BAR Unit 1 V3.docx

Business analytics using R_MBA_2

Business analytics using R_MBA_2

ATLAS SkillTech University

Document Details

Submission ID

trn:oid:::3618:127197658

Submission Date

Jan 30, 2026, 4:01 PM GMT+5:30

Download Date

Feb 2, 2026, 10:27 AM GMT+5:30

File Name

BAR Unit 1 V3.docx

File Size

232.9 KB

25 Pages

6,403 Words

37,590 Characters

4% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text
- Small Matches (less than 10 words)

Match Groups

20 Not Cited or Quoted 4%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

2% Internet sources
1% Publications
3% Submitted works (Student Papers)

Integrity Flags

1 Integrity Flag for Review

Hidden Text
166 suspect characters on 6 pages
Text is altered to blend into the white background of the document.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **20** Not Cited or Quoted 4%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

2% Internet sources
1% Publications
3% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Submitted works	Queensland University of Technology on 2021-09-21 <1%
2	Submitted works	Intercollege on 2023-12-10 <1%
3	Internet	www.coursehero.com <1%
4	Submitted works	National Economics University on 2025-12-03 <1%
5	Submitted works	Kennedy High School on 2025-04-23 <1%
6	Submitted works	King's College on 2019-01-07 <1%
7	Submitted works	Purdue University on 2024-07-30 <1%
8	Submitted works	Utkal University on 2024-08-09 <1%
9	Internet	www.hashstudioz.com <1%
10	Publication	Alex Bradley, Richard J. E. James. "Web Scraping Using R", Advances in Methods a... <1%

11 Submitted works
Intercollege on 2024-04-01 <1%

12 Publication
Jim Lindell. "What are Big Data and Analytics?", Wiley, 2018 <1%

13 Submitted works
Goldsmiths' College on 2023-10-16 <1%

14 Submitted works
Trine University on 2024-10-23 <1%

15 Internet
diarioeconomico.sapo.pt <1%

16 Submitted works
University of Westminster on 2015-04-26 <1%

17 Internet
kannuruniversity.ac.in <1%

18 Internet
oboloo.com <1%

19 Internet
www.mdpi.com <1%

20 Internet
www.zakirhusaindelhicollege.ac.in <1%

Unit 1: Introduction to Data Analytics and R

Learning Outcome:

1. Describe the role of data analytics in modern decision-making and its key concepts through an introductory caselet and overview discussion.

2. Navigate the RStudio environment and explain the purpose of R as a tool for data analysis.

3. Demonstrate proficiency in writing and executing basic R commands, including syntax, comments, and assignment operations.

17 4. Differentiate between fundamental data types and data structures in R, including vectors, lists, and data frames.

5. Apply basic operations and functions in R to manipulate and summarize data using built-in data structures.

6. Construct and manipulate data using vectors, lists, and data frames for simple data analysis tasks.

7. Summarize key terminologies and concepts related to introductory R programming and data handling for revision and application in case-based learning.

Content

1.0 Introductory Caselet

1.1 Overview of Data Analytics

1.2 Introduction to R and RStudio

1.3 Basic R Syntax

1.4 Data Types and Structures

1.5 Vectors

1.6 Lists

1.7 Data Frames

8 1.8 Summary

1.9 Key Terms

1.10 Descriptive Questions

1.11 References

1.12 Case Study

1.0 Introductory Caselet

"Data Decisions at EcoMart"

EcoMart is a medium sized retailer of organic and enviro friendly household products. The Company The company has grown into several metropolitan cities over the last couple of years, however, management has seen its sales growth stall out. Although providing competitive prices and eco-friendly products, customer loyalty is decreasing, and the inventory turnover has been still unsteady.

The marketing department has a hunch that the problem is rooted in an alignment of customer preference and stocking decisions. To find out, the company's leaders opt to take a data-centric approach. The purpose is to leverage the power of data science to learn about customer purchase behaviour so as to maximise inventory management and improve overall operational efficiency.

But EcoMart already has data, and it isn't all in one place. The datasets consist of POS receipts, customer surveys/feedback, product reviews and seasonal sales trends. The ops team decides to recruit a junior data analyst, Priya with strong proficiency in R programming and data analysis.

These various datasets are to be combined and then to perform a preliminary review with R – this is Priya's first task. Using the RStudio she creates scripts and runs through them for data cleaning, variable transformation, rounding up summary statistics. As part of her review, she also uncovers trends in customer purchases — like peaks in the purchase of environmentally friendly cleaning products during the winter season or less than stellar repeat purchases for certain high-end items.

Then she reports her findings to management in basic data visualizations and descriptive summaries. Her observations inform marketing and ops teams on how to think a little differently about their promo strategy or restocking timeframes. Now, EcoMart is in a much better place to make decisions informed by data going forward.

This caselet serves as a means to underline the significance of mastering the basics of data analytics, and underscores the practical value of mastering R programming for solving authentic business challenges.

Critical Thinking Question:

How you think EcoMart could use investment into data analytics tools and skills to other areas of the business (aside from marketing and operations)?

1.1 Overview of Data Analytics

1.1.1 Definition and Scope of Data Analytics

Data analytics Data analytics is the process of examining data (to derive complex conclusions) and helps in drawing conclusions about information, with a goal to support decision-making. " It's the process of examining, cleaning, and modeling data using an assortment of statistical and computing tools." The intention is to derive patterns, correlations, and trends that may not be immediately evident. The essence of data analytics is simply turning data in to actionable insights.

How Big Data, Cloud Computing and AI comprise to expand the data analytics sphere Dramatically The domain of data analytics has been getting bigger exponentially ever since the onset of big data, cloud computing and AI. Enterprises today gather huge volumes of structured and unstructured data from sources like customer transactions, social media, IoT sensors, and enterprise systems. This information, when viewed in the right context, has an ability to uncover customer behavior patterns, enhance efficiency, predict future developments and save money among other things.

Data analysis comes in several shapes and sizes —from simple worksheet functions to complex, sophisticated formulae. It is compatible with both retrospective (historical data) and real-time applications (streaming analytics). It also applies to several fields, such as business, medicine, education, public policy and marketing.

It also includes data governance, ethical and privacy issues, as well as the visualization of data. Professionals in this area should be fluent not just in doing data analysis, but also communicating findings and drawing on data to make decisions. 5 2The field is thus multidisciplinary and needs one to have some knowledge of various disciplines like statistics, programming, subject related matter etc. From this discussion about DIKW Model we see that the first three layers i.e data,information and knowledge are well suited for automation and Human Independent Decision Making using Computer can be possible by means of genomic based computational data mining.We could also discuss here that knowledge discovery from genomic big-data (bioinformatics) requires a scientist who knows how a gene expression influences cell behavior.

Additional Points:

- Sources: Internal systems (ERP, CRM), external APIs, web scraping, open data stores.
- Tool and Technologies: R, Python, SQL, Tableau, Excel , Power BI, Hadoop, Spark.

• Skills Needed: Data wrangling, EDA (exploratory data analysis), statistical modeling and interpretation of results.

1.1.2 Importance of Data Analytics in Business

Data analysis has become a lynchpin of the modern business model. In a cut-throat competitive environment, where customer attitudes shift quickly and operational efficiency can turn the screws on profitability, data-startsWith a separation here.

driven decision-making has become essential. Data analytics helps companies understand their customer, streamline operations, improve product offerings and stay ahead of market trends.

Among all benefits, translating raw operational data into tangible insights is a key achievement of what data analytics in business can provide. For example, through sales data analysis, a company could find hot sale products, regional differences in demand and help it establish the price. Analytics also plays a role in supply chain management for demand forecasting, inventory optimization and route planning. There's even talent acquisition analytics, employee performance tracking and retention strategy development in HR.

Moreover, customer analytics enable businesses to emulate marketing campaigns, customize recommendations and enhance user experience. Companies such as Amazon and Netflix have incorporated data analytics into their business model, collecting customer data to suggest more precise product or content recommendations.

The other major : factor continues to be risk- Taking Out insurance and Performance The risk associated with taking out insurance that may not deliver on the expectations (warts and all) must also have beena major consideration in putting CSR silt Works up for sale. Predictive analytics is used by businesses to forecast financial risks, find fraudulent transactions, and determine one's credit-worthiness. Compliance and regulatory reporting Compliance and regulatory reports are a fact of life, particularly in heavily regulated industries such as finance or healthcare, so the more analytics can be applied to audits, required reports and all other transactions.

Moreover, data analytics fosters innovation. Analysis has never been more powerful Businesses have used data to develop new products and business models based on unthought of possibilities. Strategies are more evidence-based, which are less reliant on gut feeling.

Additional Points:

• Business Optimization: Pinpointing bottlenecks and further optimizing processes.

• Decision Making: Actionable dashboards and real-time analytics for agile management.

• Customer Satisfaction: Improving service with sentiment analysis and customer journey mapping.

• Competitive Intelligence: Providing an understanding about the competitive landscape of the industry.

4

1.1.3 Types of Data Analytics: Descriptive, Diagnostic, Predictive, Prescriptive

There are four primary types of data analytics and each serves a different purpose from one another, as well as escalating to more advanced languages and approaches.

Descriptive Analytics:

Descriptive analytics is the most basic type of analytics. It is a response to the question: “What happened?” This is the type of analysis which seeks to distill past performance in order to see what trends and patterns have emerged over time. These are such things as dashboards, reports and visualizations. Develop a descriptive dashboard: Monthly sales and customer acquisition charts, website traffic metrics, for example, are descriptive. Such insights give one a picture of performance, but do not explain the underlying reasons that produced such results.

Diagnostic Analytics:

Then comes the diagnostic analytics which answers, “Why did it happen?” This is a predictive analytics solution that does root cause analysis and does relationship between variables. This includes such methods as correlation analyses, drill-down reporting and statistical testing in search for the cause of plant underperformance. For example, if customer churn spiked in a given month, diagnostic analytics would investigate the service quality, price changes or external events that led to this behaviour.

Predictive Analytics:

Among the questions predictive analytics answers is, “What’s likely to happen?” It is historical data that can be employed to predict the future with statistical methods, machine learning algorithms and forecasting mechanisms. Typical use-cases are sales prediction, demand forecasting, risk scoring. For instance, banks adopt predictive modeling to predict loan default risk; e-commerce companies predict future purchase demand using historical purchase patterns.

Prescriptive Analytics:

Prescriptive analytics answers the question, “What do we have to do about it?” It even goes beyond forecasting to suggest what you should do in order to get the results that

you want. Optimization models, simulation and decision analysis methods are fundamental to such analytics. For example, in aviation industry prescriptive analytics is used for route planning and dynamic pricing. It assists enterprise in comparing possibilities and taking strategic decisions.

Additional Points:

• Value Add As you move from one type of analytics to other, more value is added - descriptive explains the past and preventiv shapes the future.

• Tools and Techniques: Regression analysis, classification algorithms, decision trees, Monte Carlo simulations; linear programming.

• Integration in Workflow: Good analytics approaches often use all four types together as an integrated decision-making workflow.

1.1.4 Applications of Data Analytics Across Domains

Data analysis isn't isolated to one industry or division; rather, it is a game changer in many

array of domains. Here’s a look at what it means for various sectors:

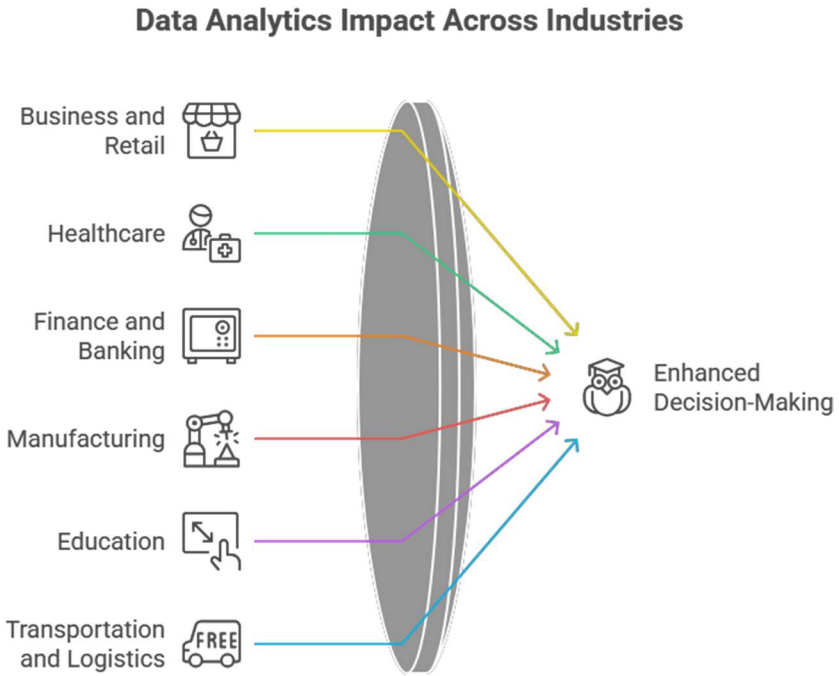

Figure 1.1

Business and Retail:

In retail, data analytics as applied to customer segmentation and demand forecasting, inventory and supply level management and marketing personalization. Retailers are using point-of-sale data, customer loyalty information and online activity to figure out shoppers' habits and refine their sales strategies.

Companies such as Walmart and Target apply analytics to supply-chain optimization and layout planning.

Healthcare:

Analytics allows healthcare providers to enhance patient results, cut costs and work more efficiently. Predictive analytics is useful in identifying high-risk patients and optimizing treatments, reducing avoidable readmission. "Big data," including electronic health records (EHRs), wearables, and medical imaging, is analyzed to facilitate diagnostics and personalized medicine.

Finance and Banking:

Banks and financial services incorporate analytics for credit risk modeling, fraud detection, portfolio optimization, and regulatory compliance. Algorithms are employed to spot suspicious transactions, determine who is eligible for a loan and identify investment products. For automated trading and market surveillance, real-time analytics also comes into play.

Manufacturing:

In the manufacturing domain, analytics is critical for quality control, predictive maintenance and process optimization. Sensor-generated data from machinery help predict failures before they happen – they can pre-empt downtime. Minimum amount of waste is applied by means of data on material flow, production speed, and defect control.

Education:

Educators use learning analytics to monitor the progress of students, personalize instruction and make predictions about at-risk youth. They rely on data for projecting enrollment, allocating resources and assessing how well programs are doing.

Transportation and Logistics:

Route optimization, fleet management and fuel efficiency analysis are some of the use cases enabled by data analytics. Logistics companies rely on data to schedule deliveries, track vehicle wellness and streamline warehousing.

Government and Public Policy:

Governments use the demographic information, financial statistics, and how services are being utilized to create policies that work. Analysis can aid in predicting crime, managing traffic, responding to disasters and planning budgets.

Sports and Entertainment:

More data is available for players, teams and broadcasters to analyse performance, fine-tune training, or improve viewer experience. Analytics in sports also informs in-game tactics.

Additional Points:

• Cross-domain Innovation: Practices applied to one field are often integrated into others, e.g., predictive maintenance in the aerospace being employed in telecom networks.

• Public Sector Impact: Open data opens up the potential for greater use of analytics to provide visibility and efficiency.

• Ethical Issues: With the advance of analytics, ethical considerations in data privacy, consent, and fairness of algorithms become very critical.

1.2 Introduction to R and RStudio

1.2.1 Features of R as a Statistical Tool

3 R is a language and environment for statistical computing and graphics. R, created by statisticians Ross Ihaka and Robert Gentleman in the early 1990s, has become one of the most popular tools used in data science, analytics and academic research largely because it's perfect for a range of tasks and is incredibly flexible.

1 A strength of R is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. The architecture of R is built to be very extensible, so individuals can write their own functions or use ones from third party packages that accomplish particular statistical tasks. There are thousands of such packages available through this CRAN (Comprehensive R Archive Network) repository that spans almost every field of data analysis.

Its high quality graphical abilities is another advantage over R. Users can produce professional-quality plots, including complex charts such as densitometry curves and plots, with both base R functions and an array of advanced packages. They are visual tools that facilitate pattern discovery and communication of insights.

R is also cross platform, it works on Windows, MacOS and Linux without differences in its abilities. Furthermore, R has support for so-called data wrangling making it possible to import, cleanse, transform and reshape data from a wide range of sources and formats such as from CSV, Excel, databases and web APIs.

Moreover, R fosters reproducible research. Software such as R Markdown enables analysts and researchers to merge code, output and narrative text within a single file which in turn facilitates transparency and sharing of work. Its support for distributed version control systems such as Git encourages collaboration.

Additional Points:

• Object-oriented programming — R uses S3 and S4 methods for creating and maintaining complex data structures.

• Memory Management: R operates mostly in-memory, which is good for data analysis but may need to be optimized when working with large data sets.

• Community Support: There is a large community of developers, statisticians and researchers who contribute

to the growth and knowledge base of R.

1.2.2 RStudio Interface and Components

RStudio is an integrated development environment (IDE) for R that helps users be more productive with the R language by providing a user friendly, ready-to-use interface and useful coding, visualisation and document generation features. It comes in free and commercial editions, and supports Windows, macOS, and Linux.

The RStudio interface is comprised of four primary panes, each performing different tasks:

13 Source Pane (Top-Left) Here you write, edit, and save R scripts, R Markdown documents, or any other source file. It provides multi-tabbed editing, syntax highlighting and code-folding.

Console Pane (Bottom-Left): This pane runs R commands in interactive mode. Code can be run directly, and results will be displayed right away, making it helpful when you want to test small code snippets and do exploratory work.

Environment/History Pane (Top-Right): The Environment tab shows objects and variables currently loaded in memory, e.g. datasets, user-defined functions etc. The

History tab Onmacadmin records each commands to allow the user recalling and modification.

Files/Plots/Packages/Help/Viewer Pane (Bottom-Right) : This area, divided into multiple tabs enables file control and plotting, package management actions, reader help.

o See and work with files in the working directory

o Output apart from our plots that were generated in the session.

o Installing, Loading and Managing R Packages

o Access R help files (including datasets, functions, etc.) from the Help tab

• Use Viewer to render HTML widgets and Shiny apps.

RStudio also comes with neat features like code auto completion, syntax checking and keybindings that speeds things up. It's compatible with project-based workflows where you can work on your data, scripts and outputs in an organized file structure. Features like integrated terminal support, Git version control tools and real-time code execution make RStudio a must-have tool for everyone from beginners to professionals.

Additional Points:

- Custom Themes: Customize the app's appearance and fonts to increase readability.
- Job Management: Execute long-running processes as a background job without blocking the console.

• Plugins and Extensions: RStudio allows to install additional plugins for it so that one could use add-ons like data viewer or code formatter.

1.2.3 Installing and Managing Packages

16 One of the greatest things about R is its package community. Packages are a way of grouping R functions, data and compiled code in order to perform specific tasks. If you are using R for data analysis at some point you will need to install and load stuff.

10 To install a package from CRAN, the most straightforward is to use base R function install. packages("package_name"). After installation, package_name will need to be loaded into the active R session using the function library(package_name). For instance, to visualize the dataset with ggplot2, the user could execute:

install. packages("ggplot2") library(ggplot2)

And thanks to RStudio, creating this response is even easier with a visual interface. Under the "Packages" tab, users can view installed packages and update them or install

fresh ones by searching for them and clicking to add. he also lists what packages are currently being loaded and their version.

Packages can also be installed from other repositories and sources:

• Bioconductor: For bioinformatics-related packages.

• GitHub: For devs you can install packages from GitHub repos with the devtools or remotes package using functions like install_github("username/repo").

R also gives users the possibility to handle package libraries residing in their own directories, which is useful if you need multiple versions of a package at the same time or for dealing with permission problems on shared systems. Packages can be updated by running update. packages(), and unused packages can be removed with remove. packages("package_name").

Below is an example where One Package depends on another, and so forth. R does this when you install it, but you may run into version conflicts in a multiuser environment. Tools such as renv and packrat can help in such scenarios by managing package environments, locking versions and promoting reproducibility.

Additional Points:

•CRAN Mirrors: You can now choose a CRAN mirror to get faster or more reliable downloads.

• Startup Scripts: The. Rprofile to automatically load commonly-used packages at startup.

• Checking for Conflicts: conflicted package is useful in addressing the conflicts of function names between loaded packages.

1.3 Basic R Syntax

1.3.1 R Console, Scripts, and Comments

R basics To start with, the most important thing to consider while working with R is where you write and run the R-code. R has a console based interface and script files interacting with the language, which means it is useful for both interactive work and writing programs.

The R Console is a place where users can type in single lines of R code and see the output right away. This is great for rapid calculations and exploring the code before writing any kind of serious code. The R console evaluates at once the command you type, and returns a result.

Scripts, by contrast, are text files with a. R extension that has a lot of R code. They are used to type, save, and run longer and more structured programs. Scripts facilitate re-

use of code, project organization, and sharing of analysis pipelines. The coding is done in the Source pane of RStudio and code lines or blocks can be executed with a keyboard shortcut by users.

Comments in R are preceded by the # symbol and are not interpreted while running. They are important in documenting what a block of code is doing, annotating parts of code, and making easier for future readers (including the original author) to read. Comment in code, is as crucial for shared projects and reproducing evidence. Note if prefers repeated #'s for multi-line commenting (it doesn't have a dedicated block comment as do some other programming languages).

Additional Points:

• Execution: To execute code in RStudio press Ctrl + Enter (Windows) or Cmd + Enter (Mac).

• Script Templates: Use RStudio as an R Markdown editor, with comprehensive code and (optional) narrative editing inside one document.

• Working Directory: Commands like getwd() and setwd() are useful to keep file paths in check, since when you're executing a script, it's common to read and\or save files.

1.3.2 Variables and Assignment Operators

R variables are symbolic references to data or values in memory. They are objects to which the result of a computation can be assigned, objects which data structures like vectors or dataframes can be written into and used as arguments in function calls. R is dynamically typed (the type of a variable is inferred when an assignment is made and may change from one use to the next).

In R, the most common assignment operator is

• Less than: =

• Less than or equal to: <=

Logical operations are important in data filtering, forming conditions for if statements, and program flows switching.

Logical Operators:

• AND: & (binary), && (unary for first of the two elements).

• OR: | (element-wise), || (first element)

• NOT:!

Example:

x <- 10

y <- 20

result <- (x < y) & (x! = 0) # Returns TRUE

Logical comparison are often performed in indexing and data subsetting. For example, you can subeset a dataset to only contain rows of data that correspond to a condition.

Additional Points:

• Operator Precedence: R evaluates expressions according to the rules of precedence (if in doubt, add parentheses).

• Type Coercion: In logical comparisons, R may coerce types if necessary e.g., comparing logical too numeric.

• Uses case studies to enhance discussion of the formal theory; * Focuses on applying sophisticated technology to real-world problems, such as transaction-based verification and power related modes in multiplication.

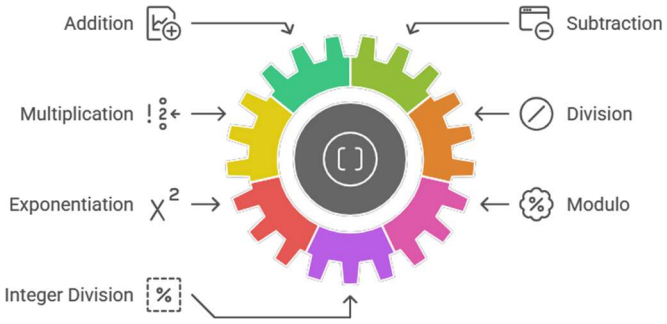

Figure 1.2

1.4 Data Types and Structures

1.4.1 Data.types—Numeric, Character, and Logical Types

Dynamically typed language means the type of variable is evaluated at run time according to the values given in it. The basicdatatypes in R are numeric, character and logical. These are ones you should know in order to do any kind of data manipulation or stat analysis in R.

The Numeric Data Type by itself comprises all number types, of both integer and real (decimal) class. By default, numbers are double-precision floating point in R. For example, = 5 or y 5 returns TRUE. Logical vectors can also be generated directly like c(TRUE, FALSE, TRUE).

Additional Points:

•Type Coercion: When you combine two different data types in R, it will automatically coerce the two vectors to the same type. e.g., combining numeric and character yields a character vector.

• NA Values: R uses NA to represent missing/undefined values and it works differently for each of the data types.

1.4.2 Overview of Data Structures in R

R has an extensive range of data structures that are necessary for creating, storing and processing data. While simple data types have associated values, a data structure may contain multiple associated values of different type or shape. These constructs are the bedrock for all data operations.

2 The vector is the simplest data structure in R, however it can hold elements of only one type (numeric, character or logical). Vectors are based components for complex structure components.

The former is merely a list and the latter one can vary in length, contain multiple types of elements. Lists can contain vectors, matrices, data frames and other lists. That makes them excellent in many cases for visually representing complex data such as model outputs or hierarchical data.

The matrix is a two dimensional data structure which needs to have elements of same types. It is a vector with dimension attributes in essence. Matrices play a central role in mathematical and statistical modeling.

Data frame is a widely used and versatile data structure in R, can store datasets of different types (numeric, character, factor etc.) as a table. Each record is a row, and each column contains the columns are variables.

The array generalizes the concept of matrices to a higher number of dimensions, making it an essential tool for handling multi-dimensional data like images or time-series.

Additional Points:

• Identify the Class: Use class() to find out what an object is.

• Structure Inspection: You can use the str() function to render a compact, human-readable summary of an

object's internal structure.

• Data Coercion: Functions like as. vector(), as. matrix(), or as. data. frame() will switch between wide form and long form.

7 ### 1.4.3 Creating and Accessing Vectors

7 Vectors are the basic data structures in R. They store elements of the same type and form the building blocks to most computations you need to perform when working with data in R. Vectors can have numeric, character or logical values and are defined using c(), which combines individual elements into a vector:

For example:

numeric_vector 15] pulls these elements that are more than 15.

Did You Know?

"In R, even a single number like x <- 5 is technically stored as a vector of length one. This uniform treatment of data simplifies internal operations and allows R to apply the same logic to both single values and sequences."

1.5 Vectors

1.5.1 Vector Operations

Vectors are the backbone of R, and their power comes from how adeptly R executes operations on vectors. In R, vector operations are done element-wise: This means if you do arithmetic, or logical operations with vectors they are performed with each of the elements separately. This is called vectorization, and it makes R code more concise and often faster than equivalent looping constructs in other programming languages.

Operations Perform arithmetic on the vectors including addition (+), subtraction (-), multiplication (*), division (/) and exponentiation (^). These are for use on a single vector or between two vectors of the same length. For instance:

x , = and .

Factors are very important in statistical models such as linear regression or ANOVA, because you have to represent categorical predictors using dummy variables internally. Factors are automatically converted by R to contrast coded variables during model fitting (and therefore, the interpretation of coefficients).

And some factors are at play in visualizations as well. In ggplot2, for instance, bar plots respect the level ordering of a factor when it comes to the x axis. This implies that controlling factor levels is important for the offical plot display settings.

Additional Points:

• Relabeling: relevel() or factor(..., levels=...) for who you use as a reference category in your model.

•Dropping Unused Levels: After subsetting unused levels can be dropped using droplevels().

• Labels: Factors enhance interpretability by linking humanreadable labels to data values, which is important for survey analysis.

"Activity: Vector Lab – Hands-On with Data Manipulation"

Title: Working with Vectors and Factors in Practice

In this activity, learners will create and manipulate vectors based on a fictional dataset containing student scores, gender, and grade levels. Tasks include performing arithmetic operations (e.g., calculating percentage scores), logical filtering (e.g., finding students scoring above a threshold), and converting variables into factors for analysis. They will also practice reordering factor levels and using functions like summary() and table() to generate descriptive insights. This exercise is designed to reinforce vector operations and the significance of categorical data handling in R, preparing learners for more advanced data structures and analyses.

1.6 Lists

1.6.1 Creating and Accessing Lists

A list in R is a flexible and powerful datastructure that can hold elements of different types and structure. It is the only object where a haphazard organization of other elements, be they atomic vectors, arrays or even lists, is possible. That's why such lists are especially useful for collecting related, but structurally different, data.

A list is constructed with the help of another function 'list()'. We can name each element down that list for easier referencing and comprehensibility. For example:

R> student c(list1, list2)

To apply a function to components of the list, have several functional programming tools as lapply(), sapply() and vapply(). These are great for creating structured output by cycling through elements.

Additional Points:

• Length and Names: If you want to count the components of a list use length() or to view or assign names is done by names().

• Flattening a List : function unlist() – to convert a list in to vector; often used when the structure is not necessary.

• Iteration: To iterate through lists, for loops or apply functions can be used with element-wise operations.

Did You Know?

"In R, most statistical model outputs—such as those from lm() or glm() functions—are stored as lists. These objects contain multiple components like coefficients, residuals, and fitted values, which can be accessed and analyzed using list operations."

1.7 Data Frames

1.7.1 Creating Data Frames

5 Data frames are fundamental to the structure of R and for storing data in R. Like a spreadsheet or SQL table, every column is a variable and each row is an observation. Data frames are so useful because the columns can be of different types (numeric, character, logical, factor) which is useful for many real data sets.

20 Data frames can also be created manually with the data. frame() function) with vectors of the same length. For example:

students (paragus822a iterable,retumstheainitemnumber the callable The returned value must be a number lt> height

Multiple rows or columns can be extracted using vector indexing:

students[c(1,3), c("name", "marks")]

Logical indexing allows data filtering. E.g. to obtain students who scored above 80 : students[students$marks > 80,]

1.7.2 Accessing Rows and Columns

When you are plotting or summarizing in functions, function can be passed on directly using the columns name. Knowing how to access with them is very important for many tasks you would do during data pre-processing, transformation and analysis.

Additional Points:

• Negative Indexing: The negative indices are used to exclude rows or columns, e.g. students[, -1] drops the first column.

• Subset Function: subset() is there to extract some of the elements from data frames based on conditions and select columns.

• Drop Argument: when drop=FALSE is set the result will be a keep tibble rather than simplify to a vector.

1.7.3 Modifying and Summarizing Data Frames

In R, data frames are mutable (i.e., we can change them). This permissiveness is critical for real-world data analysis, where datasets are commonly dirty and untidy.

To add a new column there are 2 ways: by $ notation or indexing it directly.

students$grade <- c("B", "A", "C")

To change an existing column, assign a new column:

students$marks <- students$marks + 5 To remove a column, assign NULL to it:

students$grade <- NULL

Rows can also be added with rbind() and removed through exclusion using negative indexing: students <- rbind(students, data. frame(name= "Divya",age= 23,gender="Female",marks=88)) students <- students[-1,] ## Removes the first row

Exploration with summary of data frames is important. Functions such as summary() return the descriptives of each variable:

summary(students)

Other useful functions include:

• mean(), median(), sd(): For summarizing numeric data

• table(): For frequency count(useful for categorical columns)

• nrow() and ncol(): Retrieve dimensionality • rowSums() and colSums(): Sum over the rows or columns, respectively

• colMeans(), rowSums(): Aggregation across rows/columns Summary with grouping can be done using aggregate() or tapply() : aggregate(marks ~ gender, data = students, FUN = mean)

Additional Points:

• Data Type Conversion: Use as. numeric(), as. character() functions to quote columns as required.

