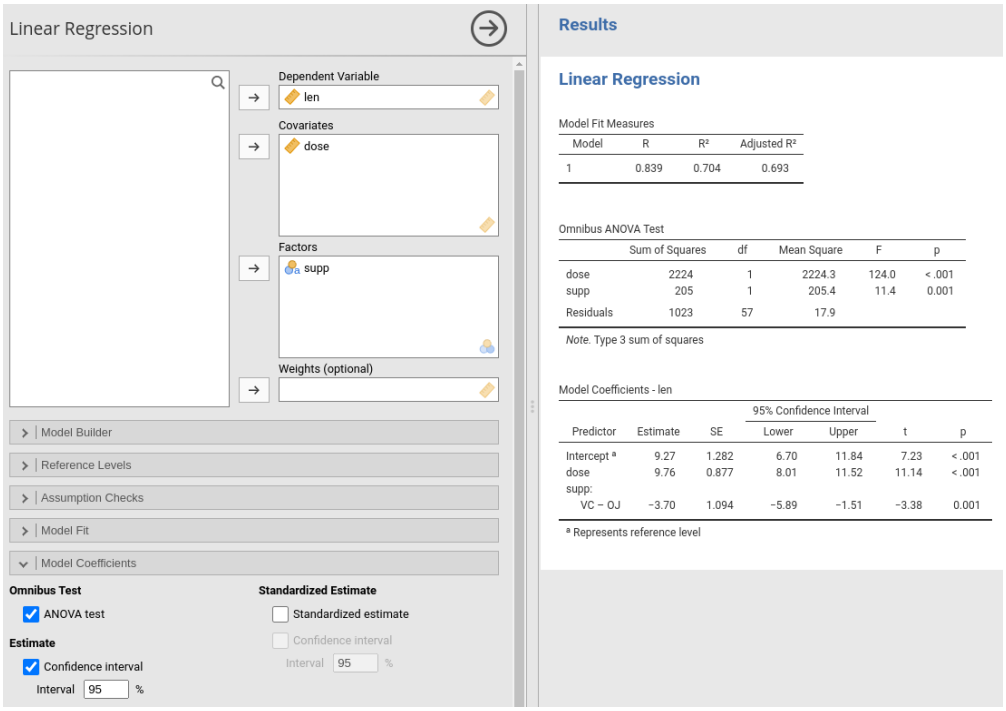
Let's first export the tooth-growth dataset to a csv flat file for use in Jamovi.

1. Write tooth-growth data with name "toothgrowth.csv" to csv in R.

```r
# load packages
pacman::p_load(readr, here, install = TRUE, update = F)

# load dataset
data("ToothGrowth")

# save R dataset to csv file
readr::write_csv(
  ToothGrowth,
  file = here::here("docs", "data", "toothgrowth.csv"),
  col_names = TRUE,
  append = FALSE
)
```

2. Open/ import the file in Jamovi. In Jamovi, the variable "`len`" should be a continuous data type, "`supp`" as nominal and "`dose`" as continuous.

3. Go to "Analyses" tab, click "Regression" button and select "Linear Regression". Perform linear regression by using "`len`" as the Dependent variable, "`supp`" in Factors and "`dose`" as a Covariate. Now compare your results to the previous results generated in R. Interpretations should be the same as those above for R stats. Note report the Adjusted R2 and Omnibus ANOVA Test table.



6.3.3 Interpreting Linear Regression Outputs

1. **Coefficients:** Interpret the coefficients of the IVs. A positive coefficient means that as the IV increases, the DV also tends to increase (and vice versa for negative coefficients).
2. **R-squared (R^2):** R-squared measures the proportion of variance in the DV explained by the IVs. Higher R-squared values indicate a better fit.
3. **P-values:** P-values test the null hypothesis that there's no relationship between IVs and DV. Low p-values (typically < 0.05) indicate statistically significant relationships.
4. **Assumptions:** Assess model assumptions, including linearity, independence of errors, homoscedasticity (equal variance of errors), and normality of residuals. Diagnostic plots help check these assumptions. Note that we did not perform any assumptions assessments prior to running the model. We'll incorporate this workflow in later models.

In ecological research, linear regression provides valuable insights into the relationships between ecological variables. It helps answer questions about ecological processes and how they are influenced by environmental factors. Proper interpretation of model results and assessment of assumptions are crucial for robust ecological conclusions.

For more information on using Jamovi for data analyses see: https://www.jamovi.org/user-manual.html, https://docs.jamovi.org/ and https://dr-jmoses.github.io/Workshop-Training-Manual/.

Reference

Bache, S. M., & Wickham, H. (2022). *Magrittr: A forward-pipe operator for r*. https://CRAN.R-project.org/package=magrittr

Csárdi, G., Hester, J., Wickham, H., Chang, W., Morgan, M., & Tenenbaum, D. (2023). *Remotes: R package installation from remote repositories, including 'GitHub'*. https://CRAN.R-project.org/package=remotes

Kassambara, A. (2023). *rstatix: Pipe-friendly framework for basic statistical tests*. https://CRAN.R-project.org/package=rstatix

Lüdecke, D., Ben-Shachar, M. S., Patil, I., Waggoner, P., & Makowski, D. (2021). *{Performance}: An {r} package for assessment, comparison and testing of statistical models*. *6*, 3139. https://doi.org/10.21105/joss.03139

Lüdecke, D., Ben-Shachar, M. S., Patil, I., Wiernik, B. M., & Makowski, D. (2022). easystats: Framework for easy statistical modeling, visualization, and reporting. *CRAN*. https://easystats.github.io/easystats/

Makowski, D., Lüdecke, D., Patil, I., Thériault, R., Ben-Shachar, M. S., & Wiernik, B. M. (2023). Automated results reporting as a practical tool to improve reproducibility and methodological best practices adoption. *CRAN*. https://easystats.github.io/report/

R Core Team. (2023). *R: A language and environment for statistical computing (Version 4.3. 1)[Computer software]. Vienna, Austria: R Foundation for Statistical Computing*. https://www.r-project.org/

RStudio Team. (2020). *RStudio: Integrated development environment for r*. RStudio, PBC. http://www.rstudio.com/

Şahīn, M., & Aybek, E. (2020). Jamovi: An Easy to Use Statistical Software for the Social Scientists. *International Journal of Assessment Tools in Education*, *6*(4, 4), 670–692. https://doi.org/10.21449/ijate.661803

Selker, R. (2017). *scatr: Create scatter plots with marginal density or box plots*. https://CRAN.R-project.org/package=scatr

Selker, R., Love, J., Dropmann, D., & Moreno, V. (2022). *jmv: The "jamovi" analyses*. https://CRAN.R-project.org/package=jmv

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., … Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, *4*(43), 1686. https://doi.org/10.21105/joss.01686

Wickham, H., Hester, J., & Bryan, J. (2023). *Readr: Read rectangular text data*. https://CRAN.R-project.org/package=readr

Wickham, H., Hester, J., Chang, W., & Bryan, J. (2022). *Devtools: Tools to make developing r packages easier*. https://CRAN.R-project.org/package=devtools

Xie, Y. (2023). *knitr: A general-purpose package for dynamic report generation in r*. https://yihui.org/knitr/