DS311
Exploratory Data Analysis

Introduction to EDA

Overview: Exploratory Data Analysis (EDA) is an approach to analyzing datasets in order to summarize their main characteristics—often using statistics and visualizations—without initially applying a formal model or hypothesis. EDA is a crucial step in any data science project: it helps us understand data structure, identify patterns or anomalies, test early hypotheses, and make informed decisions about data preparation and modeling. In contrast to confirmatory analysis (where we have preconceived hypotheses and models), EDA encourages *letting the data speak* by uncovering insights beyond what formal modeling might reveal. This learning portfolio will introduce EDA concepts and demonstrate them in R with step-by-step examples. We will also compare EDA to classical and Bayesian analyses, survey software tools for EDA, place EDA in the data science lifecycle, and illustrate how to perform univariate, bivariate, and multivariate EDA. Finally, a hands-on lab with the **Palmer Penguins** case study will allow you to practice EDA and solidify your understanding.

Prerequisites

- **Software:** You should have R installed on your system. Using RStudio as an IDE is recommended for an easier R coding experience.
- **Basic Knowledge:** This guide assumes basic familiarity with R syntax and data frames (loading data, calling functions, etc.), but we will explain each step in the context of EDA.
- **Packages:** We will use functions from base R and the **tidyverse** collection for data manipulation and visualization. If not already installed, you can install the necessary packages by running:

```
install.packages("tidyverse")       # for dplyr, ggplot2, etc.
install.packages("palmerpenguins") # for the example dataset in the lab
```

Load the packages in your R session:

```r
library(tidyverse)      # loads dplyr, ggplot2, etc.
library(palmerpenguins) # loads the Palmer Penguins dataset for the lab
```

The code examples and lab are designed for an RMarkdown document (or R script) format, so you can copy-paste the code chunks into RStudio and execute them. All datasets used are either built-in (like `iris`) or provided via an R package or simulation, ensuring that you can reproduce the analysis on your own machine.

1. Introduction to Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an approach to **explore and summarize data** to uncover its important characteristics. John Tukey introduced EDA in the 1970s as a contrast to the heavily model-driven, hypothesis-testing focus of classical statistics. In EDA, we **don't start with a strict model or hypothesis**; instead, we examine the data openly to find patterns, spot anomalies, check assumptions, and suggest hypotheses. As one definition puts it, EDA is about *"seeing what the data can tell beyond the formal modeling,"* as opposed to traditional confirmatory analysis where a model is chosen before looking at the data.

Why is EDA important? It allows us to validate that our data is correct and ready for analysis, and it often reveals insights that inform the next steps of a project. EDA helps in understanding the data's underlying structure, identifying outliers or errors, discovering interesting relationships, and guiding us on feature selection or modeling strategy. For example, EDA might show that a variable is skewed (suggesting a transformation might be needed) or that two variables are highly correlated (which could influence model selection or multicollinearity concerns). It's during EDA that you might detect data quality issues (like missing values or inconsistent entries) and decide how to handle them. Essentially, EDA is a **prelude to formal analysis** that ensures you don't go into modeling blindly.

First steps in EDA. When you first obtain a dataset, some common EDA steps include: checking its dimensions (how many rows and columns), looking at a sample of records, and summarizing each feature's distribution. Let's illustrate these basics with R's built-in `iris` dataset (measurements of 150 flowers from 3 species):

```r
# Quick peek at the data
dim(iris)      # dimensions: number of rows and columns
```

```
## [1] 150   5
```

```r
head(iris, 5)  # first 5 rows of the dataset
```

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

```r
str(iris)      # structure of the dataset: data types and sample values
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
```

```
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Running the above, we would see that `iris` has 150 rows and 5 columns. The `head()` shows the first few records of sepal length, sepal width, petal length, petal width, and species. `str(iris)` reveals that the first four columns are numeric (measurements in cm) and the last column (`Species`) is a factor with 3 levels (`setosa`, `versicolor`, `virginica`). Next, we might want summary statistics:

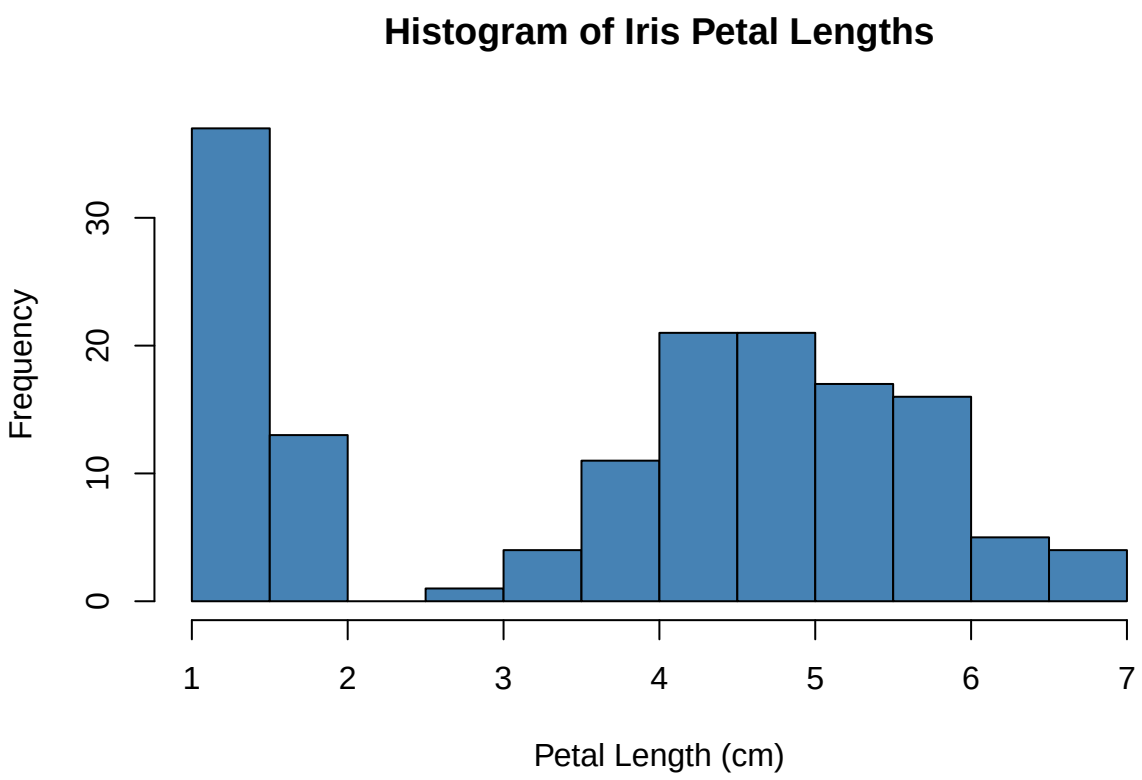
```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

The `summary()` function outputs, for each numeric column, the minimum, 1st quartile, median, mean, 3rd quartile, and maximum, and for the factor column it shows the count of each species. For example, you would find that **Sepal.Length** in this sample ranges from 4.3 to 7.9 (cm) with a mean around 5.84, and each species appears 50 times in the data. Already, this tells us the scale and rough distribution of each measurement and confirms that the dataset is balanced across species.

EDA often involves **data visualization** as well. Even a simple histogram or boxplot can be very revealing. For instance, a quick plot of the distribution of petal lengths might show a bimodal shape, hinting that different species have different petal sizes:

```
hist(iris$Petal.Length,
     main="Histogram of Iris Petal Lengths",
     xlab="Petal Length (cm)", col="steelblue")
```



This histogram would likely show two clusters of petal length values (shorter petals for *Iris setosa* and longer for the other two species), exemplifying how EDA can uncover structure in the data. We'll delve deeper into specific types of plots and analyses in Section 5, but the key point is: **EDA marries computation with visualization** to let you *explore* data, not just confirm what you expect.