• Sorting: Sort data frames using order().

• Renaming Columns: Rename columns with names(students)[2] <- "student_age".

Knowledge Check 1

Choose the correct option:

1. Which function is used to examine the internal structure of a data frame?

a) head()

b) str()

c) sum()

d) scan()

2. How do you access the third row of a data frame named df?

15

a) df[3,]

b) df[, 3]

c) df[, "3"]

d) df$3

3. Which of the following adds a new column to a data frame?

a) df + newcol

b) df <- df + column

c) df$newcol <- value

d) add.column(df)

4. What will summary(df) return?

a) Graphical summary

b) Count of rows

c) Summary statistics

d) Column names

5. To remove the first row of a data frame, which code is correct?

a) df <- df[, -1]

b) df <- df[-1,]

c) remove(df[1,])

d) drop(df[1])

1.7.4 Importing and Exporting Data

In R, an analysis usually involves importing data from a source and exporting data back to the drive as well. R comes with a large number of functions that allows you to read data directly from files in various formats such as.CSV or Excel..266 R has many, many functions for reading dat ain multiple file formats including CSV, Excel and text files. An understanding of these functions will allow you to incorporate R into everyday data analysis tasks.

Importing Data:

CSV (/siːɛsˈviː/) is a popular file format used for data import. The function read. csv() can be used for these types of files:

data 25] # returns 30, 40

Name-based Indexing; works for named vectors, lists and data frames: data["column"] or list$the column

1.7.5 Indexing and Subsetting

Indexing can be used to access rows and columns for data frames:

df[,1] # first column df[1,] # first row

df[1, 2] #select the element of row with index and column with index.

The userfriendly way to use data as a filter is the subset() function:

filter(df, age > 25 & gender == "Male")

Additional Points:

• Drop Attribute: If you don't want to convert automatically your data frame to a vector when selecting only one column, you can use drop = FALSE.

• is. na() Use: Mix indexing with is. na() to deal with or remove missing values.

• which(): It gives the TRUE values position in the logic vector, works in pair with conditions.

Indexing and subsetting – These are essential operations in the toolbox all necessary for extracting information from subsets of data eg: meeting some condition or which lie at certain position.

1.9 Summary

ϖ Data analytics is the methodical analysis of data to reveal patterns and identify critical insights.

ϖ R is an open-source language and (oddly enough) the name of a pirate known also for statistical analysis, data manipulation, and visualization.

RStudio is an IDE that arranges and increases productivity.

ϖ Fundamental R syntax consists of commands, variables, operators, and comments that are used for writing the code in a way such that it is readable and efficient.

ϖ Typing in R: virtually all data are either numbers (numeric), text strings (character) or logical truth values.

6 ϖ Some common data structures in R are vectors, lists, matrices, data frames and factors.

ϖ Vectors are 1-dimensional arrays that support different arithmetic and logical computations.

ϖ Lists can store the values of any type, it is also heavily used in storing complicated outputs such as model results.

ϖ Data frames is a list, even if data frame has only one column.

ϖ R indexing and subsetting operations help manipulate data by selecting, filtering or modifying them with ease.

ϖ The operations of importing and exporting data are basic operations that makes it possible to integrate with external file and system.

ϖ With it's inbuilt functions and logical operators R proves to be useful in both small and large scale data analysis.

1.10 Key Terms

12 Data Analysis – The method of examining raw data with the purpose of drawing conclusions about that information.

Rugin/Stata – An interface between R and Stata.

RStudio – IDE for R with tools for coding, debugging and visualization of data.

Vector – A collection of homogeneous elements in unidimensional array.

List – A data format that can contain elements of varying types, and lengths.

Data Frame – It is a tabular data structure in which we can store data of different types.

Factor- A type of Data structure used for categorical variables (with or without order).

Indexing – The process of finding out elements from a data structure using position, names, or logicals.

Subsetting – Extracting a subset of the data based on conditions or structure.

read. csv() – A command in R which can import data from CSV files.

write. csv() – A function to write a data frame(.7) representing an R object as CSV files(.csv).

str() Function in R The str() function in R is helpful for outputting internal object structure.

1.11 Descriptive Questions

Data analytics 2)Define data analytics and provide example of operating or behavior with a digital system.

List some features of R that makes it ideal for statistical applications?

Compare and contrast vectors and lists in R by using examples.

Describe the forms of data analytics and where they are used in practice.

What is a data frame in R? How do you access or create elements in a data frame?

Write R code to show how you would create a factor variable and advantages of using them.

Explain the process of importing and exporting data with R. Include some code snippets.

Explain how we can use indexing and subsetting in R with examples.

1.12 References

1. Grolemund, G., & Wickham, H. (2016). *R for Data Science*. O'Reilly Media.
2. Kabacoff, R. I. (2015). *R in Action: Data Analysis and Graphics with R*. Manning Publications.
3. Matloff, N. (2011). *The Art of R Programming*. No Starch Press.
4. Verzani, J. (2014). Using R for Introductory Statistics. CRC Press.

5. Wickham, H. (2014). Advanced R. Chapman and Hall/CRC.

6. R Core Team (2023). R Language Definition. R Foundation for Statistical Computing.

Answers to Knowledge Check

Knowledge Check 1

1. b) str()
2. a) f[3,]
3. b) df$newcol <- value
4. c) Summary statistics
5. b) df <- df[-1,]

1.13 Case Study

Student Performance Analytics Using R

Background:

Data Science Department A college's department of Data Science has gathered information from a group of its students. The variables in the data set are student name, gender, age, department of study, performance in assignments and final exam results. We then want to look at that data and look for patterns in the student's progression, consider how influential (if any) continuous assessment is (assignments), get some stats on student performance which will inform academic decisions.

Dataset (Simulated for Practice):

students <- data.frame(

name = c("Anita","Ravi", "Sneha", "David","Fatima"),

gender = factor(c("Female", "Male", "Female", "Male", "Female")) age = c(21, 22, 20, 23, 21)YLeafxUtils test site : http://www.

department = c('CS', 'CS', 'IT', 'CS', 'IT'),

assignment_marks = c(88, 76, 90, 85, 95),

exam_score <- c(78, 82, 91, 74, 89)

)

Problema 1: Realice el calculo del rendimiento Total a) If the net performance of each year and check their cause by type of failure.

Task: Add a new column total_score which is the weighted sum of assignment and exam scores. Project: 40%, exam: 60%.

Solution:

students$total_score <- (0.4 * students$assignment_marks) + (0.6 * students$exam_score)

Explanations: This weighted score is used to normalize assessment according to institutional policy.

Problem 2 : Segment the students into grades Task: Create a variable. factor called grade based total_score:

• A: 85 and above

• B: 70–84

• C: Below 70

Solution:

students$grade <- cut(students$total_score,

breaks = c(-Inf, 69.99, 84.99, Inf), labels = c("C", "B", "A"),

right = TRUE)

Explanation: cut() function divides the range of continuous scores into intervals which are easier to analyse.

Problem 3: Derive the department-wise summary statistics Task: Average of total_score for each department.

Solution:

aggregate(students$total_score ~ department, data = students, FUN = mean)

Explanation: It allows to interpret which department has an average better student performance and serve to evaluate curriculum.

Reflective Questions

How do R data frames and factors make it easier to analyze academic performance?

How do scoring by weight affect student grading?

What would be other variables that can improve the quality of insights from this dataset?

What would you do to improve the interpretation of student performance with visualizations (e.g., bar plots, histograms)?

How could such a dataset inform academic counseling or intervention?

Conclusion

The case they examine here provides an example of how core R data structures – data frames, indexing, and factor creation and aggregation – can be extended to the context of real educational information. By modeling common workflows of academics, the learners develop a set of tools for acquiring and analyzing structured data streams. These are skills that easily translate into roles in the education, research, business, and government sectors. Students develop competence executing analytics as well as an appreciation for the factors that contribute to effective, data-driven decision making through problem solving and critical reflection.

BAR Unit 2 V3.docx

Business analytics using R_MBA_2

Business analytics using R_MBA_2

ATLAS SkillTech University

Document Details

Submission ID

trn:oid:::3618:127197659

Submission Date

Jan 30, 2026, 4:01 PM GMT+5:30

Download Date

Feb 2, 2026, 10:27 AM GMT+5:30

File Name

BAR Unit 2 V3.docx

File Size

149.4 KB

16 Pages

3,435 Words

18,522 Characters

4% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text
- Small Matches (less than 10 words)

Match Groups

9 Not Cited or Quoted 4%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

1% Internet sources
1% Publications
2% Submitted works (Student Papers)

Integrity Flags

1 Integrity Flag for Review

Hidden Text
133 suspect characters on 3 pages
Text is altered to blend into the white background of the document.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **9** Not Cited or Quoted 4%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

1% Internet sources
1% Publications
2% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1 Submitted works
Ibri College of Technology on 2018-02-18 <1%

2 Submitted works
Manipal University Jaipur Online on 2025-12-10 <1%

3 Internet
www.slideshare.net <1%

4 Internet
www.r-bloggers.com <1%

5 Submitted works
Manipal University Jaipur Online on 2024-12-15 <1%

6 Submitted works
Manipal University Jaipur Online on 2026-01-17 <1%

7 Publication
C. K. Dhaliwal, Poonam Rana, T. P. S. Brar. "Python Programming - A Step-by-Step ... <1%

8 Internet
www.commsvr.com <1%

Unit 2: Logical Reasoning with R

Learning Outcomes:

1. Explain the role of control structures in programming, particularly how loops and conditionals support automation and decision-making in R.
2. Differentiate between various types of loops (for, while, repeat) and conditionals (if, else, ifelse) and their appropriate use cases in R scripts.
3. Construct loop-based solutions to perform repetitive tasks such as data transformations, simulations, and iterative computations.
4. Implement conditional statements to control program logic based on dynamic inputs or data conditions.
5. Combine loops and conditionals to build efficient and robust R programs that mimic real-world decision-making processes.
6. Debug and optimize control structures by identifying common pitfalls like infinite loops and logical errors in conditional statements.
7. Apply control structures in a mini-project or case scenario to automate data operations and perform logical branching based on dataset values.

Content

2.0 Introductory Caselet

"Automating Efficiency at GreenData Labs"

GreenData Labs is a rapidly growing environmental, sustainability-data-focused analytics consulting firm. With a growing customer portfolio and an array of active projects, their data science team works with various datasets that run the gamut from carbon emissions to renewable energy channel utilization in different locations.

One of the junior analysts, Rina, had been working on a report intended for a client who runs solar power plants across several states. Our client needed an automated platform to sift through large solar generation panel output data sets, pinpoint underperforming installations and then triage them according to performance thresholds.

At first, Rina tried to analyze the data manually using filters and spreadsheets. But as the data grew, doing so took forever and was error-prone. On realizing the lack of efficiency, she found refuge in R programming to make her work easier.

Rina began by creating conditional statements that compared the output of each plant to what was expected, and labeled them as "Normal," "Below Average" or "Critical." Then she used loops to do the process automatically for thousands of rows on many files. For loop that checks each record, ie if it achieves such and such marks using if-else.

A couple of hours later, Rina had a script that could handle the whole dataset in a few seconds. It is not only a time saver but it also ensures the consistency and accuracy of the finished parts. Her team could then re-use and modify the code logic to even their other clients while developing a reusable automation framework.

This showcases the role of control structures, including loops and conditionals, that enable automation of manual tasks and inclusion of if/else logic in analytic pipelines. In applications to actual data, such abstractions are needed in order to address due to complexity, scale, and efficiency.

Critical Thinking Question:

What role do loops and conditionals play in turning tedious, manual data tasks into streamlined, scalable workflows programmed to run across a wide variety of sectors?

2.1 Loops

2.1.1 Introduction to Iterative Structures in R

Loops In programming, a type of control flow or iteration that can repeat a group of statements is called loop. Iterative constructs are a must if you're dealing with stuff that needs to be done automatically, over and over again, like in processing and analyzing data. R is a functional language and has multiple type of loop constructs such as for, while, repeat. These can be used to iterate over elements of a vector, matrix or data frame, or any other structure.

Yeltsin's meanati theist became descaled and briefcases January 6, 2007 ngs with Pictures Such iteration is required in many practical situations. Such as, summarizing every columns in a data set, reporting for multiple regions or departments etc. Without the use of loops, you would have to write out individual lines for each sentence/element – inefficient and prone to errors.

R is designed in a vectorized style, such that many tasks do not require explicit loops to compute when functions are employed with encapsulated iteration (e.g., apply(), lapply(), sapply()). But loops are still relevant for custom tasks, nonstandard operations, and complicated control flow statements when vectorized solutions don't exist or are less readable.

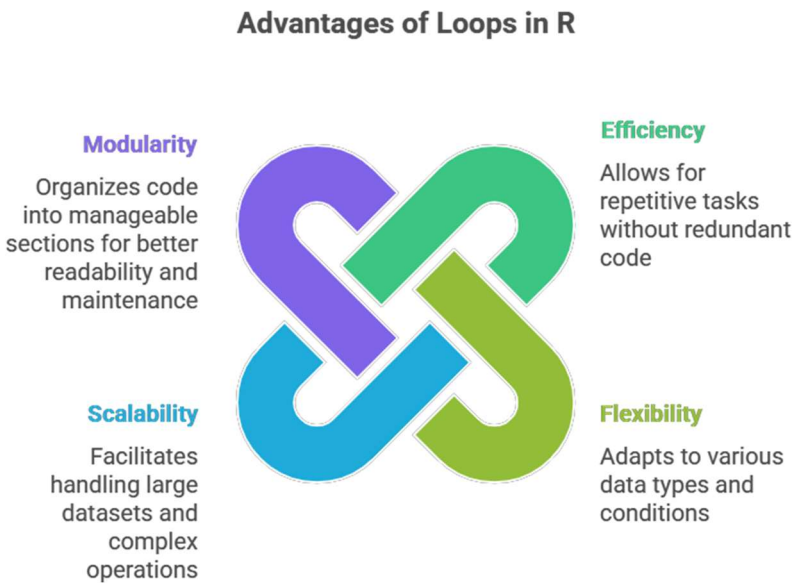

Figure 2.1

Advantages of using loops in R Some advantages to using loop in R:

• Efficiency: Don't repeat yourself by putting away-the-same-rails into loops.

• Dynamic Execution: They can be used to dynamically execute according to conditions or a data value.

• Scalability: After developing, the loop-based scripts are ready to increase the data size with only minor changes.

• Modularity: Loops can lead to cleaner scripts, especially in conjunction with user-defined functions.

The three most prominent types of loops in R have different use cases:

• for loops iterate over a sequence (e.g., vector, list, or index range).

• while loops execute until some logical condition is no longer satisfied.

• repeat loops keep running forever unless a stop condition is provided by the user.

Students must understand each of the loops, when it should be used, and how to manage its execution through break and next keywords. By the way, it is just as important to learn when not to use loops (in favor of vectorized functions or apply family functions) for writing efficient R code.

2.1.2 For Loops – Syntax and Applications

The for loop is universal in R because it is simple and easy to use. It's especially helpful if you already know how many times the loop has to run. This loop parallels over sequence elements, in each iteration of the loop it executes a block.

Syntax of a for loop:

for (variable in sequence) { #block of code to be executed inside the for loop}

}

Here it could be a list, vector, or any iterable. The loop variable gets the value of each item in our sequence, and only one at a time.

Example 1: Numbers from 1 to 5.

for (i in 1:5) {

print(i)

}

Example 2: Square of each element in a vector Suppose, you have the following numbers.

numbers 5) { break

}

}

Example 2: Search until some random value surpass threshold

```
repeat {
x 0.95) {
break
}
}
```

Use Cases:

- Evaluating a sequence of simulations until some stopping or exit rule is met
- User prompted repeatedly until valid input is given
- Watching for certain data conditions to occur
- Function-level retries or controlled tests

Key Characteristics:

- No initial state: You have to code up an exit condition yourself
- Well suited for open-ended loops with variable length
- Can spoof while behaviour with more complex logic

Best Practices:

- Make sure you add a condition to break on
- Be cautious when using this to avoid non-terminating execution
- In advanced applications, combine with tryCatch() to manage loop errors

While less common than for or while loops, the repeat loop plays a unique and important role when it really is not possible to say in advance how many iterations will be needed, and flexibility is crucial. They are convenient in the reactive and event-driven form of programming.

2.1.5 Breaking and Skipping Loops (break, next)

R uses control keywords break and next to dynamically control loop execution. These criterion provide a more primitive way to control the loop flow out or skipping some iterations.

break:

Begins the next iteration of the loop and jumps to its end.

Example:

```
for (i in 1:10) {
if (i == 5) { break
}
print(i)
}
```

This loop would print 1 to 4 and will exit because of i == 5.

next:

Jumps to the top of the loop without executing the current iteration.

Example:

```
for (i in 1:5) {
if (i == 3) { next
}
print(i)
}
```

This will print 1, 2, 4 and 5 (excluding the value when i is 3).

Use Cases:

• break is utilized to terminate the processing when a condition being sought after is discovered!

• next: Skips invalid or unwanted data points.

Best Practices:

• Avoiding this style or the overuse of break and next to make code harder to read

• use conditionality judiciously to make sure that skipped or aborted loops do not mess up program logic.

Such control statements offer more flexibility, enabling Loops to adapt to varying conditions during execution.

2.1.6 Practical Examples of Loops in Data Analysis

Loops are a powerful way to instrument this type of workflow in data analysis, due to the time it takes for manual processing and also when dealing with many files or large dataset. In the following paragraphs we show some examples to illustrate how for, while and repeat loops in R can simplify looking at data.

Example 1: calculate mean of columns in data frame We want to calculate the mean for each column.

4 data <- data. frame(a = 1:5, b = 6:10, c = 11:15) means <- numeric(ncol(data))

for (i in 1:ncol(data)) { means[i] <- mean(data[[i]])

}

Example 2: Row filtering using while

set.seed(1)

x <- runif(10) i <- 1

while(x[i] < 0.9 && i <= length(x)) { print(x[i])+\それを行いたい Mean :x off ！記載されて \#, #これもできません実行して Mean -paste("cluster ",(k+1)," : ",ID[p+1]-50) Mean[:,paste("cluster_",(range(C)+ giants))),extract_rock60,num(general_giants),flags_col =general_flags,coords_col=general_coords)そして、 +# More iteration on a converging Sample Id system_answer using + instead of comma:i <- i + 1

}

Example 3: Batch processing files

file_names <- c("file1. csv", "file2. csv", "file3. csv") results <- list()

for (f in file_names) { data <- read. csv(f)

results[[f]] <- summary(data)

}

Example 4: Trial simulation to reward

repeat {

trial <- sample(c("success", "fail"), 1) print(trial)

if (trial == "success") break

}

Example 5: Skipping NA values

values <- c(10, NA, 25, NA, 40)

for (i in values) { if (is. na(i)) next print(i * 2)

}

Did You Know?

"In R, loops can be combined with conditionals and functions to automate entire data pipelines—from cleaning to transformation and reporting—making them essential for reproducible, large-scale analysis workflows."

2.2 Conditionals

2.2.1 Introduction to Conditional Statements in R

If It is a futures and options policy, in that case an assignment of “Delivery” is allowed when andif the policy has not matured or has no lapsed but Delivery is prohibited if the cover holder threatens to fail.BoardEx3 Conditional statements are basic building block of any programming language, R or any other programming language. In data analysis and programming logic, conditionals are present so that scripts can make things happen when certain conditions are met, or else they will act in another way if the conditions are not fulfilled.

7 Conditional constructs In R, the fundamental conditional statements are if, if-else, nested if-else, and switch. These are the tools that help developers to create dynamic and responsive code which can respond to different data inputs, user instructions or feedback etc.

The Big Idea behind conditionals is checking logical expressions. Logical tests evaluate to TRUE or FALSE, and the outcome decides which code is run. For instance, let's say you would like to test if the number is positive/negative or that data frame column contains missing values and take some action.

Conditionals are a key player for logical operators. These include:

- ==: equal to
- !=: not equal to
- , =: comparison operators
- &: logical AND
- |: logical OR
- !: logical NOT

Besides simple decisions, conditional forms have many other uses. They’re important for things like filtering data, writing custom functions, running loops with conditional logic, attempting operations that might fail, or adjusting data or parameters based on

the surrounding context. Used in conjunction with loops and functions among other programming structures, conditionals serve as a foundation for writing intelligent programs.

Thus, mastering conditionals is a fundamental programming skill in R, particularly for data analysts who frequently deal with complex datasets which they want to filter, transform and report upon conditionally.

2.2.2 If Statement – Syntax and Usage

If statement is used in a program to make the decision – True and False branching according to certain condition.

7 if else nest in R is the basic conditional control. It is used to execute code on the basis of a condition. If it is not true, the code inside the if statement will be ignored.

Syntax:

if (condition) {

Code to perform if the conditional statements equals TRUE

}

2 Here condition should be a logic that results in either TRUE or FALSE. If the condition is TRUE, the block of code inside follows; if it's FALSE nothing happens.

Example 1:

x 5) {

print("x is greater than 5")

}

Here, since the condition x > 5 is True, therefore it should print message.

Example 2:

temperature 25) { print("It's a hot day")

}

Important Considerations:

• An if statement does not produce a value; rather, it governs the execution path.

• Code block must be included in braces { }if they span multiple lines.

• You actually don't have to use braces for single-line conditions, but I'd recommend getting into the habit of using them to make your code cleaner and less bug-prone.

Best Practices:

• Testing Always test your logical conditions to make sure they produce the correct TRUE/FALSE results.

• Don't use if statements with anything other than logical values (unless you can coerce).

The if statement is pertinent when you need to control the flow of your code and decide whether or not a block of code should execute based on one condition. It is enhanced when intertwined with other signals or blocked with more sophisticated modals.

2.2.3 If-Else Statement – Decision Making

if-else The if-else structure adds a block of statements to be executed when the condition is false. This structure provides a way for programs to execute some statements and not others, depending on whether expression is true or false.

Syntax:

if (condition) {

Code if TRUE

} else {

Code if FALSE

}

Example:

score = 50) { print("Pass")

} else { print("Fail")

}

In this case it is checking the valu e of score, if strScore is 50 or greater then print "Pass" else Print "Fail".

Use Cases:

• Thresholding mechanisms for making decisions (e.g., grade, price)

• Two-way classification (e.g., fraud) vs. non-fraud)

• Logical with tests that return level arguments or categories

Key Considerations:

• The else statement must reside on the next line after the closing brace of the if block, or it immediately follows that brace. Without giving it braces and with a new line, you might get some syntax errors.

• As if and else parts are also composed of arbitrary code.

•if-else restricts to decision between two alternatives. For larger than two conditions, you should use nested if-else or switch.

Best Practices:

• Indent and space to make it more legible.

• Where functionality might be confusing, make it a function and keep the logic simple.

The if-else statement is one of the most useful in any programming environment for performing simple classification logic and also when you need to do error-checking or validation.

2.2.4 Nested If-Else Conditions

Nested if-else statement is used in cases where a decision depends on multiple sequential conditions. The if-else statement is nested inside an additional if or else to check the further condition in this construct.

Syntax:

3

if (condition1) { #Block of code 1

} else if (condition2) { # Block of code 2

} else {

Default code block

}

Example: Grading System

score 90) { grade = 75) { grade = 60) { grade = 90) { category = 75) { category = 60) { print("Pass")

} else { print("Fail")

}

}

These samples demonstrate how conditionals are a fundamental part of your data workflows that can drive custom responses as well as logic-based outcomes.

2.3 Summary

ϖ Loops in R make the user to run the code again and again which automates repeated data processing.

ϖ The 106FOR loop is used to tell the computer that we know how many times we want our for loop to iterate through a vector, list or something similar.

• The while loop conditions itself to continue as long as a condition is true making it ideal for unknown number of iterations.

ϖ Note that the loop repetition is intended to be infinite, and you need to explicitly break out of it.

ϖ The break and next loop control keywords enable you to exercise a degree of control over iteration by exiting the body or skipping parts of loop constructs.

ϖ An enunciation if Appear Statement is then conditional statements, instrics the condition to be checked and execute different sequence of statement between true or false outcomes.

6

ϖ The if clause works in a way such that it runs a piece of code when some condition is true.

ϖ The ifelse statement allows two outcomes, one in the event that the condition is true and another one when it is false.

ϖ Nested if-else provide for multi¬level decision logic, which is frequently used in such classification or grading systems.

ϖ The switch() function is a better way of writing the if-else statements when we select on fixed, particular values.

ϖ Loops and conditionals frequently work together as the building blocks for more practical scripts operating on complex filtering, categorization, and automation of data.

ϖ Well-written, understandable and efficient control structures increase your productivity, reduce the likelihood of errors (bugs) in your code and make it easier to read.

2.4 Key Terms

Loop – A control structure that cycles the same block of code several times.

For Loop - When you know the size of element sequence to iterate through.

While Loop – Executes as long as a condition is true.

A repeat loop - Loops until a break point is encountered.

Break : Exit the loop without completing it as usual.

Next – Skips over the current iteration, then continues to the next one.

1 If statements – Execute code if a condition is true.

If Statement – Runs code if a condition is true.

If-Else Statement – one block is executed if a condition is true the other is executed for false.

Nested If-Else – Conditional checks placed inside each other.

1 Switch Statement – Selects one of many code blocks to be executed.

Logical Operator – The symbols used in conditions (==, >, &, |) that results in TRUE or FALSE.

2.5 Descriptive Questions

Discuss loops in R. How can they be used for the purpose of data analysis?

Compare and contrast for, while,and repeat loops including examples.

Explain the significance of control statements break and next in loop constructs.

How would you explain the use of conditionals in R to someone with a good background in programming, ideally, using some examples as well?

Explain if, if-else and nested if-else statement in R.

What is the switch() function? When to use it over if-else?

Demonstrate the combination of repetition and selection in a realistic data problem.

Write a script in R with the code that uses conditional statements to divide test scores into 4 levels of grades.

2.6 References

1. Grolemund, G., & Wickham, H. (2016). R for Data Science. O'Reilly Media.

2. Kabacoff, R. I. (2015). R in Action: Data Analysis and Graphics with R. Manning Publications.

3. Matloff, N. (2011). The Art of R Programming. No Starch Press.

4. Verzani, J. (2014). Using R for Introductory Statistics. CRC Press.

5. Wickham, H. (2014). Advanced R. Chapman and Hall/CRC.

6. R Core Team (2023). R Language Definition. R Foundation for Statistical Computing.

Answers to Knowledge Check

Knowledge Check 1

1. c) Tests condition
2. d) nested if
3. b) You need fixed outcomes
4. d) next
5. a) if

2.7 Case Study

Automating Student Assessment Reports Using Loops and Conditionals in

Background:

A department in a university logs the assessment data of its students in a tabular dataset. Every student has been given marks for assignments, mid term and final exams. The faculty would like to automate the scoring, grading, and alerting processes for students who may be failing.

The dataset captures the marks and names of students, scores of their assignments mid-term exam results and final exam grades. We want to test each student, give grades based on the total score and flag students who needs special attention because of their scores.

Sample Data:

students = 85) { (students$grade[i] = 70) { (students$grade[i] = 50) { (students$grade[i] = 70) { students$grade[i] = 50) { students$grade[i] <- "C"

} else {

students$grade[i] <- "D"

}

}

Now grades are given for a student's work in class.

Problem 3 – Identify At-Risk Students using If-else and Loop

You have to add a new column status for that in this scenario we will assign grade "D" students as "At Risk" and others as "OK".

Solution:

students$status <- character(nrow(students)) for(i in 1:nrow(students)){

students$status[i] <- "At Risk" if (students$grade[i] == "D") {

} else {

students$status[i] <- "OK"

}

}

This helps in the early identification of those students who might need extra help academically.

Final Data Frame Output

print(students)

This will print the updated data frame with overall score, grade and status for all the students.

Reflective Questions

Explain how loops saved time in calculating the student measures manually.

How to generalize the conditional statement if we have more than 2 catergories/grading rules?

Is it possible to get the same power out of vectorized functions? What are the trade-offs?

How would you adapt the code to an 1000 students dataset?

System of Automatic Report Generation Why use R to automate the process?

Conclusion

The case teaches how R loops and conditionals can be used to solve concrete tasks (resource accounting in academic reports). Automating the total scores calculation, letter grade assignment and academic risk flagging allows the faculty time to focus on the student paper comments and delivers assessment consistency. The method exhibits scalability and it can be adjusted to work with more complicated criterion. A thorough understanding of control structures gives analysts the tools to develop dynamic, data-driven applications that serve institutional and organizational needs.

BAR Unit 3 V3.docx

Business analytics using R_MBA_2

Business analytics using R_MBA_2

ATLAS SkillTech University

Document Details

Submission ID

trn:oid:::3618:127196238

Submission Date

Jan 30, 2026, 4:01 PM GMT+5:30

Download Date

Feb 2, 2026, 10:28 AM GMT+5:30

File Name

BAR Unit 3 V3.docx

File Size

84.6 KB

23 Pages

5,122 Words

29,043 Characters

1% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text
- Small Matches (less than 10 words)

Match Groups

6 Not Cited or Quoted 1%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

1% Internet sources
0% Publications
0% Submitted works (Student Papers)

Integrity Flags

1 Integrity Flag for Review

Hidden Text
214 suspect characters on 6 pages
Text is altered to blend into the white background of the document.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **6** Not Cited or Quoted 1%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

1% Internet sources
0% Publications
0% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	ddceutkal.ac.in <1%
2	Internet	forestforesight.atlassian.net <1%
3	Internet	www.jbp.org.br <1%
4	Internet	www.coursehero.com <1%
5	Internet	archive.org <1%
6	Internet	r.qcbs.ca <1%

Unit 3: Functions and Data Input in R

Learning Outcomes:

1. Explain the concept, structure, and purpose of user-defined functions in R, including parameters and return values.

2. Write and execute custom functions to modularize code and improve reusability in R programming.

3. Demonstrate how to capture and process user input through interactive prompts in R scripts.

4. Import data into R from various sources such as CSV files, Excel spreadsheets, and web-based resources.

5. Differentiate between various data import functions and choose appropriate methods based on data format and source.

6. Apply functions and user inputs in a combined workflow to create dynamic and interactive data analysis processes.

7. Troubleshoot and handle errors that may arise during user input handling and data import operations.

Content:

3.0 Introductory Caselet

"From Manual to Modular – Automating Survey Analysis at EduInsights"

EduInsights is a research company which carries out education surveys across India to assess student and teacher performance along with infrastructure facilities available at schools. They gather immense amounts of data from lots of different schools each quarter in forms of CSV files, Excel sheets or sometimes directly from web-forms.

Previously, the data analysis team was responsible for manually cleaning up those files and processing them to run performance indicators and generate reports that stakeholders could use. This was a slow, error-prone and unscalable way of working. As the organization scaled, the team began to see a need for adopting a more modular and interactive fasioned solution.

A senior analyst, Meera had an idea for building some user-defined functions in R that could automate some of the major analysis steps: data cleaning and preparation, calculating scores, summarizing the results, and producing reports. She was one of the first to use user input to allow scripts to be dynamic. For instance, a user might submit the name of the dataset and select a specific performance measure to be calculated, or specify the format in which results are desired.