2. EDA vs. Classical and Bayesian Analysis

It's helpful to understand how EDA differs from other approaches to data analysis, notably **classical (confirmatory) analysis** and **Bayesian analysis**. All three approaches ultimately aim to draw conclusions from data, but they differ in *when and how models are used* in the process.

- **Classical Analysis (Confirmatory Data Analysis):** In the classical approach, you begin with a well-defined **hypothesis or model** before looking at the data. The typical sequence is

  Problem/Question $\Rightarrow$ *Data* $\Rightarrow$ ***Model*** $\Rightarrow$ *Analysis* $\Rightarrow$ *Conclusions.*

  For example, a classical analysis might start with the assumption that data follows a normal distribution or that two groups have equal means, then use statistical tests (t-tests, ANOVA, etc.) to confirm or reject those assumptions at a certain significance level. In classical analysis, **model assumptions (like normality, linearity)** are imposed early, and the analysis focuses on estimating model parameters and testing hypotheses about them. This approach is rigorous and focuses on confirming* or rejecting predefined hypotheses.

- **Exploratory Data Analysis (EDA):** In EDA, the sequence is

  Problem $\Rightarrow$ *Data* $\Rightarrow$ ***Analysis*** $\Rightarrow$ *Model* $\Rightarrow$ *Conclusions.*

  That is, you postpone imposing any strict model until you've had a look at the data. The analysis phase in EDA involves plotting data, calculating summary statistics, and looking for meaningful

patterns without the constraint of a prior model. **No distribution or relationship is assumed a priori**. Instead, you might infer what model could be appropriate **after** exploring the data. As Tukey emphasized, EDA is about hypothesis generation* more than hypothesis testing. For example, rather than immediately performing a t-test, an EDA practitioner might first plot the data and notice a skew or outliers that suggest the need for a different test or a transformation. EDA is thus more flexible and creative, aiming to **discover the unexpected** and to check whether the assumptions required for classical analysis even hold.

- **Bayesian Analysis:** The Bayesian approach can be seen as an extension of the classical model-based approach that explicitly incorporates prior knowledge or beliefs. Its sequence is

  Problem ⇒ *Data* ⇒ *Model* ⇒ ***Prior*** ⇒ *Analysis* ⇒ *Conclusions*.

  In Bayesian analysis, like classical, you do choose a model for the data (e.g., a regression model); but in addition, you specify a **prior distribution** for the model's parameters reflecting any existing knowledge or plausible ranges for those parameters. The analysis then uses Bayes' theorem to update these priors with the observed data to produce a **posterior** distribution for the parameters. For example, if past studies suggest a parameter is likely small, a Bayesian might use that information as a prior, whereas a classical analysis would treat all parameter values as equally plausible before seeing the data. Bayesian analysis thus formalizes learning from data combined with prior information, while still ultimately relying on a model (like classical analysis) – albeit with an added layer of probabilistic reasoning about parameters.

Classical analysis is confirmatory and model-driven (you impose a model and test if data fits it), **EDA** is discovery-driven and lets the data itself suggest which models or hypotheses might make sense, and **Bayesian analysis** is model-driven but with an explicit incorporation of prior knowledge. In practice, analysts often mix elements of all three approaches as needed. For instance, you might do an initial EDA to guide which formal tests to run (classical), or perform a Bayesian analysis but use EDA to check if the model fits the data well. The key takeaway is that EDA provides a **toolkit and mindset** for **open-ended exploration**, complementing the more structured classical/Bayesian methods rather than replacing them.

Example: Suppose we have data on exam scores for students before and after a training program. A **classical analysis** might start with a hypothesis "the training improves scores on average" and perform a paired t-test (assuming normal distribution of score differences) to confirm if the improvement is statistically significant. An **EDA approach** would first plot the before vs. after scores, perhaps revealing that improvement varies widely among students or that a few students did much worse (outliers) – insights that might lead us to refine our analysis (or even question our hypothesis). A **Bayesian approach** to the same problem might begin with a belief (prior) that most students improve by, say, 5 points ± 5, then update that belief with the observed data to get a distribution for the true average improvement. Each approach provides a different perspective, and EDA in particular ensures that we **understand the data's story** before formalizing it with a model.

3. Software Tools for EDA

A wide array of software tools can be used for exploratory data analysis. Here we highlight some popular ones and where R fits in:

- **Programming Languages (R and Python):** Both R and Python are dominant tools for EDA in the data science community. **R** is an open-source language built for statistics and graphics, widely used by statisticians and data analysts for its powerful visualization and data manipulation libraries. In R, packages like `dplyr` (for data wrangling), `ggplot2` (for plotting), and many others in the tidyverse make EDA efficient and expressive. **Python**, on the other hand, is a versatile general-purpose language that, with libraries such as **Pandas** (data manipulation), **Matplotlib/Seaborn** (visualization), and **NumPy** (numerical computing), also provides a robust environment for EDA. Python's tools excel at handling data frames, missing values, and integrating with machine learning workflows. In practice, R

is often praised for rapid interactive analysis and rich statistical packages, while Python is chosen for integration with broader applications – but both are excellent for EDA.

- **Spreadsheet Software (Excel and equivalents):** For many beginners, **Excel** (or Google Sheets) is the first tool for data exploration. Spreadsheets allow you to sort and filter data, compute summary statistics with formulas, and create simple charts. They are intuitive for smaller datasets and quick exploratory checks. Excel has the advantage of a visual, cell-based interface and is quite reliable for straightforward analysis. However, it can become cumbersome or error-prone with large datasets or complex transformations. While not as reproducible or powerful as R/Python, spreadsheets remain a useful EDA tool for quick, ad-hoc exploration or for those without programming experience.
- **Interactive Visualization and BI Tools:** Dedicated data visualization or business intelligence (BI) tools like **Tableau**, **Power BI**, or **Spotfire** provide user-friendly drag-and-drop interfaces to explore data and create visuals. **Tableau**, for example, allows you to connect to a dataset and instantly try out different charts; it even suggests suitable chart types based on your fields. Such tools enable **interactive EDA** – you can filter, drill-down, and highlight patterns on the fly, which helps in understanding data in real time. Tableau Public (the free version) is popular for creating shareable interactive dashboards and is known for its ability to produce impactful visuals quickly. The trade-off is that these tools, while great for exploration and communication, may abstract away the underlying data manipulations and are less flexible for custom analysis compared to programming.
- **Statistical Software (e.g., SAS, SPSS) and Others:** Apart from R/Python, traditional statistical packages like SAS, SPSS, or **Stata** also offer EDA capabilities (with GUI and scripting options for summary stats and plots). There are also domain-specific tools like **MATLAB** (popular in engineering) and others like **Weka** (machine learning tool with a GUI) which include data exploration features. Moreover, Jupyter notebooks (for Python/R) and RMarkdown documents allow you to mix code with narrative, making them excellent for conducting and documenting an EDA session.

In this portfolio, our focus is on **R** as the EDA tool. R provides both **base R** functions and an ecosystem of packages to make exploratory analysis easier:

- *Base R tools:* Functions like `summary()`, `str()`, `head()`, `plot()`, `hist()`, `boxplot()`, `pairs()`, `table()`, etc., are readily available for quick analysis and visualization. Base R plotting is straightforward for simple graphs, and basic statistical summaries are one function call away (as we saw with `summary(iris)`).
- *Tidyverse:* The tidyverse collection (which you load via `library(tidyverse)`) offers a **"grammar" of data manipulation and graphics**. With **dplyr**, you can filter or aggregate data in a readable way (e.g., `penguins %>% group_by(species) %>% summarize(mean_body_mass = mean(body_mass_g, na.rm=TRUE))`). With **ggplot2**, you can create complex visuals by layering components (we'll demonstrate examples later). The advantage of tidyverse for EDA is that it makes code more legible and analysis steps more reproducible.
- *EDA-specific packages:* There are also packages aimed specifically at simplifying EDA. For example, **`skimr`** can give a compact summary of an entire data frame (including counts of missing values and distribution of each variable) with one function call `skim(data)`. **`DataExplorer`** and **`SmartEDA`** can generate automated EDA reports (plots and tables for every variable) to give you an initial overview. **`GGally`** extends ggplot2 to create matrix plots (scatterplot matrices, correlation plots) easily. **`janitor`** helps with quickly tabulating categorical data and cleaning variable names. As an example, using `skimr::skim(iris)` will list for each column its type, missing count, and statistics like mean, sd, histogram sparkline, etc., in a nice format. While we won't rely on these specialized packages in our core examples (to avoid extra installation steps for beginners), be aware that they exist to turbocharge your EDA when needed.
- *Interactive R tools:* RStudio's built-in **View()** function lets you browse data in a spreadsheet-like viewer, which can be handy. There are also interactive plotting libraries like **plotly** or **Shiny** apps that

can be created for data exploration. For instance, `plotly` can turn static ggplot graphs into interactive ones (so you can hover to see values). These are more advanced use-cases, but they highlight that R can support not just static analysis but interactive EDA as well.

Choosing a tool: In practice, you might use a combination of tools. For a small dataset, a quick look in Excel might precede a more thorough analysis in R. Or you might do initial exploration in R, then build a Tableau dashboard to communicate findings. The key is to use the strengths of each tool appropriately. In this guide, we'll proceed with R for all coding examples to demonstrate a **reproducible EDA workflow**. Using R means that every step of our exploration (from generating a statistic to plotting a chart) is documented in code, which is invaluable for sharing your analysis or revisiting it later.

4. EDA in the Data Science Lifecycle

Where does EDA fit into the **data science lifecycle**? In virtually every framework for data science or data mining projects (such as CRISP-DM, OSEMN, or the Data Science Workflow), EDA is a prominent stage. Typically, the process goes through phases like: **understanding the problem, acquiring data, cleaning/preparing data, exploring data (EDA), modeling, evaluating, and deploying**, often with iterative feedback loops. EDA usually occurs **after initial data cleaning but before formal modeling** – though in practice, EDA and data cleaning often go hand-in-hand.

Consider the CRISP-DM process (CRoss Industry Standard Process for Data Mining): it includes a **"Data Understanding"** phase, which is essentially where EDA happens. After you've collected or extracted the data, you need to *understand* it by exploring its characteristics. In the OSEMN model (Obtain, Scrub, Explore, Model, iNterpret), the **Explore** step is explicitly EDA. The goal in this phase is to **get to know the data deeply**: what distributions do your variables have? Are there outliers or missing values? What relationships exist between variables? Answering these questions ensures that you proceed to modeling with correct assumptions and a clear strategy.

EDA's role in the lifecycle can be summarized as follows:

- **Validating Data Quality:** EDA helps check if the data is adequate and correct for the intended analysis. For example, you might discover during EDA that one sensor in an IoT dataset was malfunctioning (showing constant values), prompting you to discard or correct those readings before modeling. Or you might realize a demographic survey has mostly respondents of one category, which will influence how you interpret any models.
- **Guiding Data Preparation:** Insights from EDA inform how to clean or transform data further. If a variable has a skewed distribution, you might decide to apply a log transform. If two variables are perfectly correlated, you might drop one to simplify a predictive model. If certain categories in a factor have very low counts, you might combine or ignore them. EDA effectively tells you *what needs to be done* to the data prior to robust analysis.
- **Informing Feature Engineering:** In machine learning projects, EDA gives inspiration for feature creation. You might notice non-linear trends or interactions between variables that suggest creating a combined feature or using polynomial terms in a model. For instance, exploring a dataset of houses, you might find that price per square foot is more informative than total price in some analyses – that insight could lead to a new feature.
- **Selecting Modeling Techniques:** The patterns uncovered in EDA can influence what modeling approach to take. If EDA shows a complex nonlinear relationship between predictors and outcome, you might opt for a more flexible model (like a random forest instead of a linear regression). If EDA reveals clusters in your data, you might consider a clustering or segmentation analysis as part of your project. Essentially, EDA can suggest *which questions are worth asking* and *which models might answer them*. It can also help verify if the assumptions of a planned model are met (e.g., checking linearity and homoscedasticity for a regression model through scatterplots and residual plots, which is a form of exploratory analysis).

- **Contextualizing the Results:** Even after modeling, EDA remains useful. For example, if a model yields an unexpected result, you might go back to the data and do further exploratory checks on specific subsets to understand why. In an iterative lifecycle, after model evaluation, you might refine your data or model based on additional EDA (this is part of the **feedback loop** in data science).

In summary, EDA is **woven throughout** a data science project. Early on, it's a distinct phase where you **explore the data without heavy assumptions**, building intuition and knowledge. Later, it can re-emerge whenever you need to interpret or troubleshoot results. Skipping EDA can be risky – it's like going on a road trip without looking at a map. You might eventually get to the destination (a model or insight), but you're more likely to get lost, take wrong turns, or miss interesting sights along the way. By doing EDA, you equip yourself with a mental map of the data landscape, which leads to better decisions in analysis and modeling.

> **Real-world example:** Suppose you're working on a data science project to predict customer churn for a telecom company. The data science lifecycle would involve understanding business goals, collecting customer data, preparing it, exploring it, modeling, etc. In the EDA phase, you might discover, for example, that customers in certain regions have much higher churn rates, or that data for a particular month is incomplete. These findings could prompt you to incorporate region as a key feature in your model or to be cautious using data from that month. Without EDA, you might have built a model oblivious to these facts, potentially making it less accurate or trustworthy.

5. How Does EDA Work? (Univariate, Bivariate, Multivariate Analysis)

Now, let's get into the **practical techniques** of EDA. We can categorize exploratory analysis by the number of variables we examine at once:

- **Univariate analysis:** Exploring one variable at a time.
- **Bivariate analysis:** Exploring relationships between two variables.
- **Multivariate analysis:** Exploring three or more variables simultaneously (or the combined effect of multiple variables).

Each level of analysis provides different insights. We will demonstrate each with examples in R, using straightforward code (base R and tidyverse) and simple graphics. For these examples, we'll continue to use the `iris` dataset for convenience. (In the lab, you'll apply similar techniques to a different dataset to practice.)

5.1 Univariate Analysis

Univariate EDA examines the distribution of a single variable. The goal is to understand the **central tendency** (mean, median), **spread** (variance, IQR), **shape** of distribution (symmetry, skewness, modality), and presence of **outliers** for that variable. The type of analysis depends on whether the variable is numeric or categorical:

- **For a numeric variable:** We typically look at summary statistics and distribution plots. Key stats include mean, median, standard deviation, quartiles (the five-number summary), etc., which we can get via functions like `summary()` or `quantile()`. Visualization tools include histograms, density plots, and boxplots.
- **For a categorical (factor) variable:** We consider the frequency of each category. A frequency table (counts or proportions) can be produced, and a bar chart (or pie chart, though bar is usually clearer) can display the relative frequencies.

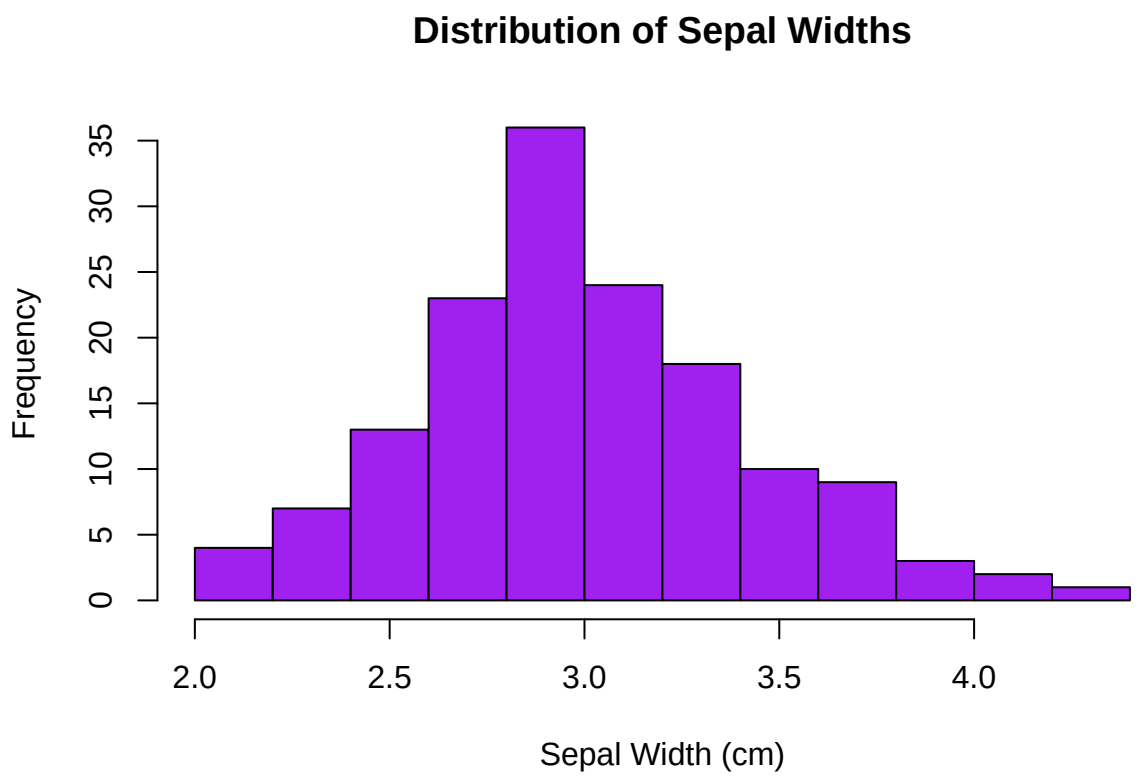
Example (numeric): Let's examine the distribution of *Sepal.Width* in the iris dataset.

```
# Summary statistics for a numeric variable
summary(iris$Sepal.Width)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   2.800   3.000   3.057   3.300   4.400
```

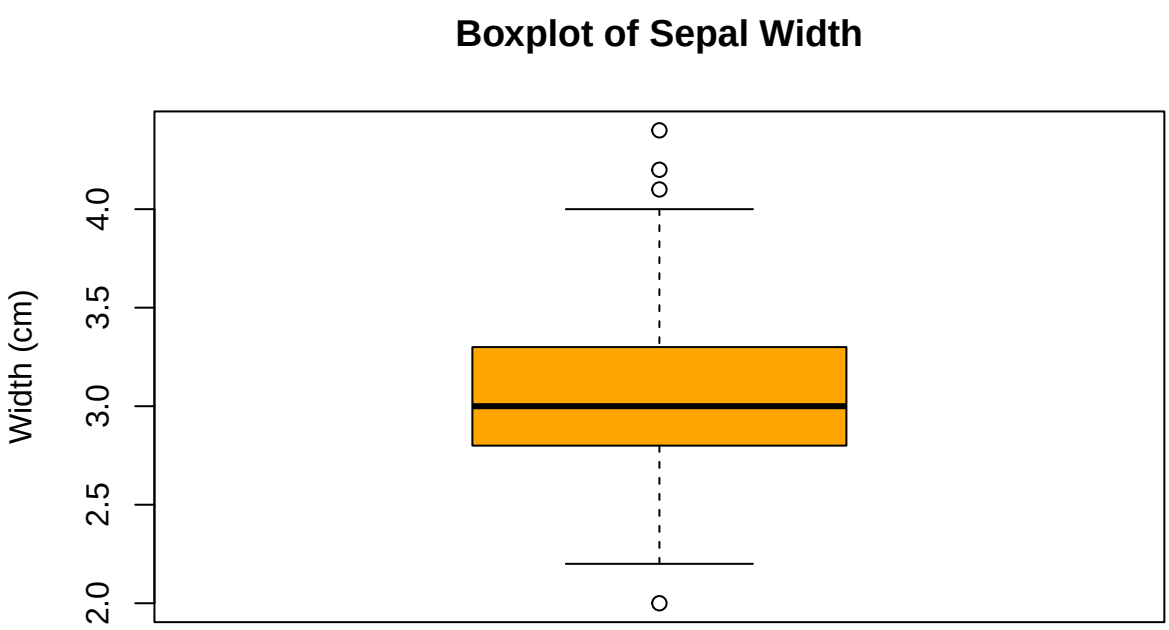
This will output something like: Min, 1st Qu., Median, Mean, 3rd Qu., Max of Sepal.Width. From the iris data, we would see the mean sepal width is about 3.06 cm, with a minimum of 2.0 and maximum of 4.4. The median (~3.0) is close to the mean, suggesting a somewhat symmetric distribution, but let's visualize it to be sure:

```
# Histogram of Sepal.Width
hist(iris$Sepal.Width,
     main="Distribution of Sepal Widths",
     xlab="Sepal Width (cm)", col="purple")
```



The histogram shows the shape of the distribution. In iris, sepal widths are roughly bell-shaped but slightly skewed (there might be a longer tail on the higher end). We might also spot if there are any unusual gaps or values. Another useful plot is a **boxplot**, which succinctly shows the median, quartiles, and any outliers:

```
boxplot(iris$Sepal.Width,
        main="Boxplot of Sepal Width", ylab="Width (cm)", col="orange")
```



The boxplot will show the IQR (box), median (line), and "whiskers" extending to within 1.5×IQR. Any point beyond that would be marked as an outlier. In this case, iris sepal widths might show a few points considered moderately high (around 4 cm) as potential outliers. Boxplots are great for quickly assessing symmetry and outliers: if the median is centered and whiskers roughly equal, distribution is fairly symmetric; if not, it indicates skew.

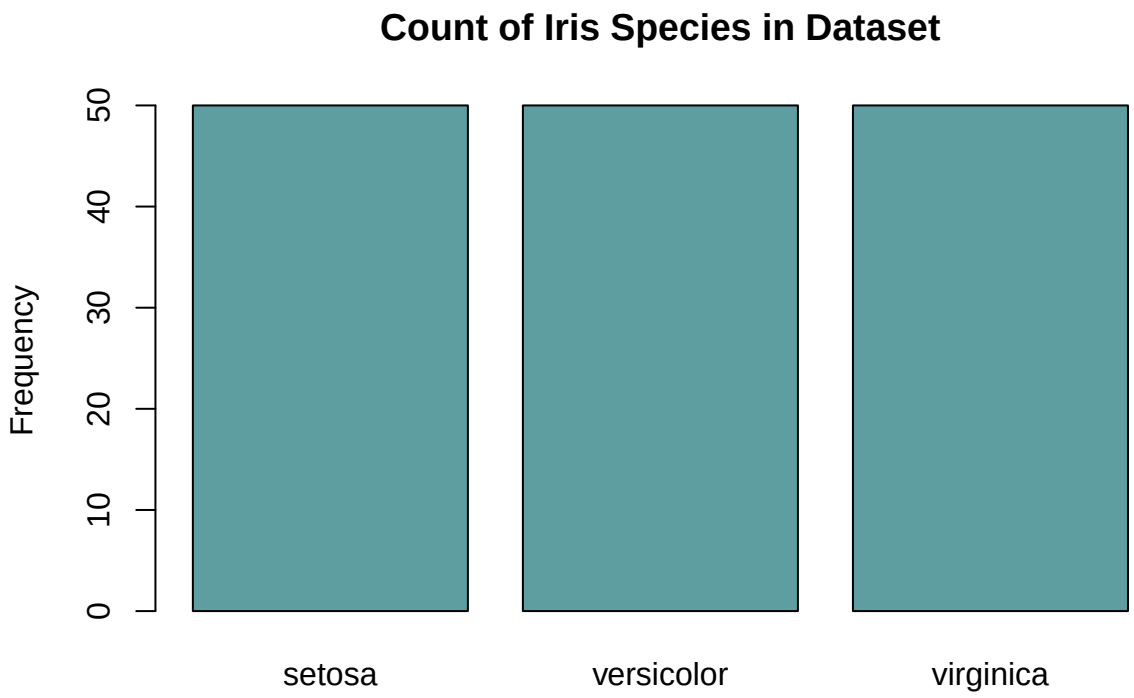
Example (categorical): Now consider the factor variable *Species* in iris. We can see how many samples of each species are present:

```
# Frequency table for a categorical variable
table(iris$Species)
```

```
##
##     setosa versicolor  virginica
##         50         50         50
```

This will output the counts for each of the three species (setosa, versicolor, virginica). We expect 50 each, since the iris dataset was constructed that way. To visualize this:

```
# Bar plot of species frequencies
species_counts <- table(iris$Species)
barplot(species_counts,
        main="Count of Iris Species in Dataset",
        ylab="Frequency", col="cadetblue")
```



The barplot shows three bars of equal height (50 each), confirming the dataset is balanced. In other datasets, this kind of plot could quickly reveal, for example, class imbalances (one category dominates) or missing category levels, etc.

Beyond counts, for categorical data you might also look at proportions (e.g., 33% setosa, etc.) or even introduce visuals like pie charts (though bar charts are usually preferred for clarity).

Key things to look for in univariate EDA:

- For **numeric variables**: Is the distribution **normal-ish or skewed**? Are there **multiple peaks** (bimodal, which might indicate subgroups in data)? Are there **outliers** (extremely high or low values that stand apart)? What is the **range** and are values within expected bounds (e.g., no negative ages, etc.)? In iris, for instance, petal lengths are clearly bimodal (setosa vs others), which would prompt further investigation by species.
- For **categorical variables**: What is the **most common category**? Are some categories very rare (which might need combining or special treatment)? If it's a high-cardinality category (many levels, like a "City" field), maybe group or visualize top N categories. No such issue in iris since species has only 3 levels, but in a dataset with 50 different species, a barplot might need to be sorted or truncated for readability.

Univariate EDA builds the foundation. By the end of it, you should have a decent idea of each variable's behavior in isolation. Next, we examine relationships between variables.

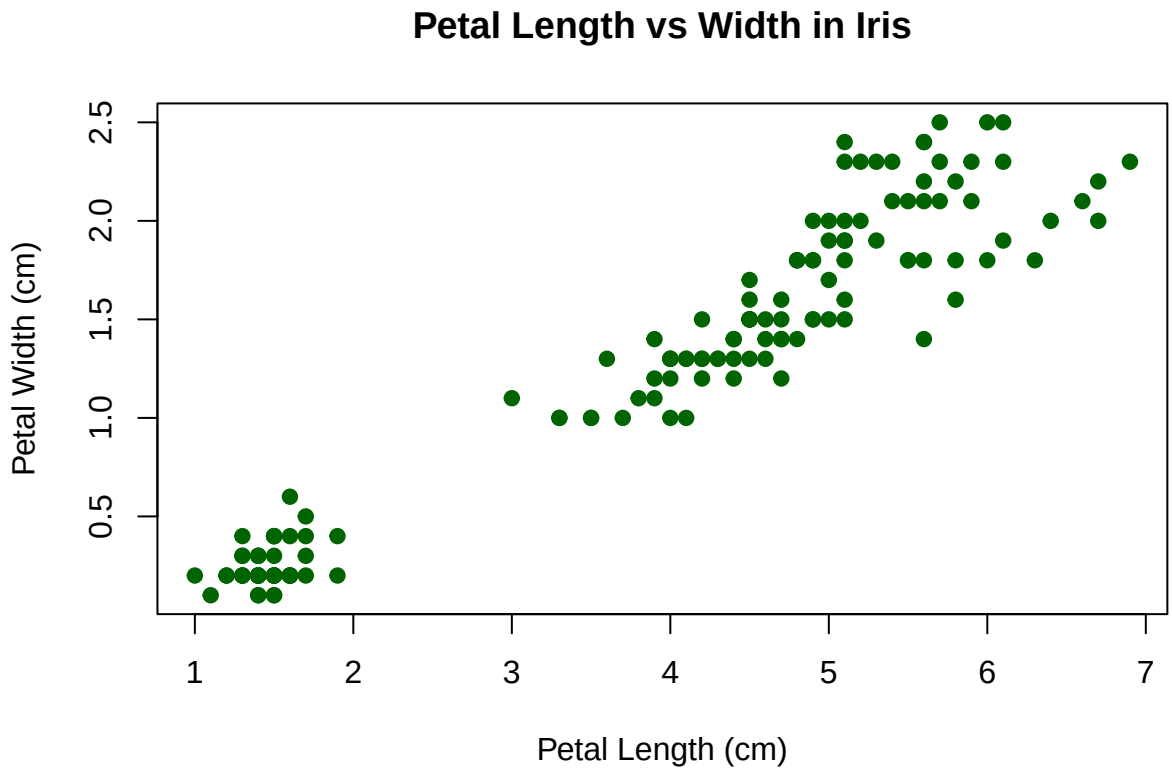
5.2 Bivariate Analysis

Bivariate EDA looks at **pairwise relationships** between two variables at a time. The combination of variable types (numeric or categorical) dictates the techniques:

- **Numeric vs Numeric:** Use scatter plots to visualize the relationship, and compute correlation coefficients to quantify linear relationship (if appropriate). You may also add trend lines or use transformation if the relationship is nonlinear.
- **Numeric vs Categorical:** Compare the numeric distributions across the categories. Techniques include side-by-side boxplots (or violin plots) for each category, or computing group-wise statistics (means, medians) to see differences. This helps identify if a category has an effect on a numeric outcome (e.g., does species affect petal length? Clearly yes in iris).
- **Categorical vs Categorical:** Use contingency tables (cross-tabulation) to see how the categories of one variable are distributed across the other. Visualization can be done via grouped bar charts or mosaic plots. One can also perform a chi-squared test to check if the categories interact significantly, but in EDA we often focus on the counts and proportions to see the pattern.

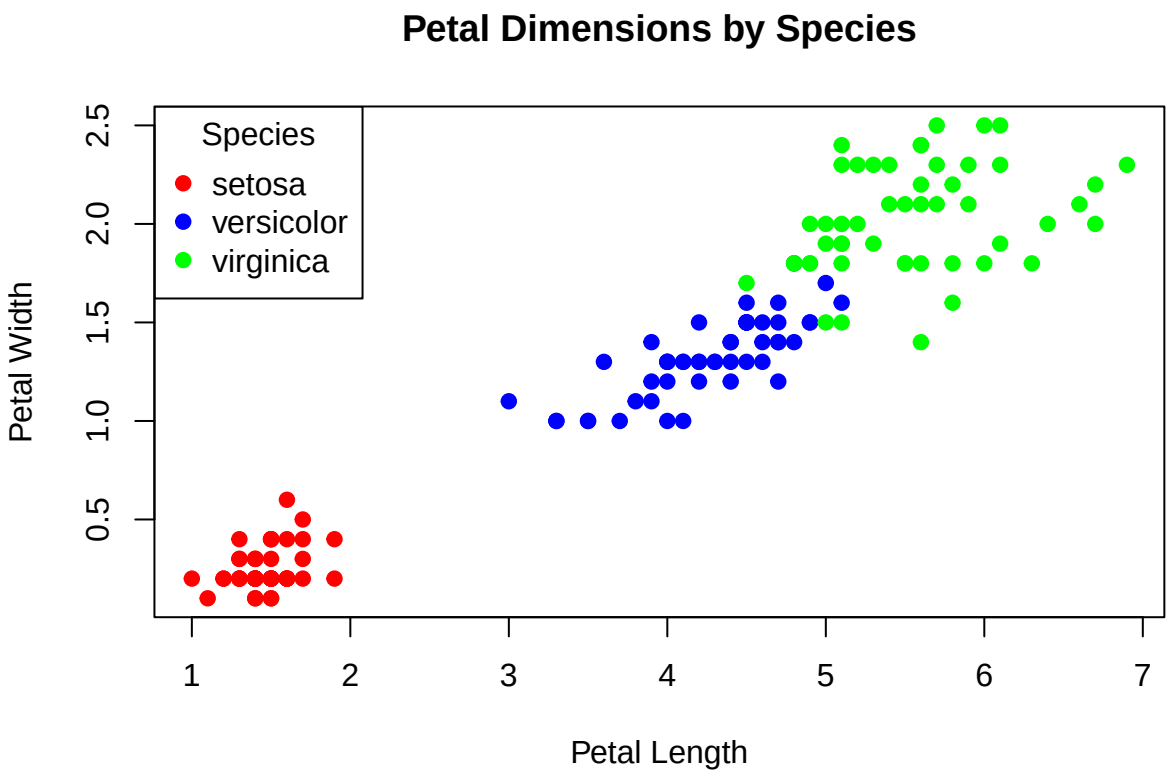
Example (numeric-numeric): Let's examine the relationship between *Petal.Length* and *Petal.Width* in the iris dataset. We suspect they are related (longer petals likely are also wider).

```
# Scatter plot for two numeric variables
plot(iris$Petal.Length, iris$Petal.Width,
     main="Petal Length vs Width in Iris",
     xlab="Petal Length (cm)", ylab="Petal Width (cm)",
     col="darkgreen", pch=19)
```



This produces a scatter plot of petal length against petal width. In iris, we see a very clear positive correlation: points form an upward sloping trend. In fact, it looks almost like three clusters of points, which correspond to the three species (setosa points form one cluster with short petals, virginica points on the high end with long/wide petals, and versicolor in the middle). If we color-coded by species, this would become even more evident. For illustration:

```r
# Scatter plot with points colored by species (requires converting species to colors)
colors <- c("red", "blue", "green")[iris$Species]
plot(iris$Petal.Length, iris$Petal.Width,
     col=colors, pch=19,
     xlab="Petal Length", ylab="Petal Width", main="Petal Dimensions by Species")
legend("topleft", legend=levels(iris$Species),
       col=c("red","blue","green"), pch=19, title="Species")
```



Now each species' points are a different color, and indeed we see distinct clusters (setosa in red forms a tight cluster at the low end of both dimensions, etc.). Even without color, the separation was noticeable as a multi-modal pattern in the scatter. This highlights how EDA can reveal not just overall trends but also hints of a **multivariate structure** (more on that in the next section).

We can also quantify the strength of linear relationship with Pearson's correlation:

```r
cor(iris$Petal.Length, iris$Petal.Width)
```

```
## [1] 0.9628654
```

The correlation coefficient here would be very high (close to 0.96 for petal length & width, indicating a strong linear relationship). It's good practice to look at scatterplots *and* correlations: the scatterplot ensures you're not being fooled by outliers or nonlinear effects (since correlation only measures linear association). For example, if we correlated sepal length and sepal width in iris, we'd get a small negative correlation (~ -0.12), and a scatterplot confirms there's no strong linear relationship (the points look more like a blob with maybe a slight downward tilt).

Example (numeric-categorical): Now consider *Sepal.Length* (numeric) across the three *Species* (categorical). We want to see if different species tend to have different sepal lengths.

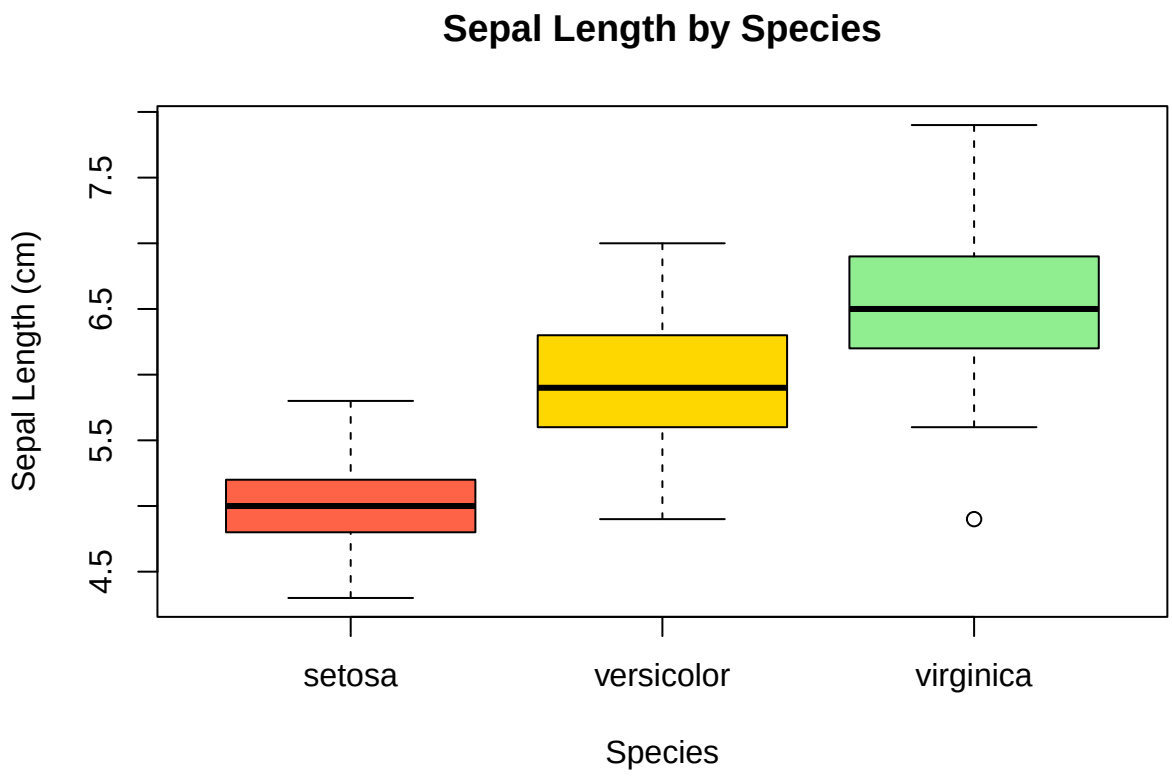
One way is to compare group statistics:

```
# Compute mean Sepal.Length for each species
iris %>% group_by(Species) %>% summarize(mean_sepal_len = mean(Sepal.Length))
```

```
## # A tibble: 3 x 2
##   Species    mean_sepal_len
##   <fct>               <dbl>
## 1 setosa               5.01
## 2 versicolor           5.94
## 3 virginica            6.59
```

This will output the mean sepal length for each species. (In base R, you could do `tapply(iris$Sepal.Length, iris$Species, mean)` to similar effect.) We would find, for instance, that **setosa** has the smallest sepal length on average, while **virginica** has the largest. But beyond averages, we should see the spread:

```
# Boxplots of Sepal.Length by Species
boxplot(Sepal.Length ~ Species, data=iris,
        main="Sepal Length by Species", ylab="Sepal Length (cm)",
        col=c("tomato","gold","lightgreen"))
```



This draws three side-by-side boxplots, one for each species' sepal length distribution. We can observe the median line for each and the IQR box. In iris, the species are pretty well-separated: *setosa* sepals are

clearly shorter (box entirely lower) than the other two species, and *virginica* sepals tend to be longest. There might be a little overlap between versicolor and virginica, but not much. This suggests species is a strong determinant of sepal length. If we saw overlapping boxes and medians close together, that would indicate species doesn't have much effect (at least on sepal length).

For numeric-categorical pairs, besides boxplots, one could also compare density plots of the numeric variable for each category (overlayed or faceted). For instance, overlaying density curves of sepal length for each species could also show the separation. However, with more categories, boxplots are usually clearer.

Example (categorical-categorical): Suppose we had two categorical variables, like if iris had another factor (say, a categorical size class for each flower). Although iris doesn't, let's demonstrate conceptually with a different example or a quick simulation. We will simulate a categorical variable for iris for illustration:

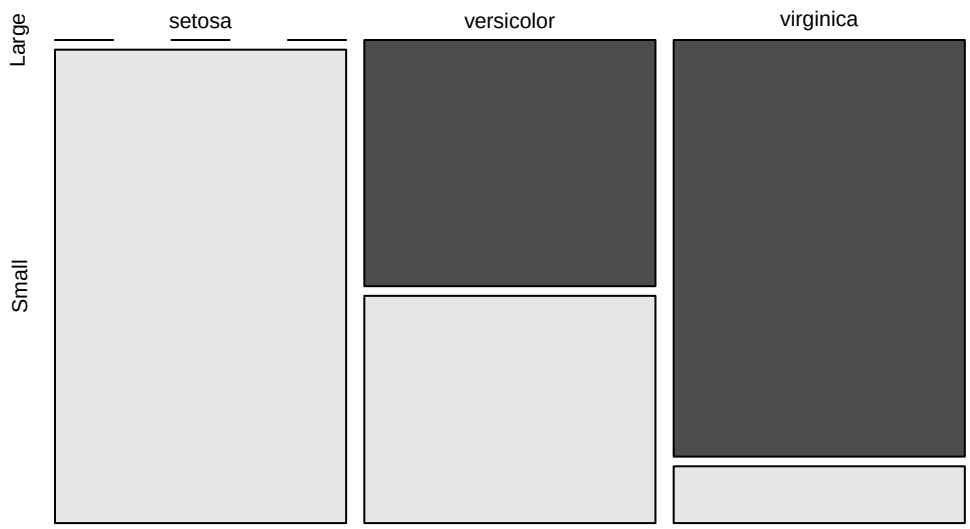
```
# (For demonstration) Create a categorical size class based on Sepal.Length
iris$SizeClass <- ifelse(iris$Sepal.Length > 5.8, "Large", "Small")
table(iris$Species, iris$SizeClass)
```

```
##
##              Large Small
##   setosa         0    50
##   versicolor    26    24
##   virginica     44     6
```

This code creates a new factor `SizeClass` labeling each flower as "Large" if Sepal.Length is above the overall mean (5.8) or "Small" if below. The `table()` will cross-tabulate species by size class, showing how many large vs small flowers each species has. We'd likely see setosa mostly classified as small (since setosa flowers are generally shorter), while virginica might be mostly large. This two-way table is the primary EDA tool for two categorical variables. If this were real data, we might compute row or column percentages to see, for example, what fraction of large flowers are each species.

A **mosaic plot** can visualize a contingency table:

```
# Mosaic plot for Species vs SizeClass
mosaicplot(table(iris$Species, iris$SizeClass),
           main="Mosaic of Species vs SizeClass", color=TRUE)
```



This will produce a mosaic where the area of the tiles is proportional to counts. It's not needed for our simulated case with such clear separation, but mosaic plots are useful in more general cases.

In an actual dataset, say we have **Titanic survival data** (with variables like Passenger Class, Sex, Survived), a contingency table of Sex vs Survived would show (for example) that a higher proportion of females survived than males. EDA of two categoricals often immediately highlights such patterns, which are candidates for further analysis or modeling (and indeed in Titanic, "Sex" is a strong predictor of survival).

Summary of bivariate EDA: At this stage, we are answering questions like: *Do changes in one variable correlate with changes in another? Does an outcome differ by group? Are two categorical factors independent or related?* We use plots to visualize these relationships and stats (like correlations or group means) to summarize them. Bivariate analysis can reveal simple relationships (like a linear correlation or a big mean difference) or more complex ones (interactions, nonlinear shapes) that hint you might need to consider more variables together, which brings us to multivariate exploration.

5.3 Multivariate Analysis

Multivariate EDA extends the exploration to three or more variables simultaneously. This is where things can get complex, because human eyes struggle to directly visualize beyond 2-D or 3-D. However, there are strategies to explore high-dimensional data:

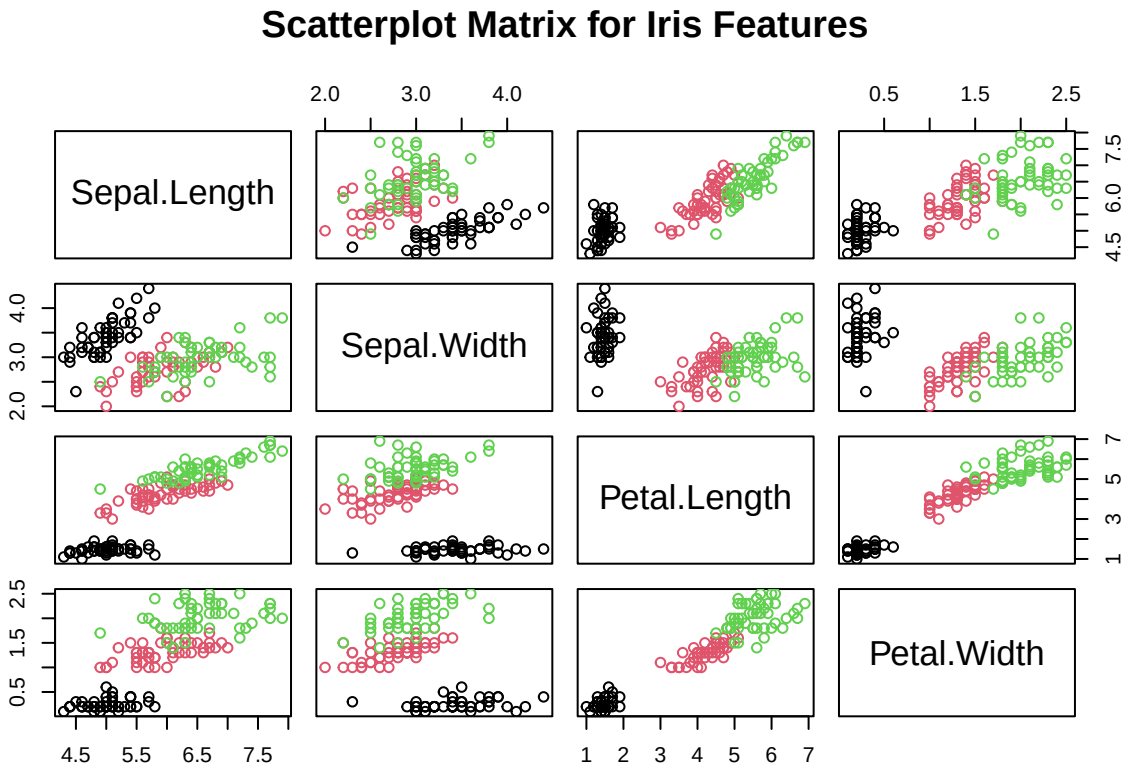
- **Scatterplot Matrices:** A scatterplot matrix (pairs plot) is a grid of scatterplots for many numeric variables. Each cell is a scatter of one variable vs another. This is a brute-force way to see all pairwise relationships at once, and perhaps notice a combination of variables that separate groups.
- **Color/Shape Encoding:** We already saw an example when we colored the petal scatterplot by species. By encoding a third variable as color or marker shape, you can embed a third dimension of information in a 2D plot. Similarly, using size of points or facets (subplots) are other ways.

- **Dimensionality Reduction:** Techniques like **Principal Component Analysis (PCA)** can reduce many variables into a few composite variables (principal components) that capture most of the variance. Plotting the first two principal components can show clustering or separation that involves multiple original variables. (This is a bit advanced, but good to mention.)
- **Clustering and Grouping:** Running clustering algorithms (k-means, hierarchical clustering) is another exploratory technique to see if data naturally groups into clusters in a multivariate way. Even without formal clustering, one can compute a distance matrix and visualize it (e.g., heatmap).
- **Correlation Heatmaps:** For many numeric variables, a colored correlation matrix (heatmap) can quickly show which variables move together in a dataset.

Let's illustrate a couple of these with iris:

Scatterplot Matrix: R's base function `pairs()` is handy for a quick look at all pairwise scatterplots.

```
# Scatterplot matrix for all numeric variables in iris
pairs(iris[ , 1:4], main="Scatterplot Matrix for Iris Features",
      col=iris$Species)
```



This plots every combination of the first 4 columns (which are the numeric measurements). We pass `col=iris$Species` to color points by species (as before, the factor will be coerced to integers 1,2,3 with a default palette, giving each species a different color). The result is a 4×4 grid of plots. On the diagonal you might see variable names or histograms; off-diagonals are scatterplots of one variable vs another. From this matrix, we can absorb a lot: We'll see which pairs have strong relationships (the petal length vs petal width cell shows a tight correlation), which are weak (sepal width vs any other has diffuse clouds), and also how species are distributed in each projection (e.g., setosa cluster is separated in many plots). This helps identify multivariate interactions – for example, maybe no single variable cleanly separates two categories,

but a combination does. In iris, petal length and width together separate species almost perfectly (versicolor and virginica overlap a bit in one dimension but less so in two dimensions).

Correlation Matrix: We can compute and view correlations among all numeric variables:

```
cor_matrix <- cor(iris[ , 1:4])
cor_matrix
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
## Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
## Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
## Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
```

This will output a 4x4 matrix of correlations. For iris, as noted, petal length & width ~0.96, sepal length & petal length ~0.87, sepal length & petal width ~~0.82, sepal width has low correlations with others (~~ -0.12 with length, -0.37 with petal width, etc.). We could visualize this with a **corrplot** (from the `corrplot` package) or just note which are high. If you have many variables, a colored heatmap using `corrplot` or `ggplot2` can highlight groups of variables that correlate (indicative of underlying factors).

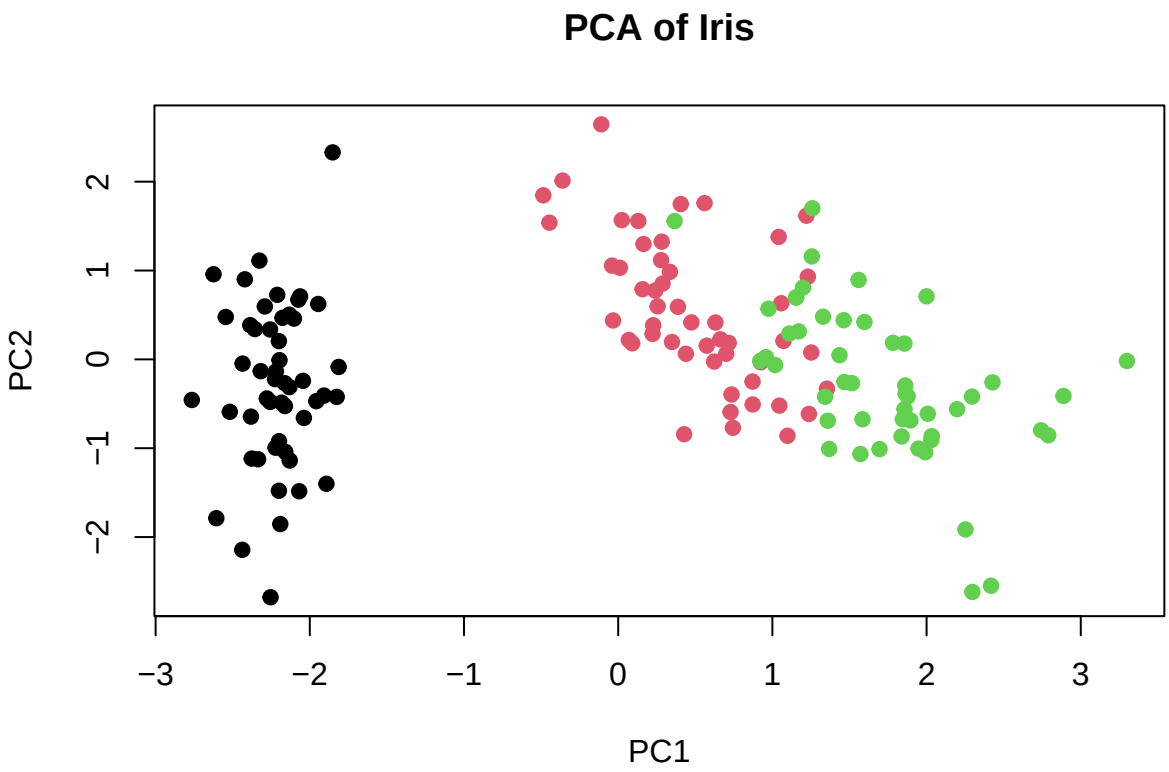
Finding multivariate patterns: Often, multivariate EDA is about seeing the **big picture structure**. In iris, because we have a target (species), multivariate EDA shows that a combination of two features (petal length & width) could almost perfectly discriminate species – which hints at good predictive features for classification. In a dataset without a predefined grouping, you might do something like PCA to see if data clusters naturally. For example:

```
# Principal Component Analysis on iris features
iris_pca <- prcomp(iris[ , 1:4], scale. = TRUE)
summary(iris_pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4
## Standard deviation     1.7084 0.9560 0.38309 0.14393
## Proportion of Variance 0.7296 0.2285 0.03669 0.00518
## Cumulative Proportion  0.7296 0.9581 0.99482 1.00000
```

This would tell us how much variance the first couple of principal components explain. We could then plot:

```
# Plot first two principal components
pca_scores <- iris_pca$x
plot(pca_scores[,1], pca_scores[,2],
     col=iris$Species, pch=19,
     xlab="PC1", ylab="PC2", main="PCA of Iris")
```



This is effectively another way to see multivariate separation – indeed iris data's PC1 vs PC2 will show three distinct clusters corresponding to species (a famous result of Fisher's iris data).

Another angle: if we had a dataset with, say, multiple clinical measurements on patients, a PCA or cluster dendrogram might reveal subgroups of patients with similar profiles – something you wouldn't see by looking at any single variable.

Interpreting multivariate EDA: It can be challenging; often we rely on domain knowledge and iterative exploration. Key points to consider:

- Are there **outliers in multivariate space**? (Points that might not be outliers in any single variable but are unusual combinations – e.g., someone with an unusual combination of age, income, and education.)
- Do multiple variables together show a trend that wasn't visible in univariate plots? (Perhaps two variables have a nonlinear relationship that only becomes apparent when plotted together.)
- Is there **multicollinearity**? (Many variables carrying the same information, indicated by high correlations – EDA might suggest dropping or combining them.)
- Can we **reduce the dimensionality** without much information loss? (If the first two PCs explain a high percentage of variance, it means the data effectively lives in a lower-dimensional space, which is good to know for modeling.)

Finally, EDA is an **iterative, non-linear process**. We described univariate → bivariate → multivariate as a progression for pedagogical reasons, but in practice you bounce back and forth. You might do some univariate analysis, notice something that makes you try a bivariate plot, which then leads you to examine another variable's distribution, and so on. The lines blur: for instance, a scatterplot with color grouping is already a multivariate visualization (2 numeric + 1 categorical). The important thing is to **ask lots of questions** and use the appropriate tools to answer them. EDA is like detective work: you form suspicions

("I wonder if X relates to Y?"), then look for evidence in the data (plots, stats), which might lead to new suspicions.

In the next section, we will put all of this into practice with a concrete case study. You'll get to perform your own EDA on a real dataset and interpret the findings.