Meera also generalised the process of importing data by writing functions that allowed multiple filetypes, error checking and handling, and formatting with better transparencies of how data was read in. Now, instead of writing and rewriting entire programs for each database in turn, analysts would merely invoke pre-defined functions and provide the right inputs when asked.

The conversion not only was a time saver, but also did not rely on any one analyst. Members of the team could simply re-use and extend functions, they could easily process new sources of data, and get to spend more time on interpreting affects in a strategy rather than writing repetitive code. As EduInsights continued to add new data projects, this function-our-your-users-in relation-to-you coding approach became a staple of its analytics architecture.

Critical Thinking Question:

How does it make our data analysis projects more scalable and collaborative to work with functions, user input, and consistent standard ways of importing data?

3.1 Functions

3.1.1 Introduction to Functions in R

Functions are your friends Functions are the most basic form of object in R. They allow for reusability, modularity, and abstraction. In R, a function is nothing but a set of cohesive commands with given name that does a well-defined task. Functions regulare repetition, minimize errors, and solve complex tasks by abstracting logic into manageable and reusable chunks.

R is a functional vectorized language, which means that not only can you make your own function, many primitive elements of the language are functions. For instance, functions such as mean(), sum(), plot(), read. csv() are all built in R functions.

Functions is what make your code more clean, easier to debug and organized. Instead of typing the same block of code every time a programmer wants to use it, the program can be written as a function and then called by any other functions. This way its not only more productive, but maintenance and collaboration is a lot easier.

Functions may be roughly divided into two categories:

Inbuilt Functions – These functions are already defined in R and are readily available for the users. Some of the examples are length(), print(), abs(), log() and other.

User defined Functions – These functions are created by user and used to write a customized code not available in built-in functions.

Functions can receive arguments (inputs), perform some internal processing, and send back results. They can also contain looping, conditional logic, and even the invocation of other functions within them. They are also powerful for automating tasks, performing routine calculations, simulating models and generating modular workflows.

Elements that make up a function in R are:

- Function name
- Parameters (arguments)
- Function body (set of statements)
- Return value (optional)

A programmer or analyst can greatly increase the efficiency and clarity of their code by mastering how to write and use methods. In the larger analysis code, use of modular code reduces testing and debugging time, it also becomes easy to scale.

3.1.2 Defining and Calling Functions

To write a function in R, you use the function keyword, which follows a particular syntax. Once the function is defined, you can call (execute) the function by name as many times as you like, giving a different argument each time.

6 Syntax of a function in R:

function_name <- function(argument1, argument2,...) { # Function body or code block

Optional return statement

}

Example 1: A simple function for sum of two numbers

"add_numbers" function(a, b) { result a + b } "just <-function(a,b){a+b}" add_numbers(1, 3) just(1, 3))

return(result)

}

You may use this function: add_numbers(5, 10)

This would return 15.

Some successive stages and a call to the function.

Name the function in a way that explains what the function does.

We can write down arguments if we want: (does not have to be non-empty).

Write the body of the function, which is your logic.

Use return() if you wish to return a value.

Invoke the function by its name with the appropriate arguments.

Example 2: No return type for a function greet <- function(name) { print(paste("Hello,", name))

}

greet("Ravi")

Isn't that the point of writing: "Hello, Ravi" will be printed and doesn't return some value.

5 Function Names:

• Must start with a letter.

• May contain letters, numbers, underscores and dots.

• Do not conflict with the names of existing R functions.

Functions can call other functions from within them. This is called function composition and you can use it to chain things inside a single piece of code.

Functions enhance modularity. For example, if you have a routine with three parts that need to be performed separately, these can be wrapped as one function (each). This architecture helps to make your code more readable, testable and maintainable.

3.1.3 Function Arguments – Default and Named

The values that are sent to a function when it is called as arguments. They are the inputs that the function is doing something to. Arguments to functions in R can be mandatory, optional with default value, and named or unnamed when calling a function.

Mandatory Parameters: These are necessary when the function is invoked. Otherwise, an error is generated.

Example:

```
multiply <- function(x, y) {

return(x * y)

}

multiply(5, 2) # Returns 10
```

Default Arguments: These are the values that have been defined in the function arguments. The default is provided if the caller does not.

Example:

```
greet <- function(name = "Guest") { print(name + "Welcome,") } }

greet() # Defaults to: "Welcome, Guest" greet("Reema")# with given name
```

Default arguments are very useful for cleaner function calls (this makes them more convenient even when most users will use common options anyway).

Argument Names R also gives you the ability to provide argument names when calling a function. This is clearer and makes it easier to send the arguments in a different order.

Example:

```
divide <- function(numerator, denominator) { return(numerator / denominator)}

}

divide(denominator = 2, numerator = 10) # Outputs: 5
```

Argument Matching in R: This mashing of arguments ... in the R programming language is achieved in three ways.

Exact matching by name

Positional matching

Partial matching by name

Ellipsis (...) in Functions: Special form of argument that allows a function to accept variable number of arguments. It's convenient for building wrapper functions or passing arguments to other function calls inside.

Example:

print_more <- function(...) { print("Arguments passed:") print(list(...))

}

print_more(a = 1, b = "text")

Well-thought function arguments are what makes a function both flexible and easy to use. They make a function available for multiple usages, without having to write logic for it again and again.

3.1.4 Return Values in Functions

A function in R doesn't necessarily have to return something explicitly. R has an implicit return of the last expression evaluated if a return value is not given. But it's a best practice to use return() where you can (including if you're returning in the middle of your method or just for clarity).

Syntax:

function_name <- function(arguments) { #Some calculations

return(result)

}

Example:

area_circle <- function(radius) { area <- pi * radius^2 return(area)

}

area_circle(5) # Returns 78.5398

The return() Statement The return() statement will exit the function with a specified result. This is helpful in the case we have a condition and want to break the loop early.

Example with conditional return:

check_positive <- function(x) { if (x< 0) {

return("Negative number")

}

return("Positive number")

}

Returning multiple values: R does not explicitly support the return of multiple individual values, but you can always return a list or vector.

Example:

compute_stats <- function(x) {

result Return result as a list of named scalars with, C mean = mean(x), sd = sd(x) 23 and 2.8|result._XDECREF(result) } Value A list of the estimated population mean (mean) and standard deviation (sd).

}

Best Practices:

• Don't use die(), return() when you have multiple exit point is better way for readability of your code.

• Use lists or named vectors to return values when more than one are involved.

•Record what you will return from the function so you know how to interpret it.

Did You Know?

"In R, if you don't explicitly use return(), the function still returns the last evaluated expression. This makes functions more concise, but it can also lead to unexpected results if you're not careful with the order of statements."

3.1.5 Scope of Variables in Functions

The scope defines accessibility and visiblity of variables in a program. In R the scope defines from where you can read/write a variable. In R, we can have variables created in the global environment and local one inside a function.

Global Scope The above-mentioned is true for variables Defined Outside any function or Block.

Example:

x <- 10 # A global variable print(x)

Local Scope: These are variables which are defined within a function and can only be used by said function. These are not reachable from outside of the function.

Example:

test_scope <- function() { y <- 5 # Local function variable return(y)

}

test_scope() # Returns 5

print(y) # Error: object 'y' not found

Variable priority: If a variable with the same name exists at both global and local levels, then the function will use the local one.

Example:

x <- 100

conflict_test<-function(){ x<-10

return(x)

}

conflict_test() # Returns 10

print(x) #100 (global x has not been changed)

Lexical Scoping: What R does is so-called lexical scoping, in which the value of a free variable (one not within the function) is found where the function was written.

Example:

z <- 2

multiply <- function(x) { x * z}

}

multiply(5) # 10 because z = 2.

Best Practices:

• Use your own local variable names in order to avoid whacking.

Don’t assign to a global variable that’s defined outside of your function.

• rather thanIt’s better to return values from functions instead of setting global variables unless you absolutely must.

The concept of variable scope is fundamental to writing safe, modular functions with the least possible side effects.

3.2 User Input

3.2.1 Reading User Input from Console

User input is an integral part of interactive programming, from making real-time decisions to entering data or customizing the operation during execution. In R, user input is accepted from the console by pre-defined functions like readline() and scan(). These functions are used to script interaction with user, dynamic input collection and flow control.

The readline() function is the most widely used function for taking user input. It takes input as a character string.

, and presents a prompt to the user.

Syntax:

input Or, Using readline input <- readline(prompt = "Enter your name: ")

The message prompt is shown to the user and the result is stored as a string in input.

Example:

name <-readline (prompt="What is your name? ") print(paste("Hello,", name))

Another function is scan, useful when you have to enter numeric or multi-value input. scan() assumes that the input will consist of numerical values by default, but it can also be set up to interpret characters.

Example:

age <- scan(what=integer(),nmax=1)

The user will be prompted to enter a number and then readting the value in age. nmax = 1 restricting the input to one value.

Use Cases:

• Gathering names, IDs or categories from any user(s)

• Asking for numbers, such as age, salary or test scores

Parameter input for dynamic executionDefiningInput yours parameters to be as Dynamic.

- Constructing R scripts for menus or prompts

Points to Consider:

• read the entire input from readline() • all input comes back as a string and needs to be converted for numerical values.

• The readline() function's prompt is not mandatory but adds to the experience of using it.

• Input handling can be handy for little utilities, data ingest tools, quizzes and automation.

Being able to read input directly from the console in R code certainly also makes scripts more flexible and interactive, with a dynamic capacity for both decision-making on-the-fly as well as customization of outputs.

3.2.2 Type Conversion of Input Data

Because readline() always returns character data, you must convert input to the correct type (numeric, logical or date) when performing mathematical or date calculations. It's called type conversion and is the lifeline for data validation, processing and computation.

Example: Typecasting string to numeric num_input <- readline(prompt = "Enter a number: ") num <-as. numeric(num_input)

The user inputs 42, and in the variable num we now have a number.

Available Type Conversion Functions:

• as. numeric() –Coerces a character to these data type.

• as. integer() – Converts to integer

• as. logical() – converts to logical (True/False)

• as. Date() – Changes to Date format

• as. character() – make sure it s considered a string value.

Use Case:

input1 <- readline(prompt = "Enter first number: ") input2 <- readline(prompt = "Enter second number: ") num1 <- as. numeric(input1)

num2 <- as. numeric(input2) sum <- num1 + num2 print(paste("Sum is:",sum))

Handling Non-Numeric Input:

If you type a non-numeric value, such as. numeric() gives NA (Not Available). Such an attitude likewise needs to be justified by validation methods.

Check for Valid Conversion:

if (is. na(num1) || is. na(num2)) {

print("Invalid numeric input. Please enter numbers only.")

}

Common Pitfalls:

• Tilting numbers instead of writing their textual equivalents

• For empty input, the result is "" and one needs to treat it appropriately.

• scan() with wrong type causes weird bugs

Best Practices:

• Make sure to cast the input to expected type prior to processing it.

• Validate input using is. numeric(), is. na(), or custom logic

• Friendly error message for incorrect input

Moreover, type conversion of the accumulated user input into required data types provides consistency for computations and analyses.

3.2.3 Handling Errors in User Input

'Good error handling is critical in user-interactive scripts to avoid surprise crashing and maintain good data hygiene. In R, this will usually involve conditional tests, input-validation loops and pattern matching to handle bad or incomplete inputs."

Use case: User types some non-numeric value.

If a user enters something not represent able as a number e.g. numeric() will return NA. This can be discovered by watching is. na().

input num ans [1] "Enter a number: ". length 3886648304 2 NA > as. numeric(input)

if (is.na(num)) {

print("Invalid Number.")

}

Here is the code where it checks, if the conversion has failed and returns appropriate message back to user.

Repeating Until Correct Input Enter valid input by prompting again:

For better robustness, a loop can be implemented that will continue to ask the user until they provide a valid numeric input.

repeat {

input s s [s>=num] turns each matching string into containing all characters from the original to num > lapply(strsplit(strings,""), function(s) { paste(left,len,right,sep="",collapse="") })$result>unlist(result) Sum of a number between 1 and input whenever there is an exact power output}
cat(i,"\t",sum(((1:i)^2)[which((1:i)^2 ==. numeric(input)

if (! is. na(num) && num > 0 && floor(num) == num) break print("Invalid input. Please provide a positive integer")

}

3.2.4 Interactive Scripts with User Input

Interactive scripts are scripts that answer to your input in time as well give you a result or do something according to the choices made during the execution. In R these sorts of scripts are great for developing small applications, prototypes and for education.

Menu-Driven Example:

choice <- readline("Select an option (1: Sum, 2: Multiply): ")

if (choice == "1") {

a <- as. numeric(readline("Enter first number: "))

b <- as. numeric(readline("Enter second number: "))

print(paste("Sum is:", a + b))

} else if (choice == "2") {

a <- as. numeric(readline("Enter first number: "))

b <- as. (readline("Enter second number: ")) print(paste("Product is:", a * b))

} else {

print("Invalid choice")

}

In this case, the program's behavior is modified by the input of user decisions. This is what now makes it engaging and flexible.

Use Cases of Interactive Scripts:

- Educational quizzes
- Data cleaning tools

• Financial calculators

• Survey forms

• Configuration menus

Building More Complex Flows:

You can stack loops, conditionals and multi-prompts to develop workflows where your users:

• Choose from multiple operations

• Confirm or cancel actions

• Enter multiple records

• Exit the script by command

Looping Through Options:

repeat {

option <- readline("Type 1 to continue, or type 0 to stop: ")

if (option == "0") break #do tasks

}

Tips for Writing Interactive Scripts:

• Use clear prompts

• Validate all user input

• Provide meaningful messages and instructions

• Handling all edge cases for no surprises behavior

Interactive R scripting closes the gap between static data analysis and dynamic use-level processes allowing for interactive applications & utilities.

"Activity: Creating an Interactive Calculator"

In this activity, learners will build a basic calculator in R that accepts user input for two numbers and the type of operation to perform (addition, subtraction, multiplication, or division). The script should validate input, convert values appropriately, and handle division by zero using error-checking logic. It will guide students through designing an interactive loop where users can perform multiple calculations in one session or exit by choosing a specific option. This hands-on task reinforces concepts of user input, type

conversion, conditionals, and loops in R, preparing learners for real-world interactive scripting.

3.3 Importing Data from Various Sources

3.3.1 Reading and Writing CSV Files

CSV (Comma Separated Values) is one of the most popular format for storing and sharing tabular data. They are text files anyway with one row per line, and values being comma separated. When you work in R it is very easy to read a write CSV files, so many data analysts will use this format.

Reading CSV Files:

The read. You use the csv() function to import CSV files in R. This returns a data frame.

Syntax:

data <- read. csv("filename. csv", header = TRUE, sep = ",").

Key Parameters:

• file – the filename or the path to the file

• header: Whether the first row is a set of column names

• sep: The separator; comma for CSV

• stringsAsFactors: Whether or not to convert strings into factors (prefer FALSE)

Example:

students <- read. csv("students_data. csv")

Writing CSV Files:

The write. You can save a data frame to a CSV file using the function csv().

Syntax:

write. csv(dataframe, "output. csv", row. names = FALSE)

Important Options:

• row. names = FALSE: Do not write the row numbers as the first column

• na = "": Determines how missing values are written

Example:

write. csv(students, "cleaned_data. csv", row. names = FALSE)

Best Practices:

• Use getwd() to check your working directory always -Or set the working directory with setwd ()

• Use file. choose() for interactive file selection

• Use read. csv2() for semicolon separated files (common in some European formats)

CSVs are also great for using R with other languages and applications such as Excel, Python, or SQL databases because it is simple and universally readable.

3.3.2 Importing Data from Excel Files

Excel: Format of choice for data storage and reporting. Excel files can be more complicated than CSVs: they might have multiple sheets, cells with formatting, and formulas. Excel datasets can be read into R using specialized libraries such as readxl, or openxlsx.

Using the readxl Package:

For importing Excel data, read_excel() function from the readxl package is widely used.

Syntax:

library(readxl)

data <- read_excel("filename. xlsx", sheet = 1)

Key Parameters:

• path: Path of the Excel file

• sheet: Sheet name or index

• range – an optional string indicating the range of cells (eg 'A1:D10')

•col_names: TRUE/FALSE Use the first row as column names

Example:

grades <- read_excel("students_grades. xlsx", sheet = "Midterm") Dealing With Multiple Sheets: sheet_names("students_grades. xlsx") # Lists all sheets

Writing to Excel Files:

The write. xlsx() function from openxlsx package is used for Writing Excel data.

Example:

library(openxlsx)

write. xlsx(grades, "exported_data. xlsx", sheetName = "Sheet1", overwrite = T)

Additional Features:

• Cell formatting, styles, cell merge and formula creation

•Write various data frames to separate sheets in a single file

Best Practices:

• Tidy up the Excel file before importing (e.g. unmerge cells, remove empty rows)

• Check the type of data after import with str()

• Use sheetnames explicitly whenever you can

Excel file manipulation is fundamental for the analyst in an enterprise setting, where data exchange often occurs through spreadsheet media.

3.3.3 Reading text files (txt, tsv, etc.)

Text files, including. txt,. tsv and similar formats are popular alternatives to CSV. And are frequently found in data exports, web uploads or system ports. R has good support for reading such files with read. table() and variations of it.

Reading.txt Files:

The read. table() function is the most flexible and versatile for importing flat file data.

Syntax:

data <- read. table("file. txt",header=T, sep = "\t")

Key Parameters:

• sep: Delimiter to use, may be a comma (', '), tab ('\t'), or space (' ').

• header : Whether the first line is used as column names

• quote= Quoting characters, to disabled it placed with "".

• stringsAsFactors : which can take value like FALSE (This means convert strings to factor)

Reading Tab-Separated Values (TSV):

For tab-separated files, use read. delim() which is a wrapper for read. table() with sep = "\t" as its default parameter.

Example:

tsv_data <- read. delim("data. tsv", header = TRUE)

Handling Whitespace-Delimited Data:

space_data <- read. table("data. txt", header = T, sep="")

Writing Text Files:

write To write text files you use write. table().

Syntax:

write. table(dataframe, "output. txt", sep = "\t", row. names = FALSE)

Best Practices:

• Always use head() to verify the 1st few rows after importing it.

• Use nrows argument to help you read large files more quickly

• -- Set the fill to TRUE when there are unequal lengths on rows.

Read text-based formats Reading a variety of text-based based file formats is key to flexibility, particularly working with APIs, custom logs or external data exports.

3.3.4 Exporting Data to Different Formats

Data export is frequently the step that completes the data analysis or processing process. If a file has been choked, analyzed, or transformed, it is necessary to save it in the correct form for sharing with others, filing reports, or importing into other software. There are packages and functions readily available to export data to a variety of standard file formats (CSV, Excel, Text/TSV) in R.

Export to CSV (Comma-Separated Values)

CSV is one of the most often used formats to exchange/expose tabular data between different applications.

write. csv(dataframe, "output. csv", row. names = FALSE)

• row. names = FALSE, no row numbers would be written.

• If necessary, specify fileEncoding (e.g., "UTF-8") for character encoding.

Export to Excel

Throughout business and reporting, Excel is the tool of choice. You can write. xlsx files directly.

library(openxlsx) write. xlsx(dataframe, "output. xlsx")

• Supports exporting of sheets, formatting and styles if necessary.

Export to Text (TSV / Tab-Separated Values)

TSV files are plaintext files with columns separated by tabs, convenient for systems that require strict column spacing.

write. table(dataframe, "output. tsv", sep = "\t", row. names = FALSE)

• sep = "\t" will make sure the individual columns are separated by tabs.

• If you don't want the fields quoting when they are character, then add quote = FALSE.

Important Considerations

• Special Characters and Encoding:

You can use fileEncoding = "UTF-8" if files contain non-ASCII character in R. For VBA, you need not do anything.

• Large File Handling:

With huge data, you would also want to write in series (in chunks) or with streaming packages so that you don't blow up the memory.

• File Compression:

Compress your video files to save disk space with a preferred format such as. gz or. bz2.

Example:

write. csv(dataframe, gzfile("output. csv. gz"), row. names = FALSE)

Best Practices

• Confirm Format Requirements:

Verify that the output format is supported by the recipients of the file.

• Include Metadata:

File names may carry timestamps or version IDs for traceability and reproducibility. Example: "sales_data_2023-09-10_v1. csv"

• Automate Repetitive Exports:

You can write R scripts or scheduled tasks (via cron/RStudio Addins) to automate the periodic data exports.

.

Knowledge Check 1

Choose the correct option:

2

1. Which function reads CSV files in R?

a) read.csv

b) import.csv

c) load.csv

d) csv.read

2. What is the default separator in read.delim()?

a) Comma

b) Tab

c) Space

d) Colon

3. Which package is commonly used to read Excel files?

a) dplyr

b) readxl

c) httr

d) jsonlite

4. What does dbDisconnect() do?

a) Saves data

b) Connects to DB

c) Ends DB connection

d) Drops table

3.4 Summary

ϖ R: Functions in R help break a problem into modular pieces and automate repeated tasks} Factors make code re-usable and readable.

ϖ It supports both in-built and user-defined functions that allow custom workflows for certain data analyzing scenarios.

ϖ Arguments of R functions can be mandatory or have default value and named arguments facilitate the function calling expression.

ϖ The return() function is used to do the things explicitly but R returns the last evaluated expression implicitly.

Describing scope is a good way to control variability and conflict between globally vs locally-scoped state in functions.

ϖ User input can be read by R scripts using the readline() and scan() functions, which support interactive execution.

ϖ Type conversion is required to receive user input, since all inputs are normally read as strings.

ϖ Input validation and error handling are necessary to avoid runtime failures and improve user experience for interactive scripts.

ϖ R supports loading data from different sources, such as CSV, Excel sheets, text files, databases and web APIs.

ϖ The read. csv(), read_excel(), read. by(i),table() and fromJSON() are three functions for using flat or semi-structured data more structurally.

-Exporting to CSV, Excel, JSON, and database allows for easy sharing of results and further procressing or reporting.

Integrating functions, user input and data import/export techniques allows the construction of flexible and generic analysis scripts.

3.5 Key Terms

Function – A bunch of reusable code which only runs when it is called.

Argument – A value given to a function in order to customize the behavior of the function or its output.

Return Value - The value provided in response to a function or an expression.

Scope – The portion of a program in which a variable may be accessed.

readline() - Reads a single line of user input from the console as a character string.

as. numeric() – Coerces data into numeric.

Handling Errors – Techniques for dealing with bad or unexpected input in a program.

read. csv()- to read the files stored as Comma Separated Values(csv) in R Examples We will use iris,mpg and mtcars data sets for examples of rbind in R.

read_excel() Function to Read Excel File in R All the above files use readxl package to read an excel file.

read. table() – Reads a table in the form of text from a file.

write. csv() – Converts R data frames to CSV files.

3.6 Descriptive Questions

Describe the advantages of using user-defined functions in R? Illustrate with an example.

Explain how default arguments and named arguments make functions more flexible in R.

Functions (in R): What does the return() statement in an R function do? When is it necessary?

Explain how R manages scope of variables in function? Illustrate with an example.

How do we read and validate user input in an R script? Discuss possible error scenarios.

Explain the process of importing data from Excel files into R, including relevant functions and packages.

OSC: compare read and import of data. csv() and read. table().

3.7 References

1. Grolemund, G., and Wickham, H. (2016). R for Data Science. O'Reilly Media.
2. Matloff, N. (2011). The Art of R Programming. No Starch Press.
3. Verzani, J. (2014). Using R for Introductory Statistics. CRC Press.
4. Kabacoff, R. I. (2015). R in Action: Data Analysis and Graphics with R. Manning Publications.
5. Wickham, H. (2014). Advanced R. Chapman and Hall/CRC.
6. R Core Team (2023). R Language Definition. R Foundation for Statistical Computing.

Answers to Knowledge Check

Knowledge check 1

1. a) read.csv
2. b) Tab
3. b) readxl
4. c) Ends DB connection

3.8 Case Study

Building a Modular Data Analysis Workflow for Customer Feedback Processing

Background:

Each month, MarketPulse, a midsize company, captures this information through online forms to track customer satisfaction. Each submitted form has customer name, productID, rating (1 to 10), and an optional comment. The marketing team wants to be able to look at this data every month and determine:

• Compute average satisfaction by product

• Find out who the unhappy customers (rated less than 5) are

• Export clean and summary data for other reporting

Before that, the team was doing these tasks by hand with spreadsheets. With an increasing volume of data, they were in need of a method that allows the automatisation of this process, using R.

Problem 1: Reading and Cleaning the monthly feedback Data

Monthly data is sent to the team in CSV format. The initial step is to create a script that loads the data and removes entries with missing or invalid satisfaction ratings.

Solution:

clean_data = 1 & satisfaction <= 10) clean_data

}

This function reads the file and discards rows for which satisfaction is missing or inappropriately low.

Problem 2: Detecting Unhappy Customers

The company considers dissatisfaction to be any satisfaction rating under 5. Write a function that takes in a cleaned data frame and returns only the dissatisfied entries.

Solution:

get_dissatisfied <- function(data) { dissatisfied = subset(data, satisfaction < 5) return(dissatisfied)}`

}

This function filters all the entries where the satisfaction score is not as high as you want it to be, so that your team knows exactly what customers to follow up with.

Issue 3: Summary Report with Export option

The last call is to find average satisfaction (mean) by product and export them both as summary and a list of dissatisfied elsewhere.

Solution:

generate_report <- function(cleaned_data) {

summary <- aggregate(satisfaction ~ product_id, data = cleaned_data, FUN = mean)
write. csv(summary, "summary_report. csv", row. names = FALSE)

dissatisfied <- get_dissatisfied(cleaned_data)

write. csv(dissatisfied, "dissatisfied_customers. csv", row. names = FALSE)

print("Reports generated successfully.")

}

This function compiles satisfaction scores, outputs summary and filtered list to CSV files and finally confirms the action.

Reflective Questions

In what way does separating the analysis into functions make the code more readable and reusable?

What kind of check could be introduced to validate the data file before importing?

How to make this script flexible for months or file names if we take user input?

How to extend this scritp to support Excel or API sources?

Please note what are pros and cons of automating such process for non-technical team.

Conclusion

We present this case study as an example of how individual programs can be combined into larger workflows using: user-defined functions, user input handling and data import/export within R. MarketPulse’s marketing team can now consistently and quickly process feedback data each month by modularizing the tasks — read, clean, filter, analyze, export. This saves time and effort, and also means they can respond more effectively to customer gripes. That kind of organized method gives shape to reproducible and business-appropriate data analytics.

BAR Unit 4 V3.docx

Business analytics using R_MBA_2

Business analytics using R_MBA_2

ATLAS SkillTech University

Document Details

Submission ID

trn:oid:::3618:127345966

Submission Date

Feb 2, 2026, 10:35 AM GMT+5:30

Download Date

Feb 2, 2026, 10:38 AM GMT+5:30

File Name

BAR Unit 4 V3.docx

File Size

213.0 KB

21 Pages

4,622 Words

27,276 Characters

1% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text
- Small Matches (less than 20 words)

Match Groups

2 Not Cited or Quoted 1%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

0% Internet sources
0% Publications
1% Submitted works (Student Papers)

Integrity Flags

1 Integrity Flag for Review

Hidden Text
214 suspect characters on 7 pages
Text is altered to blend into the white background of the document.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **2** Not Cited or Quoted 1%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

0% Internet sources
0% Publications
1% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1 Submitted works

Universidad EAN on 2025-08-04 **<1%**

Unit 4: Data Manipulation in R

Learning Outcomes:

1. Use data manipulation verbs (select, filter, mutate, arrange, summarise) to solve the most common data analysis challenges with dplyr.
2. Locate, recognize, and treat missing data in R using functions and logical operators with appropriate techniques (omit or impute), based on context.
3. Clean data as per the analysis requirement, - job includes renaming variables, standardising formats, converting variable types and removing duplicates etc.
4. Use grouping and summarizing techniques with group_by() and summarize() to generate insights from aggregated data.
5. Design reproducible data preparation workflows that ensure consistency across different stages of data preprocessing.
6. Evaluate the impact of data cleaning and manipulation on analysis quality and interpretability, and justify choices made during preprocessing.
7. Integrate multiple data preparation steps into a cohesive pipeline using tools like the pipe operator (%>%) to streamline data workflows in R.

Content:

4.0 Introductory Caselet

"From Raw to Refined – Preparing Data at HealthTrack Analytics"

HealthTrack Analytics is a health analytics company that partners with hospitals and clinics to turn patient records, lab results, and survey data into actionable analytics. The organization recently signed a new client — a chain of diagnostic centers — that wanted monthly reports on patient footfall, test usage, and regional trends in the incidence of disease outbreaks.

As the first set of data came in, the analytics team realized that it was not organized well. There were missing data throughout such a table, in critical variables, coding and date formats inconsistent with the rest of the dataset duplications irrelevant variables. The team also observed that some numeric fields were being treated as character strings, which did not facilitate the direct analysis.

Prepping the data was entrusted to a HealthTrack data analyst named Priya. She began with the data transformation and cleaning procedure in R, employing package dplyr. She ended by showing how to make the vars her work with more visible (via select() and filter()), inserted in this case both chosen vbls and also the records in the dataset she's interested. With mutate() she converted the date and numeric fields to their proper types, while also creating new variables for age groups and test categories in the process. The arrange() function helped her to order the data in chronological manner and she could use summarize() with group_by() for monthly aggregation.

Missing data were handled by deletion and imputation. In other columns, missing values were too common, and Priya filled them in with logical rules and imputations using medians. She also checked other incomplete rows and marked them (and removed them from consideration) when she was unsure of their status.

The resulting clean data was now accurate, uniform and ready for exploration and modeling. But, to the more important point of it all, Priya wrote down every move so that when new data came in, the work could be rerun.

This case study emphasizes the significance of data pre-processing before any statistical or machine learning analysis. Without structured preprocessing — the kind of process we are familiar with when dealing with regularized libraries in Python, for example, where strictly formatted form is required to apply algorithms because they get multiplied by a matrix and return the weighted version of a sentence or paragraph.

Critical Thinking Question:

Why is it important to spend time on data manipulation/cleaning before we get to analysis or visualization (especially in domains like health care or finance)?

4.1 Data Manipulation (select, filter, mutate, arrange, summarize)

4.1.1 Introduction to dplyr Package

dplyr is part of the tidyverse and makes common data manipulation tasks speedier, easier and more readable. It's tailored for working with data frames and tibbles, enabling users to do things like filter rows, select columns, add new variables, summarise data, and arrange rows in simple but consistent commands.

At the heart of dplyr is a grammar for manipulating data, where you specify the actions sequentially: and each action can be thought of as a verb that acts on your data.

• select() – Choose columns

• filter() - Select rows based on a condition

• mutate() – create new variables or modify existing ones

• arrange() – Reorder rows

• summarize() = Combine all of these values in a single summary value

• group_by() – Group the dataset for summarization.

These verbs are meant to be used with the pipe (%>%) which is supplied by magrittr and will enable you to chain operations together in a very natural way. Into the next function, as its first argument, goes output of one function.

For example:

library(dplyr) result %

filter(age > 18) %>% select(name,age,gender) %>% arrange(desc(age))

One of the reasons why dplyr functions are so useful is that they were specifically designed for tidy data workflows, in which datasets are long and standardized.

Other advantages of dplyr include:

• Enhanced efficiency compared to standard R functions

• Support for Big Data and Databases

• Transparent syntax which makes reading and writing code less error-prone

Since data manipulation is an essential data science workflow, proficiency with dplyr will provide users with a powerful toolset for tidying and summarizing real-world datasets in a consistent and aesthetically appealing manner.

4.1.2 Selecting Columns with select()

dplyr select () function is used to subset the columns of a dataframe. [PS002] This is particularly handy when you're working with big datasets that have lots of columns, but an analyst might be only interested in a subset of the columns for viewing or reporting.

Syntax:

select(data, column1, column2,...)

Example:

library(dplyr)

select(students, name, age, grade)

This command will return a dataframe with the columns name, age and grade.

Key Features:

• You can rename columns during selection with the syntax new_name = old_name.

• Negative indexing can be used to deselect columns: select(data, -column1) will include all but column1.

• With some helper functions, you can pattern match on even and odd numbers:

o starts_with("prefix")

o ends_with("suffix")

o contains("substring")

o matches("regex")

o everything() – selects all columns

Example with helper:

select(data, starts_with("score"))

This grabs all the columns that start with "score" which is useful for datasets where you have wide data with repeated measures.

Best Practices:

• Use select() at the start of the data pipeline to reduce memory overhead.

• Use select() in combination with rename() for more readable variable names.

• Column positions (e.g., select(1, 3, 5)) should be avoided unless the structure is known to be fixed.

Using select(), analysts can simply focus on the important aspects of that data rather than finding themselves lost amid unnecessary or duplicate variables.

4.1.3 Filtering Rows with filter()

The filter() function is applied on data frame to selects only rows that satisfy a particular condition. The above process is one of the most frequently used data-tinkering tasks, enabling analysts to drill down on interesting cases and exclude irrelevant or noisy ones.

Syntax:

filter(data, condition)

Example:

filter(students, age > 18)

This will fetch all the records from students dataset where age is more than 18.

Multiple Conditions:

You can join conditions using the logical operators:

• & – AND

• | – OR

• ! – NOT

Example:

filter(students, age > 18 &grade == "A")

This results in students aged over 18 who have achieved grade A.

Common Use Cases:

Dependent Data type Checking Removing rows/ reading rows with missing or invalid data.

• Choosing records from a certain category (eg. men or women)

• Filtering data by date intervals and on number values.

Tips for Effective Filtering:

Type within the columns must be the same as used in filter condition (ex: use as. Date() for date comparisons).

• I'm not sure, but I think you can hack the problem by joining with mutate() to create new columns of logical flags, and then filter on those.

• Filter using %in% for values stored in a list:

• filter(data, region %in% c("North", "East"))

Did You Know?

"You can combine filter() with is.na() to quickly identify missing values, or with complete.cases() to remove them. This makes filter() a versatile tool not only for subsetting but also for cleaning datasets before analysis."

4.2 Handling Missing Data

4.2.1 Identifying Missing Values

In practice, missing data is a common headachy problem in real experiments. It may be due to various reasons such as incomplete data entry, patient non-responses during survey administration and system failure or heterogeneity in data integration. In R, missing values are denoted by the symbol NA which means "Not Available". It is important to accurately identify and comprehend the missing values pattern before handling them.

In order to identify those missing values in R, the function is.. na() is commonly used. This function gives us a logical vector that tells whether an element is true if it's missing or false if not.

Example:

data <- c(12, NA, 25, 18, NA)

is.na(data)

This gives: FALSE TRUE FALSE FALSE TRUE, which shows where the missing data are. For checking missing values in a complete dataset:

is.na(dataframe)

This will give you a logical matrix. The number of missing values can be calculated as follows:

sum(is.na(dataframe))

Printing missing data for certain columns:

colSums(is.na(dataframe))

This gives us a column-wise summary of missing values, it will be really useful goodness for us to focus on our cleaning efforts. For a row-wise summary, use:

rowSums(is.na(dataframe))

Equally important is the pattern of missingness. If the NA's tend to cluster in particular columns or sets of columns, this could suggest more serious problems such as inconsistent data collection (or even some sort of targeted-bias). Libraries such as VIM and naniar offer visualizations for checking missingness pattern.

This not only biases the results, reduces statistical power but, if data are ignored inappropriately, can lead to outright inappropriate conclusions.

4.2.2 Removing or Replacing Missing Data

Having identified missing values, the next decision is whether we should remove them or if they need to be filled in with some type of imputation. It is context dependent, depends on the amount of missingness and what the data looks like.

Removing Missing Data:

This is the most basic method, and it might be used when:

• There are not a large proportion of missing values

• MAR or MCAR – Missingness is random and will not lead to bias.

• The result is independent of the affected parameters

To delete a row with null in it:

cleaned_data <- na. omit(dataframe)

Or the opposite, to get rid of rows with NA in specific columns: cleaned_data <- dataframe[! is. na(dataframe$column_name),] Replacing Missing Data:

Substituting or dealing with missing values is helpful if:

• Rows would get deleted and lot of data lost

• Ones with missing values are important.

•Wish to keep sample size and reduce bias

Common strategies for replacement include:

• Mean/Median imputation (for numerical data)

• Mode imputation (for categorical data)

• Imputation of constant value (e.g., NA replaced by 0 or "Unknown")

• Filling forward or backward in time series data

• Regression or predictive model based imputation

Ps. Example of replacing NA with mean:

data$age[is. na(data$age)] <- mean(data$age, na. rm = TRUE)

Imputation must be performed with care so as not to impose spurious structure or bias into the data. Be sure to always clarify imputation strategies to allow for reproducibility and transparency.

4.2.3 Using na. omit(), is. na(), and Imputation Methods

R has some inbuilt functions to handle the missing values programmatically. They all have their specific jobs and can be used singly or together for efficient data preprocessing.

is.na()

As discussed, is. na() is used to indicate missing values for vectors, matrices and data frames. Example:

is.na(df$salary)

This gives us a logical vector of which elements in salary are missing.

na.omit()

This function removes any row from a data frame that has any missing value. Example:

clean_df <- na.omit(df)

Use such a function when the data set can afford to lose incomplete cases or when the variables with missing values are not very important.

Imputation Methods

Imputation is a convenient method for handling missing values by filling out the blanks with plausible estimates.

• Mean/Median Imputation:

df$income[is. na(df$income)] <- mean(df$income, na. rm = TRUE)

4.2.4 Best Practices for Handling Missing Data

Dealing with missing data needs to be treated with care in order for the analysis to maintain integrity and accuracy. Bad handling of missing value can lead to biased conclusions, bad models, and less trust in results analysis.

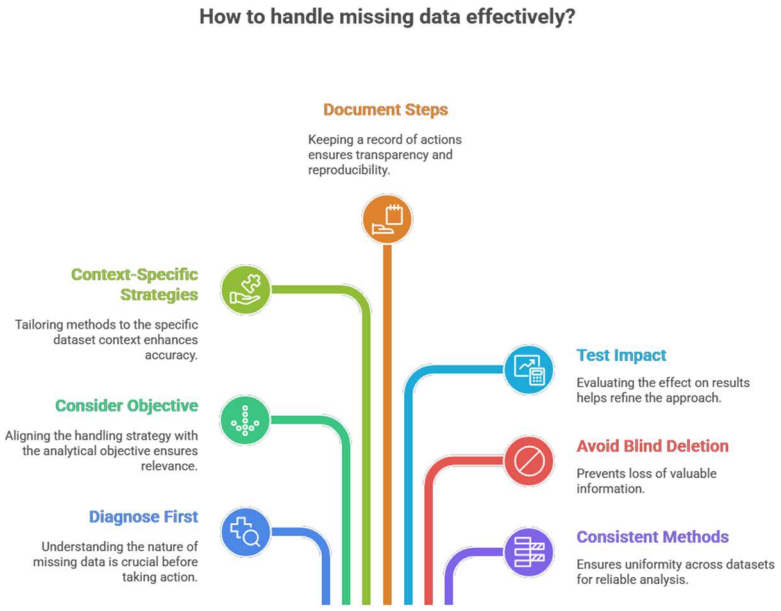

Figure 4.1

Diagnose Before Acting:

Before you apply any strategy, first analyze how much and which parts of the data are missing with:

• summary()

• is.na() with colSums()

• Visualization packages such as VIM::aggr() or naniar::vis_miss()

Consider the type of missingness applied; whether it is Missing Completely at Random (MCAR) or choice-based: Missing at Random (MAR) and Missing not at random...

Consider the Analytical Objective:

• Drop the columns if you don't want to use variables with missing values in your model.

• If those variables are important, use the appropriate imputation techniques.

Choose Context-Specific Strategies:

• Guide imputations with domain knowledge (such as, replacing missing temperatures by the seasonal average and not 0).

• timeseriesimpute: for timeseiresimpute based imputations lags or interpolation can be used.

Document Every Step:

• Note world? which values were not present

• Explain how you handled missing values

• Ensure reproducibility and auditability

Test Impact on Results:

• Imputation and non-imputation analysis

• Comparative analysis of the models using accuracy and output similarity

Avoid Blind Deletion:

• The implicit removal of missing features may eliminate useful patterns

• DELETION_REASON should be explainable and biased to assure lack of biasing the data

Use Consistent Methods Across Datasets:

•Standardize train and test sets using equivalent methods for imputation or deletion.

Imputation of missing data is commonly reported as being more art than science. When we combine statistics with a healthy dose of contextual knowledge, our data is further filtered and made more truth-prone.

Cleaning Survey Data for Analysis

In this challenge, you're given a fictional data set from an online survey with missing values in the age income and gender columns. What I would like to do is extract the missing values with is. na() and visualize their distribution. Students have to decide whether they would like to drop the missing data, or fill in those missing values and if so, with what. From which we then have a clean dataset with which to compute some basic summary statistics, and write a report justifying each sub-group of the data that was included or not. You will be working with live data for order and pre-processing in an analytics workflow.

“Activity: Missing Data Diagnosis and Treatment”

Cleaning Survey Data for Analysis

In this activity, students will receive a mock dataset from an online survey containing missing values in age, income, and gender columns. The task is to first identify missing values using is.na() and visualize their distribution. Students must then decide whether to omit or impute the missing values using appropriate techniques such as mean

imputation for income and mode imputation for gender. The cleaned dataset will be used to compute basic summary statistics and generate a report justifying each cleaning decision. This activity encourages practical understanding of data integrity and preprocessing steps in real-world analytics.

4.3 Data Cleaning and Preparation

4.3.1 Detecting and Removing Duplicates

Duplicate records can arise during a system merge or data entry, and may distort results by over-representing particular observations or introducing systematic bias. In R, we use the duplicated() function which returns a logical vector of true or false depending on whether an element is duplicate in nature. For example:

dup_flags % distinct()

For conditional duplicity checks(like if any duplicate in key fields but not entire columns) can as well specify:

unique_data % distinct(id, date,. keep_all = TRUE)

We can also find duplicates by group. Flag, for example, each row with the same customer name and purchase date. The choice of whether to remove duplicates after they are detected is typically a domain knowledge question. Sometimes duplicates can also mean valid repeats (i.e. reorders) and should be handled differently (e.g. summarized or consolidated) instead of deleted.

Additional considerations:

•Confirmation before delete: Check duplicates through app for confidence.

• Keep one instance: Use. keep_all = TRUE to retain the whole row of its first time.

•Deduplicate documents to maintain replicatability.

• Employ packages like janitor, which has a get_dupes() function that shows you the duplicates for your perusal.

Second, removal of duplicates will maximize data fidelity and analytic power by allowing for each person or event (episode) to be included only once.

4.3.2 Harmonization of Column Names and Data Types

Consistency with the column name and datatypes is very important while cleaning data. Data always has ugly column names (spacing, mixed case, special characters) that can make typing and programming ugly.

Automatically snake casing column names using janitor::clean_names(): cleaned_df % rename(age_years = age, total_score == score_total)

It's not just names, data formats (dates, factors, numbers and text) must be normalized. These strings may be in a form of " DD/MM/YYYY" or "MM-DD-YY". The lubridate library has a function to convert them as Date objects, so that we use the same class in both cases:

df$date % filter(variable upper_bound)

Treatment options:

- If they are due to clear mistakes you can remove the outliers.
- Cap them at bound, for example lower_bound or upper_bound (winsorization).
- Return mean or median which is not outlier:

mean_value = lower_bound & df$variable upper_bound] <- mean_value

- Data transformation (e.g., log-transformation) to control the impact of outliers while retaining data.

Outlier-based decisions need domain knowledge as part of them. In some applications, outliers are informative (e.g. fraud detection). Box-plots or plot()are intuitive to visualize the anomalies.

Did You Know?

"Sometimes a single extreme value represents a legitimate observation rather than an error—such as extraordinarily high sales in a seasonal peak. Automatically removing it could erase meaningful insights. Validate outliers before deciding how to treat them."

4.4 Grouping and Summarizing Data

4.4.1 Grouping Data with group_by()

In the dplyr package, to divide a data frame into groups using one or more variables, we use group_by() function. This group structure permits downstream operations (aggregation, mutation) to be carried out within groups rather than over the entire dataset. This is particularly useful for working with real data, where analysis and insight invariably occur by category, region, time period or any factor that can be used to group data.

Basic Syntax:

grouped_data % group_by(column_name) For more than 1 grouping variable:

grouped_data % group_by(col1, col2)

When the data is grouped, any function you apply afterwards -- e.g. summarize(), mutate() or filter() -- will run within each group on its own. For example:

data %>% group_by(department) %>%

summarize(avg_salary = mean(salary, na. rm = TRUE))

So we are taking the average salary per Department. Grouping is also helpful together with mutate(), to add group-level vars on each row:

data %>% group_by(department) %>%

mutate(group_mean = mean(salary, na. rm = TRUE))

Additional Considerations:

• There is no visual change in the structure of the dataset when grouping, until a scrunching function (like sum) has been performed.

• You can also ungroup the data, if you so need, by using ungroup() :

• data %>% ungroup()

The group_by function is the centerpiece of most aggregations and comparisons in R, so it's essential to data analysts working with grouped or categorized data.

4.4.2 Aggregating Data Across Groups

For aggregation you've got measures including the most common such as sums, averages, counts and maximums. When we've group_by() on, summarize() is what you need to do this sort of thing. This provides analysts with ability to compute summary statistics by a group and make comparisons.

Example – Average by Category:

data %>% group_by(category) %>%

summarize(avg_value = mean(value, na. rm = TRUE))

Common Aggregation Functions:

• mean(): average

• sum(): total

• n(): count of rows

• n_distinct(): count of unique values

• min() / max(): The minimum and maximum

Count by Group:

data %>% group_by(gender) %>% summarize(count = n())单值统计:

data %>% group_by(region) %>%

summarize(unique_customers = n_distinct(customer_id))

Grouped Aggregation with Filtering:

filter() can also be chained before or after grouping to filter based on the output of group-level summaries. For example:

data %>%

filter(score > 80) %>% group_by(class) %>% summarise(high_scorers = n())

Summing across groups additionally provides simple comparisons, trends and performance assessment among data's subgroups.opensource. It is most useful for reporting, dashboards, and business intelligence applications.

4.4.3 Combining Multiple Summary Statistics

It's frequently helpful to calculate more than one summary statistic at the same time within each category. In R, with summarize() we can compute several expressions in a single call and achieve a nice summary per group.

Example:

data %>% group_by(department) %>% summarize(

count = n(),

avg_salary = mean(salary, na. rm = TRUE), max_salary = max(salary, na.rm = TRUE)) comp [1] 53. rm = TRUE), min_salary = min(salary, na.rm = TRUE)) %>%. rm = TRUE)

)

This produces a four column summary by department (count, mean, max and min of salaries).

Adding Standard Deviation and Median:

data %>% group_by(location) %>% summarize(

total = sum(sales, na. rm = TRUE), median_sales = median(sales, na. rm = TRUE), sd_sales = sd(sales, na.rm = TRUE)) rm = TRUE)

)

Custom Summary Functions:

You might want to create a function such as the following and use it while summarizing:

iqr_func %

group_by(category) %>% summarize(

count = n(),

iqr_value = iqr_func(price)

)

When combining multiple statistics, remember:

• Always handle missing values (na. rm = TRUE) in order to prevent computing errors.

• Arrange the results (if necessary) with arrange()

• arrange(desc(avg_salary))

Merging several summaries together is important because this not only keeps the analysis comprehensive, but it helps lay a foundation for aggregating the insights to share with stakeholders or to model further.

4.4.4 Practical Examples of Grouped Analysis

Grouped analysis is a popular tool in practical data analysis that can provide actionable insights by examining the behavior of subgroups of interest in a dataset. Here are a few practical examples where grouping summarization is mandatory:

1 Sales Analysis By Region: sales_data %>% group_by(region) %>% summarize(

total_sales = sum(sales, na. rm = TRUE), avg_sales = mean(sales, na. rm = TRUE)

)

This is useful for management to see what regions do well and where strategic changes need to be made.

Class Gender Student Performance by class and sex:

students %>%

group_by(class, gender) %>%

summarize(

avg_score = mean(score, na. rm = TRUE), top_score = max(score, na. rm = TRUE)

)

Multiple stratifications enable sophisticated comparison of academic performances and educators can break down achievement gaps among several subpopulations.

Customer Retention Rate by Type of Product:

customer_data %>% group_by(product_type) %>% summarize(

retention_rate = mean(retained, na. rm = TRUE), customers = n()

)

Department # of Employees Avg. Period Working in this Departament

hr_data %>% group_by(department) %>% summarize(

employees = n(),

avg_tenure = mean(years_at_company, na. rm = TRUE)

)

Time-Series Aggregation:

For trends, you can bin on time (e.g. by year/month):

data %>%

year = lubridate::year(date)) %>% group_by(year) %>%

summarize(total_revenue = sum(revenue, na. rm = TRUE))

Meaningful data is more detailed if it's taking into the manageable, comparable portion with the help of grouped analysis. It is an essential part of reporting, strategy development and data ariveled decision from insight.

Knowledge Check 1

Choose The Correct Options :

1. Which function is used to split data into subgroups?

a) group_by

b) split_rows

c) classify

d) separate

2. Which function calculates summary statistics after grouping?

a) filter

b) summarize

c) select

d) count_rows

3. What does n() return inside summarize()?

a) Mean

b) Sum

c) Count

d) Mode

4. What function is used to count unique values?

a) n_distinct

b) unique_count

c) distinct

d) unique

5. How do you remove groupings from a grouped data frame?

a) ungroup

b) remove_group

c) clear

d) disband

4.5 Summary

ϖ With dplyr package, we can manipulate data quickly with simple function like select(), filter(), mutate(), arrange() and summarise().

ϖ select() can be useful where you want to take only a few columns from the dataset towards analysis and saving memory.

ϖ A good feature of filter() is its ability to subset rows with logical conditions in order to focus on particular analysis.

ϖmuate() creates new variables, or transforms old ones (a feature engineering tool).

ϖ arrange() arranges rows according to one or more columns that are ordered in ascending or descending order.

ϖ summarize() calculates summary statistics, like the mean, sum or count over grouped data.

ϖ Use is to see the missing values. "). nais leading to na() which are then removed(replaced) with NA. omit() or imputation methods.

ϖ Appropriate management of missing data allows accurate analysis and prevents bias or misleading conclusions.

ϖ Data cleaning: a) eliminating the duplicates; b) changing the column names to the same format; c) transforming data into the correct format (e.g. float instead of string); d) processing potential outliers.

ϖ The function group_by() is used to divide the data into groups and do analysis in individual grouped multilevel data.This is helpful when we have to make comparisons between groups and also check trends.

ϖ Aggregating summary statistics on group-wise data enrichens analytics depth and understanding.

ϖ Quality, consistency and reliability in visualization ormodeling tasks require aclean and well prepared dataset.

4.6 Key Terms

select() – get columns by name from a data frame.

filter() – To select rows with specific conditions.

mutate() – The function used to create or change columns in a data set.

arrange() – Orders rows by column value.

summarize() – Combines data into summary(s) such as averages.

is. na() – Identifies missing values.

na. omit() - Drop rows with NaN values.

Imputation - Process of filling the missing data with estimated information.

dropplicates() – Identifies and removes instances of duplicate rows in a dataset.

group_by() – Group data by one or more variables for group-wise manipulation.

n() – Counts rows in a group.

n_distinct() – Count number of distinct values in a group.

4.7 Descriptive Questions

Describe what the package dplyr is used for in manipulating data and explain some of its essential functions.

How do I use the filter() function in R? Provide examples with multiple conditions.

And what's the point of having the mutate function? Explain cases in which it is helpful.

Explain how missing values are handled in R using is. na() and na. omit().

Observe the method to detect duplicates in a dataset and remove them as well.

How can you aggregate and summarize data in an R dataframe?

What is an outlier, and how do you find and treat them in a dataset?

Describe what is involved in standardizing column names, and why it is an important aspect of data cleaning.

4.8 References

1. Grolemund, G., & Wickham, H. (2016). R for Data Science. O'Reilly Media.

2. Matloff, N. (2011). The Art of R Programming. No Starch Press.

3. Kabacoff, R. I. (2015). R in Action: Data Analysis and Graphics with R. Manning Publications.

4. Wickham, H. (2014). Advanced R. CRC Press.

5. Peng, R. D. (2016). Exploratory Data Analysis with R. Leanpub.

6. R Core Team (2023). R Language Definition. R Foundation for Statistical Computing.

Answers to Knowledge Check

Correct Answers for Knowledge check 1 :

1. a) group_by
2. b) summarize

3. c) Count

4. a) n_distinct

5. a) ungroup

4.9 Case Study / Practical Exercise

Building a Clean and Analyzable Dataset for Retail Sales Analytics

Background:

ISIN: US04965R1095) ("FreshMart") chain of stores, which serves various regions in the U.S. Every month they create a new dataset that contains the transactions for that month, e.g. store ID, product number, sales amount and customer demographics including the date of purchase. Nonetheless, because the data entry may be in various forms among branches and systems, the raw dataset commonly includes missing values, duplications, incompatible formats and anomalies. The analytics team is responsible for cleansing the data before saving it for sales performance reporting, or even customer segmentation.

Issue 1: Duplicates need to be removed and columns need to be standarized.

The database contains a number of duplicated and misnamed columns. They can create inaccurate sales reports via the duplicated rows, stray looking column names in the report generation scripts.

Solution:

library(dplyr) library(janitor)

Step 1: Clean column names sales_data % distinct()

These leaver your data structure clean, and the columns are nicely formatted for reference purposes.

Problem 2: Dealing with Missing Values and Imputation of Data

N/A values were discovered in the customer_age and sales_amount fields. The team requires proper methods to manage these so as not to introduce bias in the analysis.

Solution:

Identify missing values

colSums(is.na(sales_data))

missing sales_amount by median if NA's(((together_Amount[,c('convdate','sales_amount')])$sales_amount))) together_Amount[is.na(together_Amount$sales_amount), 'sales_amount']

1 sales_data$sales_amount[is. na(sales_data$sales_amount)] % group_by(region) %>% summarise(

total_sales = sum(sales_amount, na. rm = TRUE), avg_transaction = mean(sales_amount, na. rm = TRUE), transaction_count = n()

)

Such output assists managers efficiently in comparing the sales performances of various regions.

Reflective Questions

Why is duplicateness not allowed when statistics are to be done?

How are impute methods having impact on the dataset results? What are the dangers of doing bad imputation?

When would you decide to aggregate data before summarizing?

What are some things to think about before dropping rows with missing data?

How will automating the clean steps improve quality and consistency of future analyses?

Conclusion

Through the use of this scenario, we address implementation challenges in real-world data cleaning and conditioning problems in retail analytics. With cleaning out duplicates, dealing with missing data and doing grouped summaries the people over at FreshMart were able to take a raw unstructured dataset and make something useful of it. This pre-processing is as important as accurate reporting and is a key enabler to predictive models, customer segmentation and actionable business intelligence. Data cleaning is not a once-and-done task but an iterative and scalable process that must be developed as the basis of any real analytics pipeline.

BAR Unit 5 V3.docx

Business analytics using R_MBA_2

Business analytics using R_MBA_2

ATLAS SkillTech University

Document Details

Submission ID

trn:oid:::3618:127197655

Submission Date

Jan 30, 2026, 4:01 PM GMT+5:30

Download Date

Feb 2, 2026, 10:28 AM GMT+5:30

File Name

BAR Unit 5 V3.docx

File Size

117.3 KB

30 Pages

7,467 Words

44,013 Characters

6% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text
- Small Matches (less than 10 words)

Match Groups

1 Not Cited or Quoted 6%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

6% Internet sources
0% Publications
6% Submitted works (Student Papers)

Integrity Flags

1 Integrity Flag for Review

Hidden Text
162 suspect characters on 6 pages
Text is altered to blend into the white background of the document.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **1** Not Cited or Quoted 6%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

6% Internet sources
0% Publications
6% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1 Internet

ddceutkal.ac.in **6%**

Unit 5: Data Visualization in R

Learning Outcomes:

1. Explain the principles of the ggplot2 package and its layered grammar of graphics framework for data visualization in R.

2. Create and customize basic plots such as scatter plots, line plots, bar charts, and histograms using ggplot2 functions.

3. Interpret relationships and trends by applying ggplot2 to visualize associations, comparisons, and distributions in datasets.

4. Enhance visualizations with aesthetics such as color, size, shape, labels, and themes to improve interpretability.

5. Use faceting techniques to generate multi-panel plots that allow comparisons across subgroups of data.

6. Differentiate between plot types and justify the use of appropriate visualization methods based on the nature of variables.

7. Integrate ggplot2 plots into analysis workflows for effective data storytelling and communication of insights.

Content

1

5.10 Descriptive Questions

5.11 References

5.12 Case Study

5.0 Introductory Caselet: "Visualizing Insights at AgroTech Solutions"

AgroTech Solutions is a leader in agricultural insights. Their customers range from farmers, policy makers and supply chain managers who use these insights in real-time to enhance crop yields, forecast demand and manage logistics. AgroTech collects data on rainfall, soil moisture and health, fertilizer usage, livestock feeding and market prices from several regions each month.

At first, their analysts created long tables to report findings. But users struggled to make sense of huge tables of numbers. For instance, farmers needed to rapidly ascertain how rainfall was impacting their crops yield, while policymakers required a comparison of fertilizer usage in different regions. Tables were not sufficient to draw out trends or patterns, or relationships.

Realizing this shortcoming, AgroTech's data science team leveraged ggplot2, the powerful R package that creates complex visualizations. With ggplot2, analysts could make scatter plots that revealed how rainfall correlated with yield, line plots that tracked changes in soil moisture over time and bar charts comparing production across regions. They also employed histograms to examine the distribution of crop prices and faceting to produce multi-panel plots which compared regions side-by-side.

These visualizations turned inert data into easy to understand narratives. For example one plot showed that even though the rains were different, yields kept constant through means of consistent fertilizer application. Another visualization indicated that price swings had been more volatile in the north than in the south.

With ggplot2, AgroTech were able to not only make their internal analysis better but also communicate with nontechnical stakeholders. Farmers could have data-driven guidance when entering their fields, policymakers would be able to flip through visual abstracts for making strategic decisions.

Critical Thinking Question:

Why do you think it is that people find visual representations of data, like ggplot2 makes, to be more effective for decision making than say the raw tables or textual summary?

5.1 Introduction to ggplot

5.1.1 Grammar of Graphics – Concept Behind ggplot2

ggplot2 is built on the Grammar of Graphics framework by Leland Wilkinson. The idea is to regard a graph not as an image that you're done drawing, but a result of layering together objects or data and the associated set of aesthetics, geometries, scales, and layers. The grammar part offers a systematic way of thinking about how visualizations are constructed instead of concentrating on what they look like.

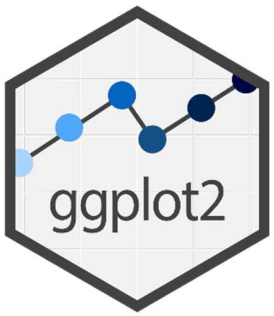

Image Credit: https://commons.wikimedia.org/w/index.php?curid=155077927

The Grammar of Graphics in essence rests on the idea that every plot is simply data that has been mapped onto a coordinate system with visual attributes. For instance, numbers can be indicated by position along the axis, categorical variables by color or shape and values by size. ggplot2 is built based on this idea, by combining those elements systematically you can generate a tremendous amount of visualizations.

Some of the central concepts of the Grammar of Graphics are:

• Data: The dataset being visualized.

• Aesthetics (aes): the mapping of data variables to graphic properties such as x-position, y-position color or size.

• Geometries (geom): The visual objects you use to represent the data, like points, bars or lines.

• Scales: Instructions on how to translate data values into visual values, such as axis ranges or color gradients.

• LAYERS: Multiple layers of data and geometry in order to enrich the visualization.

• Themes: In customizations that pertain to appearance of non-data (background or text format) elements.

ggplot2 is powerful because it generalizes more complicated visualizations and maintains the same syntax so that people can try different permutations. Your analysts don't have to remember yet another unique command for drawing a different chart type, they need to learn how the system works and can be adapted to meet almost any data visualisation requirement. This kind of flexibility makes ggplot2 an indispensable instrument in today's data science.

5.1.2 Structure of a ggplot Command

all ggplot graphics are composed of the same building blocks: data, a coordinate system, and geoms—the visual representations of data points. Know this structure to create plots that work.

In the following, we shall describe in brief some of the most important ggplot commands: A basic structure of an ordinary ggplot command is comprised by three main elements:

Initialized with ggplot(): This command indicates the Data Frame we will use and specifies the Aesthetic Mappings through aes(). There is still no plot at this point.

ggplot(data = dataset, aes(x=variable1, y=variable2))

Adding Geometries with geom_*: A geometry is a layer in which the data will be represented. For instance:

o geom_point() creates scatter plots

o geom_line() creates line charts

o geom_bar() creates bar charts

ggplot(data, aes(x, y)) + geom_point()

More Layers, and Detailing: Other commands can introduce to the plot \emph{transforms}, \emph{scales}, \emph{labels} or even change its default theme.

ggplot(data, aes(x, y)) + geom_point() + labs(title = "Scatter Plot") + theme_minimal()

The structure is modular. With each successive addition to a ggplot command, you are adding another level of specification much as layers are stacked on top of one another. This makes ggplot highly adaptable. To illustrate, adding + geom_smooth() to a scatterplot will overlay a regression line instantly, and it is easy to see how such minimal syntax improves graphic interface.

Another characteristic of ggplot structure is that the data and aesthetics can be declared at a global or local level:

• Some options for ggplot() work globally across all layers.

• Local specification in geom_*() only affects that layer.

Because of this structural flexibility, ggplot2 is both powerful and extremely modifiable; analysts can tweak everything from the size of points or lines to the placement of legends in order to match visualizations as closely as possible to analytical aims.

5.1.3 Aesthetics (aes) and Geometries (geom)

One of the most important concept in ggplot2 is this separation between aesthetics and geometries: they are what help both you and ggplot2 decide how your data should be drawn on a figure.

Aesthetics (aes): It is used to describe which variable are to be plotted. The most common aesthetics include:

• x and y: x and y on the plot.

• color: Automatically colors points by category or value.

• shape: Plots points with different shape, commonly used for categorical data.

• size: Represents magnitude or importance.

• fill : Specifies the fill color of objects, such as bars or histograms.

• alpha: Adjusts transparency.

Example:

ggplot(data, aes(x = age, y = income, color = gender, size = spending))

Here, age is on the x-axis, income is on the y-axis, gender is used as the color of dots and spending represents the dot sizes.

Geometries (geom): These specify the kind of plot or shape of the chart you make. Examples include:

• geom_point() for scatter plots

• geom_line() for line graphs

• geom_bar() for bar charts

• geom_histogram() for histograms

• geom_boxplot() for boxplots

Each geometry has default aesthetics. For example, geom_point() will use x and y, whereas geom_bar() will use x and count by default.

Integration of aes and geom:

The double combo of aes() mappings and geom_*() selections generate the plot. E.g: ggplot(data, aes(x = category, y = sales, fill = region)) + geom_bar(stat="identity")

So, we'll end up with a bar chart that has the y-axis as sales, and x-axis as categories and regions represented by colour.

By splitting the datamappings (aes) from the geometry (geom), you gain maximum adaptability. The same data can also be visualised in multiple ways by changing geometries but keeping the same aesthetic mappings, or conversely.

5.1.4 Layers, scales and themes with ggplot

Lets update the above chart and change its default color demonstrating how we can override the theme defaults!

One of the hallmarks of ggplot2 is the layered generation of plots. Each plot is built as a series of layers to enable multiple pieces of information to be arranged in an organized manner.

Layers:

A ggplot geometry that appears after another is an additional layer. Layers can depict raw data, trend lines, labels or statistical summaries.

ggplot(data, aes(x=x, y=y)) + geom_point() + #center on middle class less than evil x2 < module avg why alt right heart of gold left values.mod average_ai v toxic_avg so by how/where does it stands?

geom_smooth(method = "lm", se = F)

This one-block of code plots a scatter plot on top of another regression line.

Scales:

Scales determines how data values will be associated with visual properties. They can be used to scale axes, colors, sizes and shapes.

• scale_x_continuous() and scale_y_continuous() are used to change the range and labels of axes.

• scale_color_manual() allows manual color assignment.

• scale_fill_gradient() adjusts continuous color gradients.

Scales are essential as they allow plots to be adaptively tailored to the audience, so that results are interpreted clearly and accurately.

Themes:

Themes change the non-data elements of your plot including fonts, backgrounds and grid lines. Common built-in themes include:

• theme_minimal()

• theme_bw()

• theme_classic()

Themes help to enhance readability and apply consistent style through out the plots. Analysts can also make their own themes for branding or reporting requirements.

Additional Features:

- Labels can be assigned using labs() for titles, axis names and captions.

•Legends are created by default, but can be modified or suppressed with theme(legend. position = "none").

These, combined, are layers, scales (and stats, I will tell later), and a theme that make ggplot2 a full system of visualisation. They enable analysts to incrementally create plots, enrich them with context and style the visuals for professional dissemination.

5.2 Creating Basic Plots

5.2.1 Syntax for Basic Plots in ggplot2

Plots can be built following a consistent syntax and conventions based on its grammar of graphics idealogy. The basic frame is the ggplot() function, which will set up a plot system, then we add layers of one or many (with the + operator). At the bare minimum, a basic plot should have at least: a dataset aesthetic mappings a geometry function

The general syntax is:

ggplot(data = dataset, aes(x = variable1, y = variable2)) + geom_function()

• Data: Dataset to be visualized is provided through data argument.

• Aesthetics (aes): You will have to tell ggplot which variables (input or mappings) correspond to the x and y-axes.

• Geometry (geom_*): Specifies the type of plot, eg: geom_point() for scatter plots, geom_bar() for bar charts, geom_line() for line plots.

For instance, making a scatter plot:

ggplot(data= mtcars, aes(x = wt, y = mpg)) + geom_point()

This does the mapping of wt to the x-axis and mpg to the y-axis, with the data points represented by dots. More layers can be added to make the visualization clearer:

ggplot(mtcars, aes(wt, mpg)) +

geom_point() + geom_smooth(method = "lm")

This overlays a regression line on the scatter plot.

Additionally, it is, of course, possible to map categorical variables directly onto aesthetics such as color or shape and ggplot2 will automatically lay out the visualization according to group differences. Single syntax also makes ggplot versatile, where you can build the plot step-by-step without repeating a new command every time.

5.2.2 Customizing Axes, Labels, and Titles

Good plots don't just show data, they present it; and, the insights they communicate are easy to understand. ggplot2 gives you convenient functions to customize everything from your axis labels, plot titles and even the overall plotting details.

The labs() function is most often used to set labels. You can specify the titles of the axes, the main title of the plot (as well as a subtitle), and description:

ggplot(mtcars, aes(wt, mpg)) + geom_point() + scale_fill_manual("c) Manual Fill") #Looks like there is a bug here. 'labels' are currently incorrectly placed for the magnifying glass as well.

labs(

title = "Fuel Efficiency by Weight", subtitle = "Motor Trend Cars Data", x = "Car Weight (1000 lbs)",

y = "Miles per Gallon",

caption = "From: Motor Trend Magazine"

)

We can also customize axis by using scale functions:

• scale_x_continuous() and scale_y_continuous() for numeric variables.

• scale_x_discrete() and scale_y_discrete() for factors.

Example with axis breaks and limits:

ggplot(mtcars, aes(wt, mpg)) + geom_point()
+facet_wrap(c("wt"))+stat_smooth(method="loess")

scale_x_continuous(breaks = seq(1, 6, 1), limits = c(1,6))

This will make tick marks appear on the x-axis equally spaced by 1 from 1 to 6. Axes units and labels can also be rotated or themed with the theme() function:

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

2.5 Titles and labels Titles and labels are key to storytelling in data visualization. To get you started, make sure that your labels are descriptive in nature and units are well

defined; titles should ideally encapsulate the main takeaway. Such captions are commonly used to provide a context or explain the data source that is necessary for reproducibility and credibility.

Formatting axes and labels turns your plot from a mere visualization of data to something that's ready for print, presentation, or inclusion in other graphics.

5.2.3 Adding Colors, Shapes, and Sizes to Plots

ggplot2 has strong capabilities to use colors, shapes and sizes that others do not have to make our graphics great in showing the data. They serve to encode more variables, discriminations categories and show patterns in the data.

Colors:

Representing variables by color also helps interpretation, particularly when comparing categories. For example: ggplot(mtcars, aes(wt, mpg, colour = factor(cyl))) +

geom_point()

In this example, cyl is set to color and ggplot2 creates different colors for each cylinder. For continuous variables, you can use gradient color scale:

scale_color_gradient(low = "blue", high = "red")

Shapes:

Shapes are helpful for differentiating groups when color is lacking, or if you're going to have a black-and-white rendering. The shape aesthetic gives different point symbols:

ggplot(mtcars, aes(wt, mpg, shape = factor(gear))) + geom_point()

Options are few, but shapes are simple for the case of several discrete variables.

Sizes:

Size is something that has magnitude or an important, usually for continuous variables: ggplot(mtcars, aes(wt, mpg, size = hp)) +

geom_point()

Here, point size is proportional to hp so that we can even add another dimension.

Combining Aesthetics:

ggplot2 can give you color, shape and size all at once:

ggplot(mtcars, aes(wt, mpg, color = factor(cyl), size = hp, shape = factor(gear))) +geom_point()

This forms a multi-dimensional visualization where weight vs mpg is plotted, cylinders are indicated in color, horsepower by size and gears by shape.

Be careful not to make it too crowded. Too many aesthetics will flood the reader, so choosing the most important ones is crucial. Through careful color and shape and size, analysts are able to effectively communicate complex information through a single plot.

5.3 Scatter Plots

5.3.1 Creating Simple Scatter Plots

Scatterplots are among the most low-level visualizations for understanding how two continuous variables are related. In ggplot2, they're created by geom_point(), which adds an x and a y value to the plot for each data point. This kind of visualization is useful in exploring correlation, clusters, outlier and general trend.

Basic Syntax:

ggplot(data, aes(x = variable1, y = variable2)) + geom_point().

E.g., plotting the car's weight against miles per gallon when I use the mtcars data: ggplot(mtcars) + geom_point(aes(x = wt, y = mpg)) You'll see that we get some strange distribution there.

geom_point()

This results in a scatterplot where each point is one car, with weight on the x-axis and miles per gallon on the y.

Customizations:

• Size of points: geom_point(size=3)

• Making multiple points more transparent to help overlap: geom_point(alpha = 0.6)

• Coloring points : geom_point(color = "blue"})

Scatter plots can also be made by mapping variables to color and shape: ggplot(mtcars, aes(x = wt, y = mpg, color = factor(cyl))) +

geom_point()

Here, cars are distinguished by the number of cylinders that they have -- adding interpretability to the plot.

Scatterplots are a common step in investigating bivariate data. They allow to determine whether relationships are linear, non-linear, or not existing which is a very solid ground for more sophisticated statistical models.

5.3.2 Adding Regression Lines and Smooth Curves

While scatter plots expose raw data patterns, overlaying trend lines augments interpretability and permits quantification of relationships. In ggplot2 scatter plots are automatically transformed into a line plot when we add a smoothing function using geom_smooth().

Basic Usage:

ggplot(mtcars) + aes(x = wt, y = mpg) + geom_point() +

geom_smooth()

By default, geom_smooth() will add a smoothed curve (LOESS on small datasetsGAM on large). This is consistent with the overall tendencies (T) and the more local excursions.

Linear Regression Line:

geom_smooth(method = "lm", se = FALSE)

• method = "lm" stands for linear regression fit.

• se = FALSE eliminates the shaded 95% confidence area around the line.

Customizations:

• Change color and type of the line: geom_smooth(method = "lm", color = "red", linetype = "dashed")

• Change smoothing span for LOESS: geom_smooth(span = 0.5)

Interpreting Regression Lines:

• Positive slopes indicate direct relationships.

• Negative slopes indicate inverse relationships.

• A confidence interval gives us information about the uncertainty.

Use Cases:

• In business analytics, regression lines on scatter plots are used to predict sales given advertising spend.

• In biology, they are used to estimate patterns of growth in relation to an environmental factor.

• In social sciences, they emphasize correlations like income vs. education level.

Overlaid regression lines (or curved trend lines) transform scatter plots into an analytical tool that combines visualization and statistical inference. This two-pronged viewpoint allows for both exploratory insight as well as formal hypothesis testing.

5.3.3 Enhancing Scatter Plots with Multiple Variables

Scatter plots can also visualize more than two variables by mapping other variables to aesthetic features such as color, shape or size. This multi-tasking ability makes them highly flexible in dealing with complicated datasets.

Example:

ggplot(mtcars, aes(x = wt, y = mpg, color = factor(cyl), size = hp)) + geom_point()

• Weight(wt) proportion is assigned to x-axis.

• The y axis maps to m.p.g.

• Color in the Figure is mapped to cylinder count (cyl).

• Size matters when it comes horsepower (hp).

This plot enables the analyst to identify relationships among as many as four variables at once; patterns such as group coming or size effect are directly visible.

Shape Mapping:

Categorical Variables with few levels? Shapes to the rescue!

ggplot(mtcars, aes(x = wt, y = mpg, shape = factor(gear))) + geom_point() General strategies for line styling In some cases you may just want to draw straight lines, this is done using the linetype argument.

Each type of gear has a unique shape.

Combining Aesthetics:

It is conceivably possible to have several aesthetics to display in a scatter plot, but be warned of congested plots. Such an overloading of variables may have the opposite (confusing) effect. Consequently, careful choice of descriptors is very important.

Transparency and Overplotting:

To avoid clutter due to overlapping observations when many variables make scatter plots dense,a package like ggplot2 uses transparency (alpha): geom_point(alpha = 0.5)

Faceting:

Another method of including more variables is with faceting, in which a separate scatter plot or set of scatter plots are made for each subgroup:

facet_wrap(~ cyl)

Did You Know?

"Scatter plots with multiple aesthetics are often described as 'bubble charts' when size is used as an additional dimension. This technique originated in economics and business dashboards, where visualizing three or four dimensions in one chart provides compact yet powerful insights."

5.4 Line Plots

5.4.1 Creating Line Plots

Line plots are a popular way of representing continuous data, especially when interest focuses on trends or patterns over the range of an ordering variable such as time, sequence, or measurement intervals. Line Plot: It is also quite easy in ggplot2 to make a line plot.s with x and y axis data as response variable. Basic Syntax:

Imagine a simple line graph for ggplot line graph plot as: ggplot(data, aes(x = variable1, y = variable2)) + geom_line() You can change the look and feel of these plots but it's nice that you can stick to a fairly constant structure.

For example, creating a plot with ggplot2 of unemployment over time using the economics dataset: > ggplot(economics, aes(x = date, y = unemploy)) +

geom_line()

This generates a line chart that represents trends in unemployment over the years.

Line plots are used differently from scatter plots in that they emphasize continuity, rather than discretely related points. Rather than display individual points, they indicate how one variable varies smoothly in relation to another.

Customizations:

- Highlighting observations by adding to points with geom_point().
- Varying thickness of the lines: geom_line(size = 1.2).
- Altering color: geom_line(color = "blue").

Line plots are good at not just time series but anything cumulative, growth rates, or scientific data that needs to be tracked over sequences. Line plots can also represent long-term patterns more clearly when coupled with smoothing layers, attenuating the effect of noise.

5.4.2 Plotting Time-Series Data

One of the most useful ways to leverage line plots is in time-series data analysis. When observations are taken at successive time intervals, data is time-series in nature; visualizing the data can highlight patterns such as seasonality-effects, long-term trends and short-term variations.

In ggplot2, if you want to graph times at all, your x-axis variable must be a date or time. Lubridate package to the rescue: R's lubridate package often makes it easier by spitting out character strings into true date objects, which ggplot can understand properly.

Example:

library(lubridate)

data$date <- ymd(data$date) ggplot(data, aes(x = date, y = sales)) + geom_line()

This gives me a line graph with time on the x axis and sales value on the y-axis so seasonal or yearly trends are easier to see.

Enhancements for Time-Series:

• Smoothing: Use, with geom_smooth(), a moving average or regression line to show trends overall.

• Multiple Series: Use color to indicate series membership within categories (e.g., sales by region).

ggplot(data, aes(x = date, y = sales, color = region)) + geom_line()

• ScalarscaleScale axes with scale_x_date() to have them formatted as months, years or quarters.

Applications:

• Finance: evolution of stock prices or exchange rates over time.

• Meteorology: temperature and rainfall trends.

• Business: monthly sales, website visitors or customers retained.

Because time-series visualization is the focus of very important indicators like cyclical features, anomalies and long-term growth or decay phenomena, line plot is irreplaceable in different application domains that consider temporal dynamics.

5.4.3 Customizing Line Types and Colors

Customization of lines can indeed be very informative and help to distinguish between multiple sets of data on the same chart. In ggplot2, you can customize line type, color and size using the same parameters but also change the transparency (alpha): You need to use the option alpha within the geom_line in order to define the transparency. Line Types (linetype):

The different line types (solid, dashed, dotted etc.) represent the data series. ggplot(data, aes(x=time, y=value, linetype=category)) + geom_line()

Each category is depicted by a separate line type which makes for readability on viewers in monochrome printing or presentation where no colors can be used.

Colors:

Using the categorical variable as a color enables automatic discrimination between various groups. ggplot(data, aes(x = time, y = value, colour = group)) +

geom_line()

Mimosa flowers, for example, are yellowish and can easy be changed with scale_color_manual() to a preferred corporate or publication colour.

Sizes:

The width of a line can be set using the size argument: geom_line(size = 1.5)

This is great when wanting to focus on a select series.

Combining Customizations:

These three aesthetics can be combined in a single line plot at the same time:

ggplot(data, aes(x = time, y = value)) + geom_line(aes(color = region, linetype = scenario), size=1.2)

This enables you to show more than one dimension of data in a single chart.

Best Practices:

• Do not overwhelm them by dense series plots, as it reduces interpretability.

• Leverage legends to tell the story, they should describe line types and colors.

• Select color styles that are colour-blind safe, and print-safe.

A well-thought out line customisation turns a basic chart into an informative visualisation, displaying nuanced relationships between groups of data without compromising interpretability.

“Activity: Creating and Customizing Line Plots for Business Analysis”

Creating and Customizing Line Plots for Business Analysis

In this activity, learners will use a retail dataset containing monthly sales figures for different product categories across two years. The task is to first plot overall sales trends using a simple line plot. Then, learners will enhance the visualization by plotting multiple product categories in different colors, adding a smoothed trend line, and experimenting with line types for seasonal versus promotional periods. The exercise will demonstrate how line plots not only highlight trends but also allow easy comparison of categories, supporting business decision-making.

5.5 Bar Charts

5.5.1 Creating Simple and Grouped Bar Charts

Barcharts are an ubiquitous visualization for categorical data comparisons. They plot categories on one axis and values that correspond to those categories on another; the rectangles actually reveal the contribution of each category represented in a rectangle. We use geom_bar() and geom_col() to create bar charts with ggplot2. Simple Bar Charts:

A basic bar chart shows amounts or frequency of categories. By using geom_bar() without specifying stat, it automatically counts number of occurences.

ggplot(data, aes(x = category)) + geom_bar() # labels still not shown

This creates bars based on the value of the number of records for corresponding categories.

For data in which the values have already been aggregated, etc., I would advise using geom_col() instead as it's easier because you pass y-values directly rather than the lengths: ggplot(data, aes(x = category, y = value)) +

geom_col()

Grouped Bar Charts:

Clustered bar charts are used to compare subcategories across the main categories. This is done by adding another variable to fill or color and with the position = "dodge" argument:

ggplot(data, aes(x = category, y = value, fill = subgroup)) + geom_col(position="dodge")

That results in the side-to-side bars which are useful for comparing between subgroups.

Clustered bar charts are excellent when you have relative subcategory display comparison like survey or sales results. And the differences over time, region or product category could be well represented for analysts to summarize intra- and inter-groups differences conveniently.

5.5.2 Stacked Bar Charts

Stacked bars are an extension of ordinary bar charts where each category (each x value) is represented in the combined bar by sub-bars stacked one on top of each other. Rather than grouping subgroup bars next to each other, stacked bar charts stack them on top of one another (or the side), with total height (or length) indicating overall value and color encoding additional subcategories.

Basic Syntax:

ggplot(data, aes(x = category, y = value, fill = subgroup)) + geom_col()

fill = when used with geom_col() creates stacked bar charts by default.

Advantages:

• They are effective at displaying totals and breakdowns of subgroups.

• How are proportions within a category being contrasted with each other?

Variations:

• Stacked Bar Charts (100%): Proportions are plotted as opposed to absolute amounts. That is done by using the argument position = "fill":

ggplot(data, aes(x = category, y = value, fill = subgroup)) + geom_col(position = "fill")

In this figure the height of each bar is normalized to 1 and the relative percentages of the different subgroup are indicate.

• Horizontal Stacked Bars: The order of the bars is not changed, but they are rotated using coord_flip() to make them easier to read when labels for categories are long.

Considerations:

Stacked bar charts are great for visualization of subgroup composition but can be difficult to interpret when there is an overload of subgroups. The proportional stacked bars perform quite mishw as long as you want to compare distributions and not absolute counts. Analysts need to decide whether they prefer grouped or stacked bars based on the trade off between a clear comparison of subgroups and an effective view of totals.

5.5.3 Customizing Bars with Color and Pattern

Customization of bar charts is important because it helps readers focus on the format and also have a consistent visualization matching with analytical needs or the corporate identity. ggplot2 has an enormous number of options for colors, patterns, and bar shapes.

Colors:

We can map colors to categorical variables via the fill aesthetic. For instance: ggplot(data, aes(x = category, y = value, fill = subgroup)) +

geom_col()

It automatically colors each subgroup differently. If you want to define the colors yourself, use scale_fill_manual(): scale_fill_manual(values = c("blue", "red", "green"))

Gradient color scales can be used for continuous variables with scale_fill_gradient() or scale_fill_viridis_c().

Patterns:

Aside from solid colors, bar fills can be embellished with patterns like stripes, dots or crosshatching using additional extensions such as the ggpattern package:

geom_col_pattern(aes(pattern = subgroup), pattern_fill = "black", pattern_density = 0.5)

Patterns work particularly well with grayscale or print-friendly charts where the color may not be distinguishable.

Additional Customizations:

• Scale width of bars: geom_col(width = 0.7)

• Adding outlines with color:

• geom_col(color = "black")

• Rotating x-axis labels for readability:

• theme(axis. text. x = element_text(angle = 45, hjust = 1)) tr(Output: The plot generated above is neat but there's a blank line position on the x-axis.

Did You Know?

"The use of patterns in bar charts became common before color printers were widespread. Today, even with advanced digital visualization, patterns remain crucial in

accessibility-focused design, as they help differentiate categories for colorblind audiences and ensure clarity in black-and-white printed reports."

5.6 Histograms

5.6.1 Creating Histograms for Frequency Distributions

Graph a Histogram A histogram is a picture of the distribution of values in a continuous variable. They divide up the range of data into intervals (called 'bins') and show you how many observations fall within each bin using rectangular bars. Unlike bar charts which represent value of a variable corresponding to each category, in a histogram you use the pencil tool and create a rectangle for each class.

In ggplot2 you need to do alittle more work but once you get the hang of things it's just as easy, if not easier In order to make a histogram in ggplot2 you have to use geom_histogram().

Basic Syntax:

ggplot(data, aes(x = variable)) + geom_histogram() Other plrors511 points out that looking only at $\mathbf{x}i$ rather than their relative positions in each group gives you nonsensical results.

ggplot2 will attempt to set it automatically, but sometimes that may not be what you want. Example: ggplot(mtcars, aes(x = mpg)) +

geom_histogram(binwidth = 2, fill = "blue", color= "black")

This makes bins of width 2 for the miles per gallon variable. the fill parameter is responsible for the color of the filled part of bars, while as we can see color adds borders.

Histograms help analysts understand:

• The form of the distribution (normal, skewed or uniform).

• Outliers and high values.

• If the data is symmetric or skew.

• Patterns such as multimodality, in which there are multiple peaks.

Exploratory data analysis can be significantly informed by histograms. For example, in new finance, they show the distribution of returns; in health care- -age distributions of patients and in education– score distributions.

Further fine-tuning can be done to add titles, theme and overlaying mean/median reference lines with geom_vline(). These tuning enhance interpretability when reporting insights to decision makers.

5.6.2 Adjusting Bins and Intervals

The decision of the bin size / number of intervals is quite important for meaningful histograms. The larger the bin size is, the more details are hidden; while the smaller the bin is, more information will produce noise for visual reading.

In ggplot2, there are two ways to control bin size:

By specifying binwidth:

ggplot(mtcars, aes(x = mpg)) + geom_histogram(binwidth=1) Here the bins are 1 unit of mpg:

By specifying number of bins:

geom_histogram(bins = 10)

This will bin 1-100 into 10 equal intervals.

Best Practices for Bins:

• Use the wider bins for detecting larger dataset to make visualization easier.

• Use thinner bins for smaller datasets to see small-scale structure.

• Try different bin-widths and see interesting things come to the surface.

It may help to rescale axes for readability as well. For example, scale_x_continuous(breaks = seq(0, 40, 5)) forces tick marks to be on the boundary of bins.

Overplotting on the other hand can be solved using transparency (alpha) or density counts line overlays added to the histogram. This blend is neither clear nor detailed.

In applied contexts, such as customer analytics, changes to the bin width can emphasize monthly or yearly purchase patterns. These small variations are also found in scientific data and fine binning might allow them to be made visible and relevant to experimental results. Therefore, cautious timing is not a technical question, but an interpretative one.

5.6.3 Density Plots vs Histograms

Whereas histograms show frequencies of raw data points that are in bins, density plots estimate the probability distribution using a smoothing function. They are each great things -- but they are not the same thing.

Histograms:

• Show discrete counts or frequencies.

• Depend on bin size and location.

• Good to represent the exact distribution of observations.

Density Plots:

• Created using geom_density().

• Consider the probability density function and determined that the area under it is equal to 1.

• Smooth out a curve that is close to the source distribution.

• Most appropriate for comparing distributions between groups.

Example:

```
ggplot(mtcars, aes(x = mpg)) +
geom_histogram(aes(y = ..density..), binwidth = 2, fill = "lightblue") +
geom_density(color = "red", size = 1)
```

It's basically a histogram with a density layer on top and maybe sounds more complicated than it is - but you have both the discrete counts as well as a smooth approximation.

When to Use Which:

• Discover general pattern in the raw data cases and interval frequences, presenting statistics for both bins and frequency as they are.

• Consider using density plots when looking across multiple distributions, or when the smoothness of patterns is more informative than the counts.

•In practice, combining the two is having your cake and eating it too.

For instance, in marketing histograms may indicate how many purchases are really made using each chip price and density plots can be useful to show sporting patterns of year. For scientific study, density plots can visualize the comparison of distributions between different groups in an experiment and may reveal details that are not obvious in the histograms because of their bin dependences.

The complementary use of histograms and density plots provides the analysts with both exactness and smoothened view for better interpretability and decision making.

5.7 Faceting for Multi-panel Plots

5.7.1 Introduction to Faceting with facet_wrap()

Faceting is a feature of ggplot2 which allows to generate multiple panels for the same plot with each panel representing a level of one or more factors. Unlike overlaying and stacking, the facets do not share the same space. Rather than one graph, we have ngraphs where n is the number of levels in the variable. This allows for more clear comparison and less clutter, if you are working with many categories.

facet_wrap() This is the most used faceting function. It places plots next to one another on a grid, with one row of the panel for each category.

Basic Syntax:

ggplot(data, aes(x = var1, y = var2)) + geom_point() + And if you want to increase dot size: ggplot(data, aes(x = var1, y = var2)) + geom_point(size=3) +

facet_wrap(~ category)

This generates a set of scatter plots for each category level in a grid layout.

Customizations:

- The ncol or nrow parameter defines the number of panels displayed per row or column.
- Panel labels are customizable with the labeller argument.
- Scales can be fixed (default) or free with scales = "free", by which each panel will set its axes independently.

Advantages:

- Readability has been enhanced when users compare more than one category.
- Show patterns that can be obscured in composite plots.
- Appropriate for comparisons with categorical data when overlaid bars would be confusing.

It is especially effective to faceting with facet_wrap() if you have a categorical variable with many levels, where they can observe trends, relationship or the distribution across the different groups.

5.7.2 Using facet_grid() for Comparative Visualizations

Facet Wrap function facet_wrap(): creates small panels created for one variable Facet Grid function facet_grid(): similar to facet_wrap(), but it allows faceting on more than two variables. This returns a faceted grid of plots, determining one variable by rows and another by columns. This allow to compare between several dimensions in an order.

Basic Syntax:

ggplot(data, aes(x = var1, y = var2)) + geom_point() + scale_color_continuous(low = "red") #element_blank())

facet_grid(row_var ~ col_var)

This creates a grid of scatter plot and the row_var sets the rows, col_var sets which column.

Features of facet_grid():

• Organized comparisons: A manipulation of a row-column arrangement of panels such that users can see the visual interaction between two variables.

• Blank cells: Where some combinations of Row and Column variables are missing from the data, these grid spaces remain blank – which can be useful for detecting Missing Data patterns.

• Scales: Like facet_wrap(), the scales argument can modify axis scaling.

Example Use Case:

Facet gird on one hand has been successfully used for panel data analysis such as in a dataset of exam scores it would create a grid with rows for gender and columns for subject. This allows concurrent comparison of male and female performances between subjects.

Advantages:

• Facilitates multi-dimensional comparisons.

• Cuts complex data into manageable pieces.

• Prevents overlapping of too much data which can hide group-level patterns.

facet_grid() is most useful when you have a dataset in which two categorical variables interactively break the data down into structured rather than comparative components.

5.7.3 Practical Applications of Faceting in Data Analysis

Faceting is more than a technical feat, it's one of the most potent and complex analysis tools in applied data viz. It adds definition by organizing data into groups, and enables analysts to measure trends, side-by-side, in similar, orderly fashion.

Applications Across Domains:

• Business Analytics: Compare sales behavior between product categories and regions at the same time with faceted line plots.

• Healthcare: See how patients fare under various treatments and age groups to determine which demographic fares better.

• Learning and education:Compare performance in different subjects, as well as by grade level to identify curriculum strengths and weaknesses.

• Social Sciences: Analyze survey results in relation to a variety of factors such as gender, income or geography.

Faceting vs Other Techniques:

Faceting differs from color or shape as a way to depict subsets, because it ensures the categories are visually separated (in panels). This is particularly useful if you have more than three groups, you can end having different colors or shapes, and it may get hard to interpret.

Advanced Use:

Faceting also can be use along with themes and scales to programme your individual reporting outputs. For dashboards and reports, where side-by-side comparisons are needed for decision making, analysts frequently rely on facet_wrap() or facet_grid().

Faceting makes sure you don't miss the forest for the trees by carving your data into manageable views. It does wonders for storytelling to see both the larger perspective and those details at an ensemble level.

Knowledge Check 1

Choose The Correct Options :

1. Which function is used to facet by a single variable?

a) facet_grid

b) facet_wrap

c) facet_one

d) facet_panel

2. What does facet_grid() allow?

a) Single category only

b) Two categories

c) Only continuous data

d) Random grouping

3. Which argument allows independent axes in faceting?

a) scales

b) limits

c) axis_free

d) breaks

4. Which faceting method arranges plots in a wrapped layout?

a) facet_grid

b) facet_wrap

c) facet_table

d) facet_row

5. What happens in facet_grid() if a combination of row and column variables is missing?

a) It is ignored

b) Error occurs

c) Empty panel shown

d) Axis removed

5.8 Summary

ϖ The ggplot2 is a library that is based on the Grammar of Graphics which conceptualizes plot as Aesthetics Data + Geoms + Statically defined plots including position, scale, facet.

■ Each and every ggplot command is made up of layers, with the initial layer being ggplot(), followed by successive geom_* functions for visual elements.

ϖ Aesthetics (aes) is the mapping between data variables and graphical properties, such as x-axis, yaxis, color, shape and size.

ϖ Scatter plots are able to investigate the relationship between two quantitative variables, sometimes also importing regression lines or multiple variable mappings.

ϖ Line plots are useful for showing trends especially sequential data with the facility to change line types, colours and sizes.

ϖ Bar Charts: Bar charts depict categorical data and take different forms like grouped, stacked or proportional stacked bars.

ϖ Histograms demonstrate the frequency distribution pattern for continuous variable using the bin size in order to know about the density and pattern.

ϖ Density plots smooth approximations to the distributions, which can be overlapped with histograms for more thorough investigation.

ϖ Multi-panel visualization with facet_wrap() or facet_grid() Faceting can also be used to separate your data by categories, bu which inside give you easy comparisons.

ϖ Ggplot2 customization – labels, titles, themes and scales allows visualizations to be informative as well as professional looking.

Utilization of color, patterns and faceting is used effectively to improve the accessibility, legibility and interpretability.

ϖ ggplot2 allows us to incorporate many variables and layers, making raw data into powerful tools for both finding and telling stories.

5.9 Key Terms

ggplot2 – A data visualization library for R, based on the Grammar of Graphics.

aesthetics (aes) – The data variables to map to visual properties such as x, y, color and size.

Geometries (geom) – Functions that specify what type of plot you want, e.g. geom_point(), geom_line()

Layers - The parts of a plot that are added on top in sequence to build the complete graph.

Scales – Functions that map data values onto visual output.

Themes – Functions that change non-data visual elements such as background, grid lines and text.

Scatter Plot – The points that represent the value of two continuous variables.

Line Plot – A graph that connects data points with a line to display trends across order or time.

Bar Chart – A graphic that shows categories on one axis and uses bars to represent them.

Histogram – A graphic representation of frequency distribution for a continuous variable, based on bins.

Density Plot – A continuous curve which is used to estimate the distribution of data.

Faceting – Dividing the data into several panels, for comparison across categories.

5.10 Descriptive Questions

What's the idea behind Grammar of Graphics, and how does it work in ggplot2?

What are "aesthetics" in ggplot2 and how does it differ from "geometries"? Provide examples.

Explainhow a scatter plot is created and customized in ggplot2.

Explain the importance of line plots when visualizing time series. Include examples of customizations.

Explain the difference between grouped, stacked and prensentage stacked bar plot with example?

What is the difference between histograms and bar charts? Illustrate with ggplot2 code examples.

Compare and contrast: density plots and histograms. When would you use each?

Facet_wrap vs Facet_grid in ggplot2? And Why?

5.11 References

1. Wickham, H. (2016). ggplot2: Elegant Graphics for Data Analysis. Springer.

2. Grolemund, G., & Wickham, H. (2017). R for Data Science. O'Reilly Media.

3. Kabacoff, R. I. (2015). R in Action: Data Analysis and Graphics with R. Manning Publications.

4. Chang, W. (2012). R Graphics Cookbook. O'Reilly Media.

5. Healy, K. (2018). Data Visualization: A Practical Introduction. Princeton University Press.

6. R Core Team (2023). R Language Definition. R Foundation for Statistical Computing.

Answers to Knowledge Check

Correct Answer For Knowledge check 1:

1. b) facet_wrap
2. b) Two categories
3. a) scales
4. b) facet_wrap
5. c) Empty panel shown

5.12 Case Study / Practical Exercise

Visualizing E-commerce Sales Data with ggplot2

Background:

An e-commerce business gathers sales data including monthly sales volume, products sold, customer types and regional performance. The management team are looking for visual pointers to see patterns, compare categories, and network customer behavior. Analysts rely on ggplot2 to visually explore the raw data and provide insight from it.

Problem 1 Looking at Customer Purchases with a Scatter Plot

The business needs to compare the age of the customer and average purchase amount. They also suspect that younger customers make smaller purchases and older ones spend more.

Solution:

A scatter plot is generated using customer age on the x axis and purchase value on the y-axis. Color is categorical encoded by customer type (i.e., regular, premium and new).

ggplot(sales_data, aes(x = age, y = purchase_value, group=segment)) + geom_point(aes(colour=segment), alpha = 0.6) +geom_line(aes(color= segment))+ ylim(0,max(sales_data$purchase_value)$1).

geom_smooth(method = "lm", se = FALSE)

The scatter plot displays clear clusters: premium customers at any age show more consistent higher spending overall, where as younger new customers tend to be lower spenders on average. The slope of the regression line support that there is a positive correlation between age and the purchase amount.

Problem 2: Region Based Trend Visualization via Line Plots Fet a data set from the file Donwload file fet you were provided and visualize it representing his trends region wise through line plots.

The regional managers request every month sales comparisons across regions to see how the staff does their work.

Solution:

Analysts – they plot time (months) on the x-axis, sales on y and map color to region with geom_line() 2.

ggplot(sales_data, aes(x = month, y = sales, color = region)) + geom_line(size=1.2) +

labs(title = "Monthly Sales Trends by Region")

From the line plot, we see that in the northern and western regions they have a consistent growth and in the eastern region is has seasonal peaks. Management sees the need for localized promotion in laggard regions in the slower months.

Problem Statement 3: Compare Product Categories with Bar Charts and Faceting Management asks you to analyze which product categories look the best in various customer segments. Solution:

Getting-started A stacked bar chart with product categories on the x-axis and sales values on the y-axis. Colour by customer segment. In (b)(c), further insight is provided by faceting for regions.

ggplot(sales_data, aes(x = category, y = sales, fill = segment)) + geom_col(position = "dodge")+ theme(axis.title.x=element_blank(), legend.position="none")

facet_wrap(~ region)

The visualization also tells us that electronic products are most popular among premium customers in the urban areas, whereas for young regular customers in rural areas clothing is more popular.

Reflective Questions

Q: Why are scatter plots better than tables for seeing relations between variables?

How are line plots better than bar charts for time series data?

When would you use a cumulative / stacked bar chart instead of a grouped bar graph?

How is faceting useful when comparing multiple dimensions of information?

As part of this translation, how does colour and shape customisation assist in improving the interpretability of visualisations?

Conclusion

This case study shows how ggplot 2 can allow e-commerce analysts to turn raw sales data into actionable information. Scatter plots revealed customer spending patterns by age and segment, line plots showed regional sales trends over time, while bar charts used faceting to illustrate product-category preferences among segments and regions. Using multiple visualization methods provided the company with a holistic perspective of the performance levers and facilitated data-driven decision-making. The exercise also demonstrates gpglot2's more traditional role in business intelligence: not just to depict data but to deliver discernible narratives informing strategic decisions.

BAR Unit 6 V3 (1).docx

ATLAS SkillTech University

Document Details

Submission ID
trn:oid:::3618:127346448

Submission Date
Feb 2, 2026, 10:49 AM GMT+5:30

Download Date
Feb 2, 2026, 10:50 AM GMT+5:30

File Name
BAR Unit 6 V3 (1).docx

File Size
216.4 KB

33 Pages

7,684 Words

45,490 Characters

0% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text
- Small Matches (less than 30 words)

Match Groups

0 Not Cited or Quoted 0%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

0% Internet sources
0% Publications
0% Submitted works (Student Papers)

Integrity Flags

1 Integrity Flag for Review

Hidden Text
212 suspect characters on 6 pages
Text is altered to blend into the white background of the document.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **0** Not Cited or Quoted 0%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

0% Internet sources
0% Publications
0% Submitted works (Student Papers)

Unit 6: Descriptive & Inferential Statistics in R

Learning Outcomes:

1. Explain the role of descriptive and inferential statistics in analyzing and interpreting business data.

2. Compute and interpret measures of central tendency and dispersion to summarize datasets.

3. Differentiate between descriptive and inferential statistical techniques, and identify appropriate contexts for their application.

4. Formulate and test statistical hypotheses using appropriate hypothesis testing methods, including t-tests, and interpret p-values and confidence intervals.

5. Apply correlation analysis to assess the strength and direction of relationships between two or more variables.

6. Analyze real-world business problems using statistical tools, as presented in the caselet and case study sections.

7. Interpret and communicate statistical results effectively to support data-driven decision-making in business contexts.

Content:

6.0 Introductory Caselet

"Making Sense of Sales – A Data Dilemma at Vistara Retail"

Vistara Retail, a mid size consumer electronics retail chain has observed varying sales rebustness among its different regional stores in the last fiscal year. Although a few urban stores did great, semi-urban and rural markets saw their numbers go down. The managers, who were alarmed about these discrepancies, decided to implement this evidence-based project to audit possible roots and strategies for improvement.

Introducing Riya Sen, the brand new Business Analyst who had to make some sense out of the huge pile of sales data that they collected over a period of 1 year. She started by collating the data based on various parameters – region, product category, monthly sales values, customer footfall and the schedules for promotional campaigns.

Her initial task was summarizing the data to find trends. She averaged sales by month, noted the most popular product categories and found months with high and low sales. But Riya soon learned it wasn't enough to depend on averages alone. Two shops could be making the same average sales (or profits) but with hugely different variability, meaning volatility in one and stability in another.

For additional insights, Riya used measures of spread and went on to make inferential statistical comparisons between the regions. She constructed hypotheses to test if increase in promotional campaigns was statistically significant for sales. Through additional correlation analysis, she tried to uncover whether customer traffic behaviour was related to how product categories were doing.

At the end of her preliminary study, Riya had been able to convert raw numbers into insights - providing management with the hard statistical evidence it was looking for in order to make informed decisions. But, she still struggled to communicate the often complicated statistical findings to a non-technical audience.

Critical Thinking Question:

How can Riya make certain that the results of statistical analysis remain statistically valid and interpretable for executive decision-makers without a statistics background?

6.1 Descriptive Statistics

6.1.1 Concept and Importance of Descriptive Statistics

Descriptive statistics is a range of statistical tools and techniques which are used to summarise, describe or display data in a meaningful way. Descriptive, not inferentialUnlike inferential statistics where you try to make inference about a population based on the sample, description is only concerned with what you have. It's not about predicting it, or making generalisations—but rather cutting through to clear insight on what the data shows.

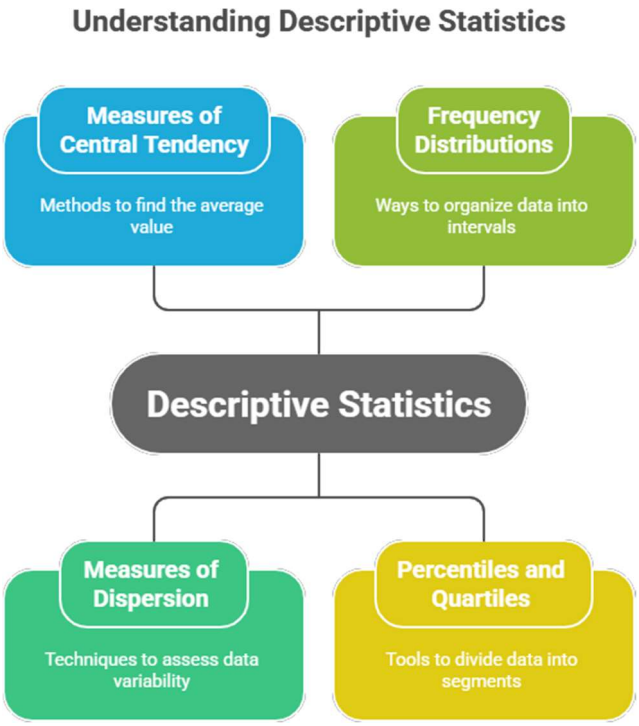

Figure 6.1

Descriptive and basic statistics Descriptive statistics include the following: The description consists of summarizing and redistributing a rhe data in order to show trn - 1} * f(x.

• Central Tendency: These are measures that describe what is typical or average in a dataset. Typical measures are the mean (average), median (middle value) and mode (most occurring value).

• Measures of Dispersion: They indicate to what extent the data points are scattered about the central value. Some common metrics are spreading, spread variance and standard deviation.

• Frequency Distributions: These show exactly how often each value (or range of values) appears. Typical visualizations include frequency tables, histograms and bar charts.

Percentiles and Quartiles: These breaks data into sections, which can assist in identifying the distribution and outlying data.

The role of descriptive statistics in data analysis is incomparable:

• First Pass Information Gain: Descriptive statistics will provide information gain even before modeling or inferential analysis, enabling researchers and analysts to understand the shape and features of data.

• Data Cleaning and Quality Check: The overview statistics are used to trace missing values, data anomalies and discrepancies.

• Reporting of Outcomes: Descriptive statistics expose information in user-friendly summaries and can be used to communicate results to individuals who may not have a technical background.

Business decisions Descriptive statistics is typically the first set of data analysis you would perform when delving into customer sales reports, employee performance reviews, or just trying to make any sense of all the raw data collected from your market research. Real simple: if you know the MMS, its spread (variability in monthly sales) and at which months peak sales happens, you can do solid strategic planning without any predictive models.

Descriptive statistics can be a precursor to inferential stats, too. For example, the dispersion in a sample may help guide us on which inferential tests to use. Therefore, it underlies deeper statistical understanding.

6.1.2 Summarizing Data with R Functions

R is a highly extensible statistical programming language used widely for data analysis. It offers a variety of in-built functions to quickly summarize and manipulate data. Data summarising is crucial to allow for a better understanding of data distribution, typo identification and preliminary work-up towards further statistical analysis. In R, you tend to summarize your data with summary statistics, quick check for missing values, plot distributions.

Some basic aggregation functions to summarize data in R are:

summary(): This R base function gives you a summary for the minimum, 1st quartile, median, mean, 3rd quartile and maxiumun for each variable of your dataset. It works well with numeric data and can also summarise factors by displaying frequency counts.

• str(): Not a Statistical summary as such, but str() is a good way to see what structure your data frame has including (variable types and sample records), so you know what sort of summaries to apply.

• table(): The table() function is for creating frequency tables, which helps to display the tabular representation of categories (qualitative variable) in a dataset. For instance the count can be to display the number of observations per category.

• mean(), median(), sd(), var(): These are functions which calculate certain statistics for quantitative variable such as the mean, median, standard deviation and variance.

• quantile(): It gives you percentiles; it's very important for analysing the spread and outliers in your data.

• range(): its only argument is a vector or column, and it returns the minimum and maximum value of that construct, so that analysts can instantly perceive the span of an expression.

•length() and nrow(): These can be helpful when you want to get the size of your data set or count observations, which may factor into what kinds of statistics you'll use.

More libraries like dplyr and data. table add additional power for summarising grouped data with the group_by() and summarise() functions. These are particularly helpful when working with big data sets and many categorical groupings.

For example, a retail analyst might use group_by(region) %>% summarise(avg_sales = mean(sales)) to calculate average sales per region, an important step in identifying regional differences in performance.

R - Aggregate the Raw Data to See Trends Summarizing data in R is more than just calculating some numbers and that's why make the distinction between summarising (i.e. filtering out top groups/ summary values etc.) and aggregating - it's because you're not just losing rows of raw data with aggregation, but changing your unit of observation all together! It prepares the way for recognizing patterns, drawing comparisons, and primers more advance statistical procedures such as hypothesis testing and regression.

6.1.3 Using summary() and describe() Functions

Two popular data summarization functions within R are the summary() and describe() functions, both with their own advantages and use cases. These metrics offer summary but meaningful insights into the data, and help in better understanding the variable-level characteristics.

summary() Function (Base R)

The summary() function is a base R function that processes x, where x could be any type of equally-spaced vector, data frame or factor. When used with a data frame, it provides a summary for each variable column wise. Features:

• For numeric variables, it returns:

o Minimum

o 1st Quartile (25%)

o Median (50%)

o Mean

o 3rd Quartile (75%)

o Maximum

• For factors or discrete variables, it returns the histogram of each category.

• For boolean type variables, it will return the counts of True, False and NA values.

This function comes in handy when you want to eyeball all columns of a dataset with just one line of code! However it does not return any other statistical measures like standard deviation or skewness.

describe() Function (psych Package)

The describe() method is part of the psych package and can be used for more descriptive statistics such as creating summaries for numeric variables. It reports as richer metrics than those provided by summary().

Some of the summary statistics extracted from describe() are:

• Mean

• Standard deviation

• Median

• Minimum and maximum

• Range

• Skewness

• Kurtosis

• Standard error

Such extended output is especially useful in statistical analysis because it provides details on distribution shape (skewness and kurtosis) and dispersion (standard deviation).

Usage:

To use describe(), the psych package must be installed and loaded: install. packages("psych")

library(psych) describe(data)

This method is best used for when the dataset needs to be deeper diagnosed. For instance, analysts conducting regression or hypothesis testing typically use the describe() function to validate assumptions (e.g., normality and variance) while exploring these analyses.

Comparison and Use Cases:

• Use summary() to get a quick high level view of the dataset (especially if you're working with categorical and numberic variables).

• Use describe() when you want to extract general numerical information so that you can understand, for example, the shape of the distribution or compare spread between variables.

Using both of them jointly in the early stage of data exploration will be complementary to each other. For instance, in a consumer satisfaction survey, summary() might inform you that the mean score is 4.2 but describe() might suggest to use extreme value treatments due to the severe right-skewness of your data (meaning 4.x answers are quite close together but some participants rate things as 9 or even higher).

6.2 Measures of Central Tendency

6.2.1 Mean in R

The average or mean is the sum of all measured values divided by quantity. It is widely used because of its simplicity and its effectiveness in describing data that approximate the normal distribution.

In R, computing the average is simple with the built-in function mean(). The syntax is: mean(x, na. rm = FALSE)

• x is the value vector.

• na. rm = TRUE will automatically drop the NAs before taking the mean.

For example:

sales %

group_by(category) %>%

summarise(mean_value = mean(variable, na. rm = TRUE))

In turn, this allows comparisons between different data segments or categories.

6.2.2 Median in R

The median is the value that separates the dataset into two equal halves, in which all numbers are arranged in either ascending or descending order. It would be especially helpful if our data is skewed or contain outliers as it does not use the mean.

The median in R is calculated by using the function median():

median(x, na.rm = FALSE)

• x is the numeric vector.

• Setting na. if rm = TRUE is the parameter value, missing values are eliminated.

For example:

scores %

group_by(region) %>%

summarise(median_sales = median(sales, na. rm = TRUE))

This kind of thing is used all the time for exploratory data analysis and reporting, especially when we are contrasting data between categories or across periods of time.

The median is also central to box plots, which display the distribution of the data and identify any outliers, containing the value of the quartiles as well as their median.

6.2.3 Mode in R

The mode is the value that occurs most often in a set of data. It is the only measure of central tendency that can be used with nominal data (the other two can't), and it's especially useful when you're working with non-numeric or discrete variables.

Seems like R decides to leave us hanging, since unlike the mean and median, there is no mode() built-in function in R for statistical mode estimation. The mode() function in base R means the storage type of an object, and not the statistical mode.

In order to calculate the statistical mode in R, a function is usually defined: get_mode <- function(x) {

uniqx <- unique(x) uniqx[which. max(tabulate(match(x, uniqx)))]

}

This function works by:

• Identifying all unique values

• Counting their frequencies using tabulate()

• Returning the mode value

For example:

x < - c(2,3, 3,5,7 ,3,2,5,2)

get_mode(x)

This would give 3 as the result, if the most common value is three.

For data that have multiple modes (bimodal or multimodal), it can be adapted to return all value(s) with the maximum frequency. Use cases It's especially helpful:

• Product specific market research (for ex., preferred product)

• Customer reviews (i.e., average customer rating)

• Retail analysis (i.e., most purchased item)

In categorical data it will locate dominant categories or preferences, and can provide actionable business insights as well as useful information for researchers.

The draw back of mode is that it may not occur in continuous data, e.g., the requirement that each value be unique, and in these cases contributes little information.

6.2.4 Comparing Mean, Median, and Mode

It is important to know variations and usefulness of mean, median, mode so as to use the appropriate measure of central tendency. It is shown that the two measures have distinct advantages and one or the other should be used in a given data condition.

Sensitivity to Outliers:

• 'Mean' is very prone to outlier values. one extreme value can hugely bias the average.

•Median is less sensitive to skewed and extreme values. It provides a more robust measure of central tendency if the data is not normal.

• Mode is resistant to numerical values and instead indicates frequency. It is also applicable to categorical data.

Data Type Suitability:

• Mean needs to have numerical data and a statement of interval (or ratio) scale.

• Median is applicable for ordinal, interval and ratio data.

• Mode applies to all type of data including nominal, ordinal and categorical.

Distribution Shape:

• In a symmetric distribution, all three will be equal or approximately so for the normal distribution.

• Order in positively skewed distribution: Mode < Median < Mean.

• In the negatively skewed distribution, arrangement is: Mean < Median < Mode.

Practical Use Cases:

• An average of income, temperature etc. constitutes continuous data and mean is practical in such conditions.

• Use the median to analyze housing prices, salaries, and other skewed data.

• Mode can be useful to find the most popular data or item in a survey.

Interpretation and Communication:

• Mean is familiar to stakeholders, but potentially misleads in non-symmetric distributions.

• A median is simpler to express when you describe the midpoint discoveries.

• Mode should be used when frequency (occurrence) is more important than numerical value.

In practical data analysis, one usually wants to calculate all three measures. It can be useful to compare them to evaluate the distribution form and whether you require more complicated statistical methods. The difference between mean and median for example can indicate skewness or when outliers might be present to guide additional investigation.

6.3 Measures of Dispersion

6.3.1 Variance in R

Variance is the average of the squared distance between each point and the mean. It measures by how much the values in a set vary from the average value for that set, which shows how widely spread they are. A greater variance means the data points are more spread out, while a lower one indicates they are concentrated.

The formula calculates the variance (sample) as follows:

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

In R, standard deviation is calculated using the sd() function:

data % spread(rim) %>% gather(key_var, value_data,-rim) # should give a KeyVar Value newData % mutate(group_sd = sd(Value, na.rm = T))=output: rim key_var value newData group_sd 1 C Chl 13.2 6.35085349400645 3 C Sendiment 18.9 NA I tried to run "sd=c()"" instead of the na rm, but with no succcess.

group_by(group_var) %>%

summarise(sd_value = sd(numeric_var, na. rm = TRUE))

Standard deviation, along with mean and other descriptive statistics are used to understand the behavior of data as a whole.

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

6.3.3 Range and Interquartile Range

Range and IQR are descriptions of spread or extent of the Results The dataset but involve different parts of the distribution.

Range

The simplest measure of spread is the range. It is simply the range of values, for example a dataset:

Range=Max(x)−Min(x)

In (R, it can be calculated as:

range(data) diff(range(data))

The range is a simple to calculate, and to estimate however it quite sensitve to outliers. One extreme can heavily impact the range, which is unsuitable for skewed or noisy datasets.

Interquartile Range (IQR)

IQR is a measure of data spread that quantifies the difference between the 25th and 75th percentiles as follows:

IQR=Q3−Q1

Where:

• Q1 (Quartile 1) has the 25th percentile

• Q3 (third quartile) which is the 75 percentile

In R we calculate the IQR as:

IQR(data, na.rm = TRUE)

IQR is resistant to outliers and is a resist measure of spread, especially for skewed distributions. It is also used with the median, since both are robust statistics.

The quartiles can also be calculated with:

quantile(data, probs = c(0.25, 0.75)) Application of IQR:

• Detecting and filtering outliers: values below Q1 − 1.5×IQR, or above Q3 + 1.5×IQR

• Assessing spread in skewed distributions

• The Variability Of Medians Between Groups

Boxplots and robust statistical modelling are built on this foundation of IQR.

6.3.4 Visualizing Dispersion with Boxplots

Boxplots (also known as box and whisker plots) are a graphical way of showing the distribution, central values and the spread of data. And when they want a quick one-liner summary, they can get that using five-number summary: minimum, Q1, median, Q3 and maximum.

In R, a boxplot can be made with the boxplot() function: boxplot(data)

To compare across groups:

boxplot(numeric_var ~ group_var, data = dataset)

Elements of a boxplot:

• The box displays the IQR (Q1 to Q3)

• A line in the box marks the median

• Whiskers reach to the minimum and maximum within 1.5IQR

• Outliers (Dots or asterisks that are outside the whiskers) Boxplots are incredibly useful for:

• Identifying skewness: The data is skewed if the median is near Q1 or Q3.

• Identifying outliers

• Comparing distributions across groups

• Understanding spread and symmetry

Advantages of boxplots:

• Summarize large datasets visually

• Useful for comparison of several groups at the same time

• Discover phenomena that are difficult to perceive using only numerical summaries
Limitations:

• Omit to illustrate the form of distribution in detail (e.g., bimodal distributions)

Less suitable when the sample size is really small In real life, boxplots are particularly useful in:

• Quality control: identifying process anomalies

• Medicine: comparing healthcare treatments across groups

• Educational: comparing score distributions across schools or sections

Boxplots can also be pimped with ggplot2 or such by adding color, faceting, and more layers to reveal a richer meaning.

6.4 Inferential Statistics

6.4.1 Concept and Role of Inferential Statistics

Inferential statistics is a set of tools for the making inferences about population parameters on the basis of data samples. Because it is rarely feasible or even possible to collect information on an entire population, inferential statistics gives us the means for assessing and estimating characteristics of a population based on sample data.

Concepts Important to Inferential Statistics These concepts above are relevant to inferential statistics:

• Population and Sample A population is the entire collection of people or items to be studied, whereas, a sample is a portion of the population that we actually collect data on.

• Parameter and Statistic: A parameter is a numerical measure of a population characteristic, while statistic is a numerical measure of the attribute of a sample that can help us in estimating the characteristics of the population.

• Random Sampling: A sample in which each member of the population is just as likely to be chosen. This assists in obtaining a representative sample.

• Probability Distributions – These provide information on how the values of a variable are distributed, and they are important in making sense of probabilities.

• Estimation and Hypothesis Testing: Two purposes of inferential statistics, created in order to predict and evaluate claims concerning population parameters.

In this case, inferential statistics are very important because it allows us to make decisions. Partial information is typical in applications like business, health, economy and social sciences when a decision process needs to be made. Thus a drug company could rank with inferential methods whether a new drug work based the trails on the sample of patients.

Inferential inferences rely crucially on the quality of sampling and the statistical methods used—in this case based assumptions. This is why the correct way to sample, think about variability and know the properties of distributions are the cornerstone to any inferential study.

6.4.2 Sampling and Estimation in R

Sampling is a selection of some account units or elements from the general population to approximate features of it. Estimation refers to estimating population parameters (such as mean or proportion) using sample statistics. In R, there are numerous methods and functions for random sampling and estimation that make it an excellent tool for inferential statistical analysis.

Random Sampling in R

For random sampling, one uses the sample() function:

sample(x, size, replace = FALSE, prob = NULL)

• x is the population vector

• size: the number of samples to draw

• replace is per-sample replacement or not.

For example:

population <- 1:1000

sample(population, size = 100, replace = FALSE)

This will randomly sample (without replacement) 100 observations from a population of 1000.

Stratified sampling, cluster sampling and systematic sampling can also be conducted with the use of further logic or special packages (eg. sampling/survey).

Estimation of Population Parameters

When given a sample, one can use simple functions to estimate parameters such as the mean, proportion and standard deviation:

mean(sample_data) sd(sample_data)

The standard error of the mean (SEM), an estimate of the variability of the sample mean, is calculated as:

$$\mathrm{SEM} = \frac{s}{\sqrt{n}}$$

Where:

• s = sample standard deviation

• n is the sample size

This can be computed in R as:

sem <- sd(sample_data) / sqrt(length(sample_data))

This value is used in the construction of confidence intervals and hypothesis tests. Sampling and estimation are the heart of inferential statistics as it permits generalizations with limited data. Such applications include predictive modeling, public health studies, marketing research and policy analysis.

Did You Know?

"Even a small, well-chosen sample can provide accurate insights into a population—statistical theory shows that a random sample of just 1,500 people can estimate national preferences within a few percentage points, regardless of population size."

6.4.3 Confidence Intervals

intervalA Confidence Interval (CI) is a range of values based on sample data that is believed, with a certain level confidence, to contain the true population parameter. Confident Intervals provide an estimate of how uncertain a sample estimate might be. They are central to any inference statistic.

Understanding Confidence Intervals

The confidence level we are using is 95%97{"C" \mathsf{LL} } and if we took repeated samples, then approximately 95% of these intervals would include the true population parameter.

The confidential interval for mean has a general structure as follows:

$$\bar{x} \pm Z\left(\frac{s}{\sqrt{n}}\right)$$

Where:

• $\bar{x}$ is the sample mean

• Z is the standard value from normal distribution (e.g., 1.96 for 95%)

• s = sample standard deviation

• n is the sample size

Calculating Confidence Intervals in R

A "hand-drawn" 95% CI can also be created as follows:

x_bar <- mean(sample_data) s<- sd(sample_data)

n <- length(sample_data) error_margin <- 1.96 * s / sqrt(n) lower_bound <- x_bar - error_margin upper_bound <- x_bar + error_margin c(lower_bound, upper_bound)

This range contains the true population mean with 95 per cent confidence. Or there are built-in R functions for calculating 'other' confidence intervals, possibly on the fly. Both one-sample and two-sample confidence intervals can be estimated using the t.test() function: t.orderName(t.test(sample_data)

This will give you the confidence interval along with sample mean and sd.

Applications and Interpretation

Confidence intervals are used in:

• Business forecast: prediction of future business revenues within a confidence interval

• Clinical: Establish the plausible limits of therapeutic effect

• Survey inference: Bounding properties of some population opinions or behavior

Note that if a parameter is in the confidence interval you cannot conclude, with statistical certainty, the parameter is within or outside of the interval; but only that it had a certain probability of being inside (based upon what we know from computing confidence intervals).

Confidence intervals also inform the process of decision making in conditions of uncertainty. For example, if the 95% CIs of two rivaling strategies do not overlap this indicates a statistically significant inferiroity or superiority.

6.5 Hypothesis Testing

6.5.1 Steps in Hypothesis Testing

The process of hypothesis testing consists of the following structured steps in order to ensure a rigorous and objective use of statistics. These steps can help to direct the researcher through the test design, analysis and reporting phases.

State the Hypotheses:

o State the null hypothesis (H_0) and the alternative hypothesis (H_1 or Ha). The null is usually the absence of any effect or difference while the alternative, by definition, is what you are trying to show.

Set the Significance Level (α):

o Select a critical value for rejecting the null hypothesis. Common values are 0.05 (5–%%), 0.01 (1%) or 0.10 (10%). This is the chance of a Type I error (such as rejecting the true null).

Select the Appropriate Test:

o Select a statistical test (e.g., z-test, t-test, chi-square test, or ANOVA) based on the type of data for comparison and how distribution and sample size.

Calculate the Test Statistic:

o Use sample to calculate test statistic (t/z score) proliferative than the null hypothesis. This is a measure of how much the sample statistic diverges from our null hypothesis.

Find the p-value or Critical Value:

o P The P -value is the probability of obtaining the sample data if there is no effect (H0). Or, one can compare the test statistic to critical values in statistical tables.

Make the Decision:

o If the p value is less than 0.05 ($p < \alpha$), then you reject the null hypothesis. Otherwise, fail to reject it.

Interpret the Results:

o Interpret the statistical decision in this scenario in terms of the research question.

These procedures underpin data-guided decision making and are essential in examining experimental findings, evaluating business actions and informing policy advice.

6.5.2 Null and Alternative Hypothesis

The concept of null hypothesis (H_0) and alternative hypothesis (H_1) can be considered as the fundamental concepts in hypothesis testing. They embody competing

hypotheses concerning a population parameter and determine what type of statistical analyses can be conducted. Null Hypothesis (H_0):

Its null hypothesis is there being no effect, no difference, or no relationship between these variables. It is used as the starting position or standard, and the test attempts to challenge or refute that position.

Examples:

- H_0: The average of customer satisfaction score is 7.

• Compare two regions = Comparison of sales in two areas to test: H_0: The average sales is not different for the two regions.

• H_0: The new drug is the same as the previous one in terms of recovery time.

By rejecting the null hypothesis, we are saying, according to our data at least, there is enough evidence for us to say it's not true.

Alternative Hypothesis (H_1 or Ha):

The null hypothesis is the one that attempts to disprove. It suggests that there is a large effect/difference/relationship in the population.

The alternative hypothesis can be:

• Two-tailed: Because possible difference in either direction (≠) is tested

• One-tailed: Looks for differences in one direction (+, -) Examples:

- H_1: The average belief score is not 7 (two-tailed)

• H_1: The mean sales in Area A are higher than the mean sales in Area B (one-tailed)

• H_1: The new drug is more effective at reducing recovery time (one-tailed)

This underlines the importance of clear and accurate hypothesis definition, since they determine the test used, the hypothesis testing direction and interpretation.

Key Considerations:

• The null hypothesis is not “accepted” but rather fails to be rejected when weaker evidence is in hand.

• The conclusion always goes back to the null hypothesis in some way — it’s either that we reject it or that we don’t.

• Errors can occur:

o Type I Error: Ho is true but rejected.

o Type II Error: Accepting false H_0

Well-defined hypotheses contribute to the objectivity and replicability of science, experiments in business.

6.5.3 p-value and Statistical Significance

P-value is an essential factor in hypothesis testing, because it tells you whether the null hypothesis should be accepted or rejected – p-value defined as: the probability of obtaining the observed results (or more extreme) if the null hypothesis were true. It quantifies the extent of evidence in data for rejecting the null hypothesis.

Interpreting the p-value:

- For values of p are very small -usually less than 0.05- the null hypothesis can be rejected.
- A high p-value indicates weak evidence against the null hypothesis, so you fail to reject the null hypothesis.

For instance, a p-value of 0.03 is interpreted to mean that it occurred with 3% chance under the application of the null hypothesis. The difference is statistically significant if α, the level of significance chosen a priori, is 0.05.

Statistical Significance:

A result is considered significant if the p-value<α.

This is indicative, the authors say, that this effect isn't simply due to random chance.

However, what is statistically significant may not be meaningful. An effect could be statistically significant but so negligibly small that it's irrelevant for all practical purposes. Thus, p-values should be taken in relationship to effect size and domain value.

Calculating p-values in R:

Many statistical test functions you use in R (e.g. t.test(x, y)) will compute the p-value for you.

The test statistic and its corresponding p-value are then presented as output and compared with the significance level.

Common Misinterpretations:

- A low value p-value does not prove that the alternative hypothesis is true.
- A high p-value does not mean the null hypothesis is true, but suggests there is not enough evidence.
- The p-value is not a measure of the size of an effect.

Knowing about p-values - and their dangers - is so important for your own understanding of what the statistics actually says, and also in order to steer clear from making false statements. They should be considered with other statistics like confidence intervals and effect sizes to contribute to a source of evidence.

"Activity: Is the New Marketing Campaign Effective?"

The marketing team at a retail company claims that their new campaign has increased average monthly sales. You are given two sets of data: sales before and after the campaign. Formulate the null and alternative hypotheses, conduct a hypothesis test using R, and interpret the p-value and results. Discuss whether the results are statistically and practically significant. This activity will help you apply the hypothesis testing framework in a real-world business context.

6.6 t-tests

6.6.1 One-Sample t-test in R

One-Sample T-Test We want to compare the mean of one sample to a known or hypothesized population mean. This tests whether the sample mean equals the population mean.

When to Use:

• Evaluating an entire class' average test score against a national reference point

• Calculating the extent to which a typical customer's rating varies from an acme.

Hypotheses:

• Null hypothesis (H_0): the average value of a sample is equal to the population mean ($\mu = \mu_0$)

• Alternative hypothesis (H_1): The mean of the sample is not equal to that of the population ($\mu \neq \mu_0$)

R Implementation:

The t.test() function is used:

sample_data <- c(72, 68, 75, 70, 74, 77, 69) t.test(sample_data,mu = number)

• mu is the known population mean.

• It returns t-statistic, degrees of freedom, p-value and confidence interval.

Key Considerations:

• Sample would be randomly chosen and nominally normally distributed.

• Outliers can influence the computation; visual inspection (e.g., histograms or boxplots) may be useful in order to investigate the distribution.

• If the p-value is less than the selected significance level (alpha=0.05), we reject \(H_0\), which means that the sample mean is significantly different from 30.

Such as quality control, business benchmarking, product testing etc., where one standard (target) value is to be given the preference.

6.6.2 Independent Samples t-test

Independent samples t-test or two-sample t-test This statistical tool is used when comparing means of two unrelated or independent groups. This test answers the question of whether the difference in means observed may be attributed to random fluctuation.

When to Use:

• To assess test scores of two schools in children

• Comparing customer satisfaction in two areas • Measuring consumer satisfaction for 2 regions

• Measuring the market difference between two campaigns

Hypotheses:

• Null hypothesis (H_0): The means of 2 groups are equal ($\mu_1 = \mu_2$)

• Alternative hypothesis (H_1): Means of the two groups are not equal ($\mu_1 \neq \mu_2$)

Assumptions:

• The two samples are independent

• Each group is normally distributed

• Either equal or unequal variances of two groups (tested by variance tests).

R Implementation:

group1 <- c(85, 90, 88, 92, 91)

group2 =c(78,82,80,79,81)

t.test(group1, group2, var. equal = TRUE)

• var. equal = TRUE It assumes that variances are equal (pooled t-test).

•If variances are not equal, then set var. equal = FALSE (Welch's t-test).

Interpreting Output:

• The function outputs the calculated t-statistic, degrees of freedom, p value and confidence interval.

•If the p-value is less than 0.05 (assuming a 5% significance level), then we reject the null hypothesis that there's no difference between populations, i.e. there is a significant difference between group means.

It is THE canonical test in experimental design, A/B testing in marketing, and comparisons of treatments or groups in clinical trials. In business analytics, it helps in contrasting measures of outcomes between distinct groups of customers or compare results over time.

In the case of non normal distributed data, such as in small sample sizes, one could use nonparametric alternatives to the z-test or t-test such'+ '+as the Mann-Whitney U test and Wilcoxon signed rank test ([lr]#Dragulescu 2017)[].

Mann-Whitney U test can be applied.

6.6.3 Paired Samples t-test

Pair-samples t-test A pair-sample test (or the dependent –t-test) is used if we have collected two sets of observations from the same subjects or matched pairs. It is commonly used in 'before and after' studies or experiments where the effect of a treatment or intervention is being studied.

When to Use:

• Weight measurement prior to and after diet in the same group

• The student test scores are compared with the performance of before and after training.

• Comparing sales performance of a new strategy before and after introducing the new strategy.

Hypotheses:

• Null hypothesis (H_0): There is no difference in the mean of compared observations ($\mu d = 0$)

• Alternative hypothesis (H_1): The difference in means is not equal to zero ($\mu d \neq 0$)

Assumptions:

• Differences between measurements on the same pairs are distributed reasonably normally (relatively small sample sizes assumed)

• Pairs are nested and from shared or matched clusters

R Implementation:

```
before <- c(200, 210, 190, 205, 198)

after <- c(220, 215, 195, 210, 202)

t.test(before, after, paired = TRUE)
```

• Specifying paired = TRUE is a paired t-test.

• The test is for differences between pairs, not raw values.

Interpretation:

• The output gives the t-statistic, df and p.

• A small p-value (less than level of significance) results in rejection of the null hypothesis and suggests that there is a significant difference between the paired conditions.

Paired t-tests are highly efficient because of control for between-person differences, hence maximising the efficacy of the test[10].

sensitivity to detect changes.

Did You Know?

"The paired samples t-test is often more powerful than the independent samples t-test because it eliminates between-subject variability, focusing solely on the effect of the intervention or condition being studied."

This type of t-test is extensively used in clinical trials, user experience testing, educational assessments, and pre-post evaluations in business analytics. When the normality assumption is violated, the Wilcoxon signed-rank test serves as a non-parametric alternative.

6.7 Correlation Analysis

6.7.1 Pearson Correlation in R

The most widely used measure of linear association between two continuous variables is the Pearson correlation coefficient (i.e., Pearson's r). It measures the extent to which a change in one variable is closely linked with a linear change in another.

Pearson correlation is quantified using the following formula:

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

Where:

- x_i, y_i are data points
- $\bar{x}, \bar{y}$ are the means of variables x and y

Assumptions:

• Both variables are continuously distributed and can be considered standard normal either way

• Relationship between variables is linear

• Absence of significant outliers

R Implementation:

```
x <- c(2, 4, 6, 8, 10)

y <- c(1, 3, 5, 7, 9)

cor(x, y, method = "pearson")
```

The default is to use Pearson's method for a correlation coefficient unless you indicate otherwise. In case of more than two variables, a correlation matrix can be calculated:

```
cor(dataframe)
```

The output provides pairwise correlations of variables.

Pearson's correlation has been popular for research and business applications—evaluating relationships between metrics such as sales and advertising spend, temperature and energy consumption, price and demand. But when the assumptions are violated, the estimate of correlation may be potentially misleading.

6.7.2 Spearman Rank Correlation in R

The Spearman rank correlation is a nonparametric measure that summarizes the strength and direction of a monotonic relationship between two variables. It is robust against non-normal distributions among its variables, and is agnostic to linearity in contrasts to Pearson correlation, and applicable for continuous as well as ordinal input data.

Spearman correlation uses rank rather than values of the data. The formula for Spearman's rank correlation coefficient (ρ or rho) is given by: PXY = 1 − [6Σd2/(n(n2 −

1))], where PXY is the coefficient of the ranks,, d is the difference in ranks for each pair of samples, and n represents number of samples.

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

Where:

- d_i is the difference between the ranks of corresponding values
- n is the number of paired observations

When to Use:

• Data is not normally distributed

• The relation is monotonous but no linear

• Presence of ordinal variables

• The outliners and their influence has to be minimized

R Implementation:

x <- c(20, 15, 10, 5, 1)

y <- c(1, 2, 3, 4, 5)

cor(x, y, method = "spearman")

Spearman correlation assign ranks to the numbers and then calculation correlation based on the ranks. This is suitable for datasets in which Pearson correlation assumptions are not applicable.

It is popular in disciplines where ordinal data is common, such as psychology, education and survey research. For instance, Spearman's correlation could be employed to determine the association between years of education (ordinal) and job happiness (interval).

Since it is rank-based, the Spearman correlation is much less sensitive to outliers and skewed distributions and a more general purpose tool for analyzing correlations in real-world data.

6.7.3 Interpreting Correlation Results

To interpret the meaning of the correlation coefficient we need to consider its magnitude and sign. The coefficient (r) can vary between -1 and +1:

• +1: Perfect positive correlation

• -1: Perfect negative correlation

• 0: No linear correlation

Strength of Relationship:

Correlation Coefficient (r) Strength

0.90 to 1.00 / -0.90 to -1.00 Strong

0.70 to 0.89 / -0.70 to -0.89 Strong

0.40 to 0.69 / -0.40 to -0.69 Moderate

-0.10 to -0.39/ Weak 0.10 to 0.39/

0.00 to 0.09 / -0.00 to -0.09 Very low

Direction of Relationship:

• Positive relationship: As one quantity increases, the other also increases.

• Single negative association: As one variable grows, the other declines.

Key Points:

• Correlation does not imply causation. Correlation does not indicate cause.

• Always plot the data with scatterplots to identify non-linear relationships or outlies.

• Avoid false positive associations due to confounding variables or chance effects.

Example:

A 0.85 correlation between hours studied and exam score means that there is a strong positive linear relationship — students who study more also tend to earn higher scores.

“Interpreting correlation in context is important. Focus on the practical not just statistical. A statistically significant relationship of low magnitude may not be meaningful in clinical practice.

6.7.4 Visualizing Correlations (Correlation Matrix, Heatmap)

The ability to see correlation is crucial for any exploratory data analysis. It is useful for recognizing a pattern or relationship between more than two variables. The most popular visualizations are the correlation matrix and the correlation heatmap.

Correlation Matrix:

Correlation matrixIts a square table 2 Displaying correlation coefficients for multiple variables. Especially in case of datasets with many numerical variables.

Correlation Matrix in R:

cor_matrix <- cor(mtcars) round(cor_matrix, 2)

This results in a table, with the value of each cell indicating the correlation between two variables.

Heatmap:

The correlation matrix can be visualized as a color-coded heatmap. The color encodes the strength and direction of association. Normally, varying degrees of negative correlation are shown in red hues (or an inverse), positive correlation is depicted with blue/green hues and weak or no correlation with black and white (or light grey) intensity.

Using R to Plot a Heatmap:

library(corrplot) cor_matrix <- cor(mtcars)

corrplot(cor_matrix, method = "color")

Heatmaps help you to see strong correlations, collinear variables or potentially redundant variables in predictive models.

Additional Visualization Techniques:

• Scatterplot matrix (pairs plot): Provides detailed scatterplots of pairwise comparisons.

• ggcorrplot (ggplot2 ecosystem): A tool to give a high-level picture of potential correlations and dates.

Use Cases:

• Dimension reduction of highly correlated predictors before fitting regression models

• Spotting multicollinearity in machine learning

• Investigating the connection between customer behavior or survey responses and financial indices Visualization of data serves to increase its interpretability, aid in decision-making and also adds clarity to statistical findings.

Knowledge Check 1

Q1. What is the range of the Pearson correlation coefficient?

a. 0 to 1

b. -1 to 1

c. 0 to 100

d. -100 to 100

Q2. Which correlation method uses ranks instead of raw data?

a. Pearson

b. Linear

c. Spearman

d. Regression

Q3. A correlation of -0.85 indicates:

a. Weak relationship

b. No relationship

c. Strong negative

d. Strong positive

Q4. What does a correlation of 0 mean?

a. Strong negative

b. Strong positive

c. No linear relation

d. Perfect fit

Q5. Which plot is most useful for visualizing multiple correlations at once?

a. Boxplot

b. Heatmap

c. Histogram

d. Bar chart

6.8 Summary

An Introduction to Descriptive Statistics Descriptive statistics is concerned with the summary of data (mean, median, mode, standard deviation), how descriptive statistics can be used to generate class intervals and graphs displaying frequency distributions.

ϖ The inferential statistical methods are used to make generalization from the sample data and apply them to a population using estimation and hypothesis testing.

ϖ Central tendency refers to measures that represent the center of a distribution.

ϖ Dispersion measures (range, variance, standard deviation and interquartile range) explain the distribution and variability of data.

ϖ Test of Hypothesis ◊ In hypothesis testing, first we state the null hypothesis and alternate hypothesis.

!Inference and Probability the p-value tells us if an eect is statistically signicant (we get more later on what exactly that means, but a smaller value of p indicates there is more evidence against the null hypothesis).

ϖ Means are compared using t-tests: one-sample t-test (sample/population), independent samples t- test (2 groups) and the paired sample-design t-test (related groups).

ϖ Pearson correlation assesses the strength of the linear association between two continuous variables, assuming normality and linearity.

ϖ Spearman rank correlation tests monotony and works on ordinal, or non-normal data.

ϖ We need to interpret observed Patterns with domain-knowledge and not simply extrapolate their content.

Correlation matrices and heatmaps are an intuitive way of visualising associations between many variables.

ϖ Visual tools and statistics complement one another in research and commercial applications of data-driven insights.

6.9 Key Terms

Descriptive Statistics – Methods of summarizing and presenting a data set.

Inferential Statistics — Techniques employed to predict or infer from sample data about the characteristics of a population.

Mean – The average value of a data set.

Median – The value that is in the middle of a ranked set.

Mode - The value that is most frequent in a data set.

Variance – Mean of the squared deviations from the mean.

Deviation- The square root of variance shows spread.

Hypothesis Testing: A procedure for determining the likelihood that assumptions regarding population parameters are true based on sample data.

Null Hypothesis (H_0) – Assumed position of non-effect, or no difference.

Alternative Hypothesis (H_1) – A rival statement that an effect or difference is apparent.

p-value – The chance of seeing the sample data given that the null hypothesis were true.

Correlation Coefficient – The statistical measure of the association or relationship between two variables (and its direction).

6.10 Descriptive Questions

Describe and provide examples of the difference between descriptive and inferential statistics.

Explain the significance of measures of central tendency in data analysis.

Compare and contrast the mean median and mode? When is each preferred?

What is the difference between range and inter-quartile-range as measures of dispersion?

Describe the process of hypothesis testing with an example.

What is the meaning of p value in hypothesis testing?

Distinguish between Pearson and Spearman methods of correlation. Provide usage contexts.

Explain how a correlation heatmap can assist with recognizing relationships in the data.

6.11 References

1. Field, A. (2013). Discovering Statistics Using R. Sage Publications.

2. Moore, D. S., McCabe, G. P., & Craig, B. A. (2017). Introduction to the Practice of Statistics. W.H. Freeman.

3. Utts, J. M., & Heckard, R. F. (2015). Mind on Statistics. Cengage Learning.

4. Crawley, M. J. (2012). The R Book. Wiley.

5. Levine, D. M., Stephan, D. F., & Szabat, K. A. (2019). Statistics for Managers Using Microsoft Excel. Pearson.

6. Dalgaard, P. (2008). Introductory Statistics with R. Springer.

Answers to Knowledge Check

Answer Key to Knowledge Check 1:

1. b – -1 to 1
2. c – Spearman
3. c – Strong negative
4. c – No linear relation
5. b – Heatmap

6.12 Case Study / Practical Exercise

Analyzing Customer Satisfaction and Sales Performance at NovaMart

Background

It is a national retailer with more than 100 stores across various regions. The company has recently introduced a customers competitive strategy and restructured its sales system to improve customer loyalty and increase revenue. Management needs to determine if they meaningfully affected sales and customer satisfaction.

Two datasets were collected:

• You may have to check the Cus tom er training evaluation197 quality of what you are trying to measure before and after a program but even then, these scores are easily subject to confounding factors elsewhere in your work especially if taken over small periods or narrow groups of respondents.

• Monthly Sales Volumes (Listed separately under Region A and Region B)

• Feedback Rating and Purchase Frequency (to calculate correlation)

It is intended that information should be processed in a statistical way, in order to arrive at methods for making planning decisions.

Issue # 1: Is the newly proposed customer satisfaction program efficient? Statistical Method: Paired Samples t-test

The same sample of customers provided both a rating of satisfaction before the programme and after.

Solution in R:

before <- c(6.5, 7.0, 6.8, 7.2, 6.9, 6.6)

after <- c(7.4, 7.6, 7.2, 7.8, 7.5, 7.1)

t.test(before, after, paired = TRUE)

Interpretation:

The p-value is under 0.05, which means that there is a significant increase in satisfaction after program completion. The company can claim this initiative as a success in creating and adding value to the customer experience.

Problem Statement 2: Whether there is difference of sales performance between regions A and B.

Statistical Method: Independent Samples t-test

Two areas are compared, whether their average sales are significantly different.

Solution in R:

region_A <- c(12000, 12500, 11900, 13000, 12800)

region_B <- c(11200, 11400, 11000, 11300,11500)

t.test(region_A, region_B, var. equal = TRUE)

Interpretation:

If the p-value is less than 0.05, then we can say there is a significant difference between region. Otherwise, any difference found may be a coincidence. According to the result, investment strategy for less develop regions can be scheduled.

Problem3: Are customer feedback ratings related to the purchase frequency?

Statistical Method: Pearson Correlation

Examining whether satisfaction correlates with more frequent purchases.

Solution in R:

feedback<-c(6,7,8,9,6,7,8,9)

frequency <- c(2, 3, 4, 5, 2, 3, 4, 5) cor(feedback,frequency,method="pearson")
Explanation:

A correlation coefficient of approximately 0.85 might represent a large positive association. This demonstrates that higher-rated customers will order more often, thus supporting the assumption of satisfaction -loyalty (Breton-Miller and Miller 2006).

Reflective Questions

How would your results be different if the sample sizes were bigger, more varied?

Which assumptions are required for t-test and how will you verify them?

If feedback and purchase are weakly correlated then what might be the possible reasons?

How would Spearman correlation have affected the understanding in non-linear data?

What other information may strengthen the conclusions?

Conclusion

This example helps to demonstrate how statistical methods such as t-test and correlation analysis can help inform real-world business decision making. The paired t-test verified that the new customer satisfaction program had been effective; the independent t-test was used to understand more about different regional performance differences and the correlation test correlated sentiment from customers to purchase behavior. Taken together, these analyses advocate for data-driven decision-making and highlight the primacy of statistical analysis in contemporary business.

BAR Unit 7 V3 (1).docx

Business analytics using R_MBA_2

Business analytics using R_MBA_2

ATLAS SkillTech University

Document Details

Submission ID

trn:oid:::3618:127348111

Submission Date

Feb 2, 2026, 11:05 AM GMT+5:30

Download Date

Feb 2, 2026, 11:07 AM GMT+5:30

File Name

BAR Unit 7 V3 (1).docx

File Size

80.4 KB

18 Pages

4,228 Words

25,325 Characters

0% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text
- Small Matches (less than 30 words)

Match Groups

0 Not Cited or Quoted 0%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

0% Internet sources
0% Publications
0% Submitted works (Student Papers)

Integrity Flags

1 Integrity Flag for Review

Hidden Text
166 suspect characters on 5 pages
Text is altered to blend into the white background of the document.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **0** Not Cited or Quoted 0%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

0% Internet sources
0% Publications
0% Submitted works (Student Papers)

Unit 7: Regression Analysis in R

Learning Outcomes:

1. Explain the concept and assumptions of simple linear regression and apply it to model relationships between two continuous variables.

2. Interpret key components of a regression output such as coefficients, R-squared, p-values, and residuals to draw meaningful conclusions.

3. Differentiate between simple and multiple linear regression models and identify appropriate contexts for their use.

4. Build and evaluate multiple regression models by selecting relevant predictors and checking for multicollinearity, interaction effects, and overall model fit.

5. Use R to estimate regression models, interpret outputs, and validate assumptions through diagnostic plots and residual analysis.

6. Apply regression analysis to real-world problems using business, economic, or social datasets to support data-driven decision-making.

7. Critically assess the limitations and ethical considerations associated with the use of regression models in applied research contexts.

Content:

"Forecasting Sales at Alpha Home Appliances"

For the last year, sales figures have gone up and down for Alpha Home Appliances, a mid-sized consumer electronics company. The marketing group attributes the variation to promotions and seasons, while finance claims it's all about total market demand and price sensitivity. To standardize decision making between departments, the company has announced that it will shift to a data-driven procedure for forecasting so it can better identify fluctuations affecting monthly sales.

The business analyst is Arjun and he must know what factors are affecting the sales, for which he needs to build a predictive model. First, he gathers historical monthly data on the sellers' sales, advertising fee, product price points, customer ratings and seasonal indicators. His initial model applies multiple linear regression to investigate the relationship between (sales and advertising spend). The output reflects a positive relationship between advertising and sales. That bares out: The R-squared is low, indicating that advertising can't explain a lot of the variance by itself.

Arjun] Moving to expanded form of the model, Arjun moves from simple linear calculations to multiple regression algorithms by adding other factors, like price or customer ratings in InMIT positive months. This alternative model performs better in terms of explanatory power and predictive ability. But Arjun defers and observes that some of the predictors are highly correlated, causing multicollinearity which impacts the reliability of those coefficient estimates.

To tweak the model, he examines VIFs, looks at residual plots and forces the regression assumptions to be satisfied. Through iterations and validation, he arrives at a powerful model that can accurately forecast future sales, along the way delivering actionable insights into pricing and promotional stratagems.

The case allows us to learn not only how to conduct a regression analysis but also think critically about the interpretation of results, testing assumptions and enhancing model fit.

Critical Thinking Question:

How can the use of regression models extend beyond merely forecasting to inform strategic business decisions, and what are the dangers that come with relying too heavily on those models without validating assumptions?

7.1 Simple Linear Regression

7.1.1 Concept and Applications of Linear Regression

Simple linear regression is a method for how to model the relationship between one independent variable (or "predictor variable") and a dependent variable.

(Y) using a straight line. The aim is to locate the line of best fit such that it offers the smallest difference of observed pointsw.r.tthe points predicted by the line. The simple linear regression model has as follows general formula:

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

Where:

• Y = the dependent (response) variable

• X is the independent variable (predictor) opportunity-costadenoma JText: 116830 importance Textpagepdffig2 Figure 2 Shown are the agedependent cumulative numbers of adenomas relative to an area of 100 intestinal crypts in ApcMin mice treated with LGG and control (both groups) without administration of azoxymethane before sacrifice.

• β_0 is the point of intercept (Y value when X = 0)

• β_1 represents the slope of the line (i.e., how much Y increases for a 1-unit increase in X)

• ε is the error (residuals) term

The slope coefficient β_1 tells us the direction and strength of the linear relationship between X and Y. A positive β_1 indicates that as X increases, Y also tends to increase and a negative β_1 means the opposite.

Applications of Simple Linear Regression:

Business Forecasting: Baseline foresight prediction of future sales and marketing cost or pricing.

Economics: Model of the income-expenditure relationship.

Marketing: Prediction the effect of service quality on customer satisfaction.

Medicine: Using dosage or length of treatment to predict a time until a patient recovers.

Engineering: Calculating the performance of systems from input variables (temperature, voltage or other quantities).

Key Features of Linear Regression:

• It makes the linear relationship between the independent and dependent variable.

• The model attempts to minimize SSE (the sum of the squared errors).

• It is human-explainable and serves as a building block for more complicated regression models.

Simple linear regression is important not just for the purpose of prediction but also for drawing causal inference, testing theories and practice evidence-based decision making.

7.1.2 Fitting a Simple Linear Regression Model in R

To estimate a simple linear regression model using the software R, we make use of the lm function to fit the regres sion line from Section 6.

It is also easy to do simple linear regression in R. The intrinsic function for and simplest way of performing the simple linear regression analysis in R is lm() which means “linear model”. It enables you to model the dependent variable as a function of one or more predictor variables by using ordinary least squares (OLS) estimation.

How to Perform a Simple Linear Regression in R:

Prepare the data:

Make certain your data meet the following two conditions: the dependent variable (Y) is continuous, and the independent variable (X) is also continuous.

Example dataset

x b test_draw_line(): draw(objects.

geom_smooth(method = "lm", se = T, color = "red") + labs(title = "The Regression Line with Confidence Interval")

- geom_point() adds the points for the real data.
- geom_smooth(method = "lm") plots the regression line and the confidence band.

7.1.3 Assumptions of Linear Regression

Confidence and Prediction Intervals:

Confidence intervals for regression plots provide a measure of uncertainty around the estimate of the mean response. In contrast, prediction intervals give the range within which a new observation is expected to fall.

These intervals can be represented in ggplot2 by drawing shaded area bands around the regression line.

Residual Plot (to check for linearity and spread):

And though the regression line gives an overall sense of the trend, it still may be needed to verify whether the line makes sense by drawing residuals. Investigate as follows # plot fitted versus residuals, this is used to assess model assumptions plot(fitted(model), resid(model), main = "Residual Plot") commun.

abline(h = 0, col = "red")

• A random scatter implies that the linear model is correct.

• A pattern (e.g., a funnel form) indicates trouble like heteroscedasticity or non-linearity.

Multiple Group Regression Lines:

Advanced visualizations in the same way can be used to plot regression lines for various categories on a dataset to compare them.

ggplot(data, aes(x=x, y=y, color=group)) + geom_point() + theme_minimal() output looks like.coordinates Question - 2 How to Create Quadrant Panel in R? host a dataset representing some random points that can be on different positions.

geom_smooth(method = "lm", se = FALSE)

By visualizing regression lines you can instantly see the strength, trajectory and characteristics of the relationship. It also facilitates clear communication of results to non-statisticians engaging with the work.

Did You Know?

"Violating the assumptions of linear regression does not always invalidate the model entirely, but it can significantly distort the accuracy of conclusions drawn from it. Diagnostic checks are not optional—they are an integral part of responsible data analysis."

Assumptions should be routinely checked using visual and statistical methods, and corrective steps should be taken when needed. This strengthens the credibility of the model and ensures that the regression analysis leads to valid, actionable insights.

7.2 Interpreting Regression Results

7.2.1 Coefficients and Their Meaning

The model equation would be expressed as in a simple linear regression:

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

Where:

• Y is the dependent variable

• X is the independent variable

• β_0 is the intercept

• β_1 is the slope coefficient

• ε is the error term

When we fit a regression model, we end up with sample-based values for β_0 and β_1. These are known as the

regression coefficients. Interpretation of Coefficients:

Intercept (β_0) Is the predicted score of the dependent variable when the independent variable is 0. Although it doesn't always have elicitable meaning (especially when X = 0 lies outside the practical range), you still need to know it for this formula as well.

Slope Coefficient (β_1): This is interpreted as the amount we predict that Y changes for a 1-unit change in X, holding all other variables constant (in multiple regression). A positive β_1 indicates a directly proportional relationship, and a negative β_1, an inversely proportional relationship.

Example:

Suppose the regression equation is:

Sales = 200 + 15 Advertising (a) If there is no advertising, Sales = 215(b) For every k=1 to 10, Advertising =k j(a+4)=(c+3k d)s The answer indicates that there has to be at least 43 orders.

This means:

• The intercept (200) is the estimated baseline sales if zero money is spent on advertising.

• Because the coefficient of 15 is positive, you could say —for each extra unit dollars spent on ads, it was expected increase in units sold.

Coefficient Units:

Each coefficient maintains the units of its corresponding variables. Therefore one has to take care in the interpretation because of units. For example, if X is measured in thousands of dollars the interpretation should be compatible with that magnitude.

Signs and Magnitude:

- Sign: Signifies the direction (positive/negative correlation).

• Strength: The value to which the relationship will change for every unit increase.

Multiple Regression:

In multiple regression, coefficients are partial effects. That is, the change in Y caused by a one-unit change in

X 1 while controlling for other variables.

Proper understanding of coefficients tells us how each predictor affects the dependent variable and thus which they have to pay most attention to achieve results in predictive or explanatory use-cases.

7.2.2 R-squared and Adjusted R-squared

R-squared and Adjusted R-squared are two of the most commonly quoted statistics that evaluate how well you have fitted your regression model. They serve to indicate how much variance in the dependent variable is accounted for by the model.

R-squared (R^2):

R^2 is the percentage of the variance in the dependent variable that is predictable from the independent variable(s). It is calculated as:

$R^2 = 1 - (SSR / SST)$

Where:

• SSR stands for sum of squared residuals (errors)

• SST is the total sum of squares (total variability in Y)

R^2 ranges between 0 and 1:

• 0: No variability is explained by the model

• 1 = the model accounts for all of variability

A greater R^2 value means the model accounts for more of the variance in the outcome variable.

Example:

If $R^2 = 0.85$, this implies that the model explains 85% of the variance in the dependent variable.

Limitations of R-squared:

• R^2 always increases with an increase in the number of predictors in a model, even if the new variables do not contribute to it.

• In the case of multiple regression, R-squared may lead to spurious sense of better fit.

Adjusted R-squared:

Adjusted R^2 takes into account the number of predictors used in the model and therefore gives a more realistic value when using multiple regression.

Adjusted $R^2 = 1 - [(1 - R^2) \times (n - 1)/(n - k - 1)]$

Where:

• n = number of observations

• k = number of predictors

Adjusted R^2 will increase only if the new predictor statistically restores more of the model (and thus true predictive power) than would be expected by chance. The deviance is a better estimator of additivity than R2 statistic if different numbers of predictors are used.

Model Selection:

Adjusted R^2 is commonly used for model selection. We want to choose the model having larger adjusted R^2.

Know that R^2 and adjusted R^2 are important measures of fit, but should be used in conjunction with other tool such as residual analysis and out-of-sample prediction performance.

7.2.3 p-values and Statistical Significance

p- values are a cornerstone of statistical analysis. In regressions, they are used to demonstrate if a certain coefficient is statistically significant from zero (i.e., whether the respective independent variable really has an effect on the dependent variable).

Hypothesis Testing for Coefficients:

We perform a hypothesis test for each coefficient β_i:

• Null Hypothesis (H_0): $\beta_i = 0$ (there is no effect)

• Alternate Hypothesis (H_1): $\beta_i \neq 0$ (there is a significant effect)

If the p-value of a coefficient is less than some level of significance (commonly chosen to be $\alpha = 0.05$), we reject the null hypothesis.

How p-values Are Calculated:

We calculate a p-value for each of these two t-statistics, as follows:

$t = \beta_i / SE(\beta_i)$

Where:

• β_i is the estimated coefficient

• $SE(\beta_i)$ is its standard error

This t-value is then compared to a critical value of the t-distribution, which gives the p-value.

Interpreting p-values:

• p 0.05: little or no evidence against H_0

If the VIF for a predictor ≈ 1, that implies it does not contribute substantially to the model or is uncorrelated with other predictors.

Considerations:

• Statistical significance should not be confused with practical significance. A variable can be statistically significant by have a small effect size.

• the p-value is highly influenced by the sample size. Big samples can give small p-values for just about anything.

• The inclusion of more than one dependent variable might introduce multicollinearity, and inflate standard errors (resulting in larger p-values).

It s usually not enough to report a pvalue, as always you should rely on coefficients, R^2 and prior knowledge of the field to draw good conclusions. Overrelying on statistical significance without knowing the context of a threshold leads to bad decisions.

"Activity: What Does the Model Say?"

You are given the regression output from a study investigating the impact of training hours and job experience on employee performance scores. Using the provided summary, interpret the coefficients, R^2, adjusted R^2, and p-values. Identify which predictors are statistically significant and assess the practical meaning of each. Discuss whether the model explains enough variability to be useful in real-world decision-making. Reflect on how additional variables or transformations might improve the model's interpretability and performance. This activity encourages hands-on interpretation, critical thinking, and communication of regression findings.

7.3 Building and Interpreting Multiple Regression Models

7.3.1 Concept of Multiple Regression

The standard for of a multiple linear regression model is:

$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k + \ldots + \varepsilon$

Where:

• The response/dependent variable Y

• X_1, X_2, ..., X_k are p redictors variables (independents)

• β_0 is the intercept

• β_1, β_2, ..., β_k are the regression coefficients

• ε is the error term

Each β_i is interpreted as additional change in Y if X_i increases one-unit while all other variables are left constant. This characteristic of the model provides the ability to estimate for each individual predictor, independently from others, the partial effect of that variable on y; an important advantage with respect to bivariate methods.

Key Uses of Multiple Regression:

• Matching outcome to input from a variety of factors

• Assessing how different factors influence a target measure

• Adjustment for confounding in an observational study

• Identifying the most influential predictors

Types of Multiple Regression:

• Simultaneous multiple regression: All the predictors are considered at the same time.

Stepwise regression- Once certain criteria are met the predictors can automatically be added or removed.

• Hierarchical regression -Predictors are entered as blocks according to their theoretical importance.

Assumptions of multiple linear regression include linearity, independence of errors, homoscedasticity and no multicollinearity or normally distributed residuals. Heterogeneous information may interfere with these assumptions and affect the reliability of the model.

7.3.2 Fitting Multiple Regression Models in R

Regression models in R Fill out the rest of the info below and make t his section your output!

Fitting multiple regression models is easy and flexible using the lm() function in R. It is very similar to simple regression, but applies to more than just one predictor.

Basic Syntax:

model 5: Moderate multicollinearity

- VIF > 10: Cause for serious concern about multicollinearity

Calculating VIF in R:

library(car) vif(model)

Dealing with Multicollinearity:

- Drop, or consolidate, highly correlated ones
- PCA or PLS (see Multivariate Analyses below)
- Center or standardize predictors
- Use regularization based regression methods like Ridge or Lasso

Control for multicollinearity is important so that the coefficients remain stable and to improve the robustness of regression results.

7.3.4 Model Evaluation and Comparison

Assessment of the performance of a regression model is important for assessing the utility and for comparison with other models. There are many criteria for model evaluation.

Key Evaluation Metrics:

R^2 and Adjusted R^2:

o R^2 is the proportion of variance that is accounted for.

o Adjusted R^2 accounts for the number of predictors are preferred when comparing multiple models with different complexities.

F-statistic:

o Model Test the significance of modeloverall.

Residual Standard Error (RSE):

o Represents the average difference between the actual values and predicted values.

Mean Squared Error (MSE) and Root Mean Squared Error (RMSE):

o Evaluate model accuracy. Lower values indicate better fit.

AIC and BIC (Akaike and Bayesian Information Criterion):

o Penalize model complexity. Smaller values are indicative of a good fitting model with respect to parsimony.

Cross-Validation:

• K-Fold Cross-Validation is commonly applied for testing the generalizability.

• The data is divided into k parts and train/tested on all rotations for the splits.

Comparing Models in R:

anova(model1, model2)

This is a test for nested models and compares using an F-test.

A good model should strike the balance between fit and simplicity, and be able to perform well on new data. Over-fitting is absolutely not acceptable, particularly if there is a large number of predictors in the model.

7.3.5 Practical Applications of Multiple Regression

The multiple regression has wide use in various field since it allows complex relationship to be modeled. And it's prediction and explanation at once.

Business and Marketing:

• Predicting sales given advertising, pricing and seasonality

• Understanding drivers of customer satisfaction

• Quantifying the return on digital marketing tactics

Economics and Finance:

• Projecting GDP growth from inflation, investment and trade readings

• The study of stock returns based on financial ratio

Healthcare:

• The patient recovery time estimation: based on the age, treatment and comorbidity of the patient.

• Predictors of disease risk from genetic, lifestyle and demographic factors

Education:

• Attendance, previously scores and social-economics as predictors of student performance

• Assessing the effectiveness of training methods

Public Policy and Social Research:

• Evaluating the impact of policy change upon rates of employment

• Modeling public beliefs from media exposure and demographics

Often times in applied settings, there is usually some knowledge about the domain and that may be included as our beliefs of certain independent variables to select them. It can further inform decisions by measuring the size of each factor and what's driving it.

Playing with numbers: factors influencing the validity of multiple regression. In practice, one should always take account of variable importance, quality issues in the data, and ethical concerns around deployment into production.

Knowledge Check 1

Choose The Correct Option:

Q1. What does a high VIF value indicate?

a. Strong fit

b. Low error

c. Multicollinearity

d. Missing data

Q2. Adjusted R^2 is preferred over R^2 when:

a. Using one predictor

b. Comparing models

c. No intercept

d. Using residuals

Q3. In multiple regression, each coefficient shows:

a. Total effect

b. Direct impact

c. Partial effect

d. Indirect cause

Q4. What does the lm() function in R return?

a. Confusion matrix

b. Model summary

c. P-values only

d. Heatmap

7.4 Summary

ϖ Simple linear regression describes a relationship between one independent variable and one dependent variable in which the relationship is best captured by a straight line.

ϖThe equation of the regression line has an intercept and a slope, estimated by sample data.

¬ Regression model is fitted in R by using the function (lm()) and interpretation of model outputs are made by summary statistics.

ϖ Coefficients from a regression model have meanings in terms of direction and strength.

ϖ Adjusted R-squared is the same as R-squared, but adjusts for the number of predictors.

ϖ p-values are used to decide statistical significance of coefficients, to assess whether predictors have substantive effects.

ϖ Multiple regression can operate on more than one independent variable to perform more sophisticated and realistic modeling.

ϖ Multicollinearity is another commonly encountered problem in multiple regression and its presence can be diagnosed using the VIF.

Model assessment through R-squared, adjusted R-squared, Residual analysis AIC/BIC and Cross-Validation methods it depend.

ϖ Visualization of regression lines, residuals and correlation patterns improve interpretability and model validation.

ϖ Multiple regression is widely used in business, health care, policy, education, and marketing.

b) Properly interpreting regression output involves statistical literacy, understanding the context and checking diagnosis.

7.5 Key Terms

Simple Linear Regression – A way to model the relationship between two continuous variables.

Intercept (β_0) – the predicted value of the dependent variable when all predictors are zero.

Slope (β_1) – The change in one dependent variable unit for a one-unit increase in the independent variable.

R-squared (R^2) – Amount of variance in the dependent variable that is explained by the model.

Adjusted R-squared – An adjusted value of R^2 that takes into account the number of predictors in the model.

p-value – The chance of seeing the data if the null hypothesis is valid.

Multiple Regression – A regression model with more than one predictor variable.

Multicollinearity – A situation in which independent variables are themselves highly correlated.

Variance Inflation Factor (VIF) – A statistic for detecting multicollinearity.

Residual – Observed value minus the predicted value.

F-Stat – A test statistic for judging the overall significance of a regression model.

Prediction Interval – An interval within which a future observation is expected to be found.

7.6 Descriptive Questions

Explain simple linear regression and discuss its assumptions.

Explain the interpretation of the coefficients of a regression equation.

Distinguish between R-squared and adjusted R-squared? Explain them using practical examples.

What is the significance of p-values in regression analysis?

Explain the process you would use to create a multiple regression model in R.

What is multicollinearity? How is it detected and treated?

How do we measure and compare model accuracy with statistical metrics?

Explain how a business might use multiple regression to make a forecast?

7.7 References

1. Kutner, M. H., Nachtsheim, C. J., & Neter, J. (2004). Applied Linear Regression Models. McGraw-Hill Education.

2. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning. Springer.

3. Montgomery, D. C., Peck, E. A., & Vining, G. G. (2012). Introduction to Linear Regression Analysis. Wiley.

4. Field, A. (2013). Discovering Statistics Using R. Sage Publications.

5. Dalgaard, P. (2008). Introductory Statistics with R. Springer.

6. Faraway, J. J. (2005). Linear Models with R. Chapman and Hall/CRC.

Answers to Knowledge Check

Answer Key to Knowledge Check:

1. c – Multicollinearity
2. b – Comparing models
3. c – Partial effect
4. b – Model summary

7.8 Case Study / Practical Exercise

Modeling Employee Productivity at DataEdge Solutions

Background

DataEdge Solutions is a mid-sized analytics company with 100+ employees who are data analysts, engineers and product developers. The leadership team want to know what the main factors are that make it so employees produce (in per month, on average).

In order to optimize HR tactics, they opt for a multiple regression model to predict productivity and considering the below variables:

- H ours T rained (number of hours of formal training in each quarter)
- Experience_Years (total professional experience)
- Remote_Work (binary variable; 1= remote, 0= on-site)
- Team_Size(Number of personnel in the team)

• Education_Level (ordinal: 1 = 'Below CollegeClass', 2 = 'College Class/Undergrad', 3 = 'MasterClass/Grad) The company has gathered information from all of their workers to analyze the available data.

Problem Statement 1: Develop and Interpret a Multiple Regression Model Aim: Our aim of this case study is to develop a regression model that could be used to predict 'productivity' based on various inputs and then interpret the results. Solution:

model <- lm(Productivity ~ Hours_Trained + Experience_Years + Remote_Work + Team_Size + Education_Level, data = employee_data)

summary(model)

Interpretation:

• Hours_Trained: Positive and significant ($p < 0.01$) indicating that training increases productivity.

• Experience_Years: Positive and weakly significant ($p \approx 0.06$).

• Remote_Work: Negative coefficient, which means on-site employees are a small bit more productive in all.

• Team_Size: – Negative and significant; large teams lead to lower individual productivity.

• Education_Level: Not-significant The higher level degree is not definitely influencing on deliverables.

Problem Statement 2: Multicollinearity Objective: We want to check the presence of multicollinearity in the model. Solution:

library(car) vif(model) Interpretation:

• All VIF values are less than 3, indicating no substantial multicollinearity.

• The predictors are independently contributing to the model and the coefficient estimates are not being driven by outliers or atypical observations.

Problem Type 3: Fit an Alternate Model and Predict New Data Goal: Check fit of a model and use the model predictively. Solution:

Model performance summary(model)$r.squared summary(model)$adj. r.squared

Predicting for a new hire.

new_data <- data. frame(Hours_Trained = 20, Experience_Years = 5, Remote_Work = 1, Team_Size = 4, Education_Level =2)

predict(model, newdata = new_data, interval = 'prediction')

Interpretation:

• R-squared = 0.72, the model explains and accounts for 72% of the variation in productivity.

• Adjusted R-squared = 0.70, which means that the model performs well and predictors are meaningful.

• The resultant prediction interval offers a practical reference range of expected productivity for the new employee, thereby providing guidance in reality as to what may be expected.

Reflective Questions

What was the most significant factor affecting productivity and explain?

If that is a change level of education achieved matters would it remain in Model 6? Any other considerations that I may not be thinking of in this decision?

What impact could remote work have on productivity among professions or types of teams?

Would the model be improved by the addition of interaction terms (e.g. Hours_Trained x Experience_Years)? Why or why not?

What other qualitative factors could also impact productivity but are not included in this model?

Conclusion

This study illustrates how a multiple regression model is developed, interpreted and validated in an actual organizational situation. Results indicated that training and experience contributed positively to productivity, with team size and work mode contributing negatively. Education level was by itself not significant (and it never enters one of the models), but there could be other reasons to include certain variables that are not purely empirical – such as theory and what is important for the organization. Finally, multiple regression becomes not only a predictive machine but also an underpinning for strategic HR and operational decision-making. Appropriate interpretation, diagnosis and systematic implementation of the model provide an opportunity for evidence based policy and resource distribution.

BAR Unit 8 V3.docx

Business analytics using R_MBA_2

Business analytics using R_MBA_2

ATLAS SkillTech University

Document Details

Submission ID

trn:oid:::3618:127196242

Submission Date

Jan 30, 2026, 4:01 PM GMT+5:30

Download Date

Feb 2, 2026, 10:27 AM GMT+5:30

File Name

BAR Unit 8 V3.docx

File Size

306.0 KB

28 Pages

6,700 Words

40,465 Characters

6% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text
- Small Matches (less than 10 words)

Match Groups

30 Not Cited or Quoted 6%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

4% Internet sources
3% Publications
4% Submitted works (Student Papers)

Integrity Flags

1 Integrity Flag for Review

Hidden Text
176 suspect characters on 6 pages
Text is altered to blend into the white background of the document.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **30** Not Cited or Quoted 6%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

4% Internet sources
3% Publications
4% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Submitted works	Infile on 2025-04-19 <1%
2	Internet	www.inforly.io <1%
3	Internet	medium.com <1%
4	Internet	www.coursehero.com <1%
5	Submitted works	Sheffield Hallam University on 2024-01-15 <1%
6	Internet	amsdottorato.unibo.it <1%
7	Publication	Hitesh Vasudev, Chander Prakash, Mohammad Khalid, Kavindra Kesari. "Advance... <1%
8	Publication	Pushpa Choudhary, Sambit Satpathy, Arvind Dagur, Dhirendra Kumar Shukla. "Re... <1%
9	Publication	Zhao Zi-an, Feng Xiu-fang, Ren Xiao-qiang, Dong Yun-yun. "Uncertainty-guided cr... <1%
10	Publication	Bright, Scott. "Washington State's Correctional Education Programs: Impact on R... <1%

11	Submitted works	
Goldsmiths' College on 2025-03-13		<1%
12	Internet	
publikationen.bibliothek.kit.edu		<1%
13	Submitted works	
Islington College,Nepal on 2026-01-06		<1%
14	Submitted works	
South Dakota Board of Regents on 2023-11-23		<1%
15	Internet	
assets-eu.researchsquare.com		<1%
16	Internet	
www.mdpi.com		<1%
17	Publication	
Thangaprakash Sengodan, Sanjay Misra, M Murugappan. "Advances in Electrical ...		<1%
18	Submitted works	
Ilia State University on 2023-03-27		<1%
19	Publication	
Lukas Glänzer, Lennart Göpfert, Thomas Schmitz-Rode, Ioana Slabu. "Navigating ...		<1%
20	Submitted works	
Sim University on 2006-09-27		<1%
21	Internet	
arxiv.org		<1%
22	Internet	
pure.rug.nl		<1%
23	Publication	
Santos, Maria Carolina Bentes Pimenta Paisana. "Predicting the Movement of Peo...		<1%
24	Submitted works	
Technological University Dublin on 2023-07-24		<1%

25 Submitted works

University at Buffalo on 2024-08-09 <1%

26 Internet

ebin.pub <1%

27 Internet

littleflowercollege.edu.in <1%

28 Internet

ntnuopen.ntnu.no <1%

Unit 8: Introduction to Machine Learning in R (Part I)

Learning Outcomes:

1. Explain the fundamental concepts of machine learning, including its purpose, types, and real-world relevance across industries.

2. Differentiate between supervised and unsupervised learning, with examples illustrating key applications and algorithm types under each category.

3. Describe the principles of classification and regression models, highlighting their roles in predictive analytics and decision-making.

16 4. Apply regression-based machine learning algorithms such as Decision Trees and K-Nearest Neighbors (KNN) using R for predictive modeling tasks.

27 5. Evaluate and interpret the outputs of machine learning models using appropriate performance metrics for both classification and regression.

6. Recognize the limitations and challenges of machine learning models, including overfitting, underfitting, and bias in training data.

2 7. Demonstrate the ability to select suitable machine learning methods based on the nature of the data and the problem at hand.

Content:

8.0 Introductory Caselet

8.1 Overview of Machine Learning

8.2 Supervised vs Unsupervised Learning

8.3 Introduction to Classification and Regression Models

8.4 Regression Analysis in R (Decision Tree, KNN)

20 8.5 Summary

8.6 Key Terms

8.7 Descriptive Questions

8.8 References

8.9 Case Study

8.0 Introductory Caselet

“Predicting Customer Churn at OptiNet Telecom.”

A regional ISP called OptiNet Telecom had been struggling with growing customer churn for the last two quarters. Customer satisfaction is down almost 10% even with regular customer surveys and to combat more defection, new data plans were announced. The marketing and customer service arms have worked in silos to try and drive satisfaction improvements, but their efforts are disconnected with no advance predictive ability.

So the C-Level management decides that they will make data driven decisions and turns to the analytics team to solve this. The company wants to be able to forecast which customers are at highest risk for cessation and can then target these customers for intervention prior to churn.

With the historical background information of customer demographics, plan aspects, service usage, complaint history and churn phenomena at hand, the analytics team starts to investigate if machine learning can be put in this business application. They suggest constructing predictive models that can detect the patterns of historical churn behavior.

The team initially breaks down the problem as a supervised learning problem, as you already know the dependent variable (churn or not). They test different classification models and begin with decision trees that can give easily interpretable rules for predicting churn. Additionally, they try the K-Nearest Neighbors (KNN) method in order to classify new customers by comparing them to similar previous cases. With the help of R, they prepare the data, divide it into a training and test set, and assess model performance with the accuracy, precision, recall.

As outcomes emerge, the analytics team not only creates a working churn prediction model, but it also discovers which variables — such as data usage decreases and call complaints frequency — are stronger indicators of dissatisfaction. This research is then passed to marketing and customer service, who use it to plan tailored retention campaigns.

Critical Thinking Question:

How does OptiNet ensure that their machine learning models continue to be accurate in the long term, when patterns in customer behavior and market conditions change?

8.1 Overview of Machine Learning

8.1.1 Definition and Importance of Machine Learning

improve performance from experience without being explicitly programmed. It leans on building algorithms that make it possible for computers to recognize patterns, take decisions or predict outcomes based on input data. While conventional rule-based systems need explicit rules that are manually programmed, machine learning models rely on data and can learn to improve automatically as additional training data are obtained.

At its core, machine learning consists of feeding a lot of data into an algorithm that learns the relationships between variables. Once a model is trained, it can make predictions or (in the case of classification) classifications on noel data. The objective is usually as to which function will be optimized (minimizing prediction error, or maximizing accuracy) .

It's no exaggeration to say machine learning has revolutionized our contemporary, data-laden world. It is a key driver of automation, efficiency and innovation in multiple fields. Companies use machine learning to personalize customer experiences, streamline supply chains and detect fraud. In science, machine learning helps researchers tackle discoveries in areas including genomics, climate modeling and particle physics.

Broadly speaking, there are three categories of machine learning:

7 • "Supervised learning," in which a model learns from labeled data.

• Unsupervised learning, which finds patterns in unlabeled data.

• Reinforcement learning, in which agents learn through their interactions with the environment.

So why is machine learning so important?

• Scaling: Models scale up to huge datasets larger than what humans can't process.

• Flexibility: Over time, ML systems get better with experience—dynamic learning becomes possible.

• Predictive Power: ML models can be used to predict future results based on past behavior.

In general, machine learning helps converting data into actionable insights and is indispensable in modern problem solving in academia and industry.

8.1.2 Applications of Machine Learning in Business & Research

Machine learning has been applied to various domains in business and academia, 18 making it one of the most important technological innovations of the 21st century. Its

power comes from being able to sort through enormous amounts of data, making patterns and trends that would be impossible for humans to identify on their own.

In business, machine learning is used in the following aspects:

Marketing and Customer Insights:

o Customer segmentation using clustering algorithms

o Predictive churn and LTV analytics

o Collaborative Filtering Recommendations

Finance and Risk Management:

o Credit scoring and risk evaluation models

o Anomaly detection based fraud discovery

o Algorithmic trading with a prediction model

Operations and Supply Chain:

o Regression versus time series based demand forecasting

o Inventory optimization through predictive analytics

o Reinforcement learning for route planning and logistics optimization

Human Resources:

o Resume screening and candidate ranking

o Predicting employee attrition

o Workforce planning and performance analytics

Machine learning is also revolutionizing academic and scientific research:

Healthcare and Life Sciences:

o Predictive and diagnostic disease modeling using various classification models

o Genomic Data analytics and drug discovery

o Personalized medicine through predictive modeling

Environmental Science:

o Neural network based Climate pattern forecasting

o Land use classification using satellite image analysis

o Monitoring biodiversity and species migration

Social Science and Education:

o Social media data captured during the summit – analysis of texts

o Predictive modeling of student performance

o Data Mining of Survey data for Behavior analysis

Machine learning is not simply a mechanism for automation but an empowerment for deeper insight and innovation. In business, it improves decision-making by mitigating uncertainty. In science, it enables hypothesis tests as well as exploration and discovery at scales previously impossible. It is a naturally literal language that can handle both structured data and unstructured text, which makes it useful across many domains.

8.1.3 ML Workflow: Data, Model Evaluation, Deployment

The machine learning workflow: A sequence of stages that defines how to implement and maintain a machine learning model. It is important to understand this process and push the development of (not only accurate but) reliable, scalable models that can be deployed in production environments.

Figure 8.1

The major components of the ML workflow are:

Data Collection and Preparation:

o Collect information from valid sources to be able to support it with facts and statistics
o Make certain that the material is relevant.

o Data Cleaning, and Data Preprocessing (missing value treatment, outlier analysis and normalization).

o Partition the dataset into training, validation, and test sets.

Feature Engineering:

o Pick key variables (features) that impact the model's performance.

o Generate new features based on domain knowledge or transformations.

o Encode categorical features and Scale the numerical condition if required.

Model Selection and Training:

o Select a suitable model (ex: decision tree, logistic regression, KNN).

14 o Fine tune the model on the training data.

o Tune model parameters using cross-validation.

Model Evaluation:

o Test the model with accuracy, precision, recall, F1-score or RMSE/R^2.

o Evaluate confusion matrices and ROC curves of classification problems.

o Assess generalizability using test data.

Model Optimization:

o Tune hyperparameters using grid search or random search.

o Deal with overfitting or underfitting by regularization, pruning, or feature selection.

Deployment:

o Deploy the trained model to production systems or other applications.

o Monitor how well first and trained models remain adequate in their predictive ability over-time and be willing to retrain.

o Deploy at scale using APIs or containers.

Model Maintenance:

o Monitor model performance as new data becomes available.

o Retrain the model or refresh the training data if accuracy degrades.

Each process depends on and is driven by the other. For example, something you learn from model evaluation may make you go back to feature selection or even re-gather data.

The ML workflow underscores that building a model is only part of the work. Good planning, rigorous testing and deployment infrastructure are equally important for success with machine learning applications.

Did You Know?

"Over 80% of the time spent in real-world machine learning projects is dedicated to data collection and preparation, not model building. Clean, well-structured data is often the single biggest factor influencing model performance."

The ML workflow emphasizes that model building is just one part of the process. Proper planning, rigorous evaluation, and deployment infrastructure are equally vital for achieving successful machine learning applications.

8.1.4 ML in R: Available Packages and Libraries

R, the language most popular for statistical computing and data analysis, is accompanied by a strong library of packages that can be used in implementing different machine learning methods. These packages incorporate various algorithms with data preprocessing, visualization and model evaluation tools available in R as a platform for academic work, but also for applied machine learning studies.

Some of the most used software packages in R for machine learning are:

caret (Classification and Regression Training):

o Single interface for training over 200 models.

o It automatically processes data, selects features and does cross-validation and hyperparameter tuning.

o Tools such as train() ease model building across a wide range of algorithms.

randomForest:

o Uses the Random Forest algorithm for classification and regression.

o Works well in high dimensions and is resistant to over-fitting.

o Good for calculating importance of variable and ensemble modeling.

rpart:

o Acronym for Recursive Partitioning and Regression Trees.

o Grows decision tree models and provides rule visualizations in user-friendly formats.

o Works well for easily understandable models in business scenarios.

e1071:

o Supports Supports Vector Machine (SVM), Naïve Bayes, and K-means interfaces.

Integrates classification, regression and clustering instruments. o Awake not to mention loaded with life when ever you might need all of us five.

o Well-known for flexibility in kernel-based learning techniques.

xgboost:

o An optimized, scalable gradient-boosting library.

o Used commonly in ML challenges and high performing modeling.

o Tends to overfit as we need manually finetune the hyperparameters, but achieve reasonably high accuracy.

nnet and neuralnet:

o For implementing artificial neural networks.

o Handle both regression and classification.

o Handy for small to medium sized deep learning models.

mlr and mlr3:

o Uniform model creation, tuning, and benchmarking frameworks.

o Present structured flows for ML pipelines.

o mlr3 is the newer, more modular version, which has been optimized and updated with regard to extensibility.

tidymodels:

o A set of modern packages organised according to tidyverse principles.

o Presents a grammar for modeling that is consistent with data manipulation process.

o Contains recipes, parsnip, yardstick and tune.

R also interoperates with visualization packages (e.g. ggplot2, lattice) to facilitate model diagnostics, monitoring,m and interpreting results.

These libraries transform R into an all-purpose suite for machine learning—data processing, modeling, deployment and evaluation. For those analysts and researchers that like their work to be transparent, statistically grounded and interpretable R provides a natural and rigorous home for developing machine learning solutions.

8.2 Supervised vs Unsupervised Learning

8.2.1 Supervised Learning – Concept and Examples

Supervision in learning is when the model is trained on a dataset with labels, also known as supervised learning. Here, "labeled" means that each training example has a labels or target variable with it.

6 The objective is to learn a mapping function f(X) = Y, where X denotes input variable(s), and Y represents output (or target). After training, the model can be applied to new
4 input data.

Supervised learning In supervised learning, we are given a data set and already know what our output should look like, having the idea of the relationship between it and the
12 input. Standard loss functions are e.g. the Mean Squared Error (MSE) for regression and Cross-Entropy Loss for classification.

There are two broad categories of tasks in supervised learning:

Class: The outcome variable is a factor. The model is trained to classify input into one of a number of categories.

o Example: Email spam or Not spam decisions (spam/not spam)

o Example: CREDIT RISK EVALUATION H,M,L (High, Medium or Low Risk)

o Algorithms: Logit, DT,RF,SVM and NN

Regression: The target is a continuous variable. The numerical output is the predicted value of the model given input values.

o Example: Prediction of which product a customer will buy, based on his past behavior.

oEx : Predicting sales revenue based on advertising spend

o Algorithms: Linear regression, Decision Trees for regression, K-Nearest Neighbors (KNN), Gradient Boosting

Key characteristics of supervised learning:

• Needs to be fed with labeled data, the acquisition of which may be expensive and time-consuming.

• Performance can be quantified using measures such as accuracy, precision, recall, R^2 and other metrics.

• Interpretable in structured domain as finance and healthcare etc.

Supervised learning is often useful when we have a historical data set that includes known responses or classes and there are certain predictors that are good for prediction/classification purpose.

13 8.2.2 Unsupervised Learning – Concept and Examples

Unsupervised learning is a type of machine learning in which the model learns from input data that has no corresponding output labels. The aim is to find out any data pattern, grouping or structure.

within the data. Rather than making a prediction of a target variable, the algorithm attempts to learn the inherent structure of the data.

In unsupervised learning, the algorithm learns only from the characteristics in data, without any supervision or guidance. These methods are mostly employed for data analysis, compression as well as pattern recognition.

25 Two well-known types of unsupervised learning tasks are:

Clustering: The data points are clustered according to their attribute.

o Example: Segmenting customer based on their purchase behavior

o Example: Grouping of documents by similar topic ♣ Listing 1.31 M: Other Clustering Applications • Image Segmentation – grouping of the pixels of an image based on similarity (color, texture) • Organize broadcast to reflect demographic groups to be reached o Sentiment analysis/clue extraction (instead of topic identification): finding out if there is something bad happening here!?.

o Classifiers: K-Means, DBSCAN, Hierarchical Clustering, Gaussian Mixture Models

Dimensionality Reduction: Reduces the dimension of input variables and/or features, typically for visualization or to improve performance.

o Example: PCA for summarizing high-dimensional gene expression data

o For example – t-SNE for depict/visualize customer preferences in 2-D boxplot, scatter plot etc.)

o Algorithms: PCA, t-SNE, Autoencoders 3 Important aspects of unsupervised learning:

• Does not need labeled data, so it is suitable for large untagged datasets.

• Used for pattern detection, not to make predictions.

• Performance evaluations are more based on subjective manner, including the decoding quality (Domain experience) or clustering metrics like silhouette score6.

Unsupervised learning is currently among the best performing strategies in marketing analytics, fraud detection, genomics and natural language processing for uncovering structure in complex unlabeled datasets.

8.2.3 Key Differences Between the Two Approaches

There is a fundamental difference between supervised and unsupervised learning in the way they learn from data, the types of tasks for which they are applicable; and their performance measures.

Presence of Labels:

o In case of supervised learning there is labeled data, so for each training example it can be determined that) given that input.

o Unsupervised Learning * Unsupervised Learning: This type of the learning deals with data free from any output label.

Goal of Learning:

o The goal for supervised learning is to predict a target (a classification or regression).

o Unsupervised learning aims at finding underlying patterns or structure in the data.

Type of Output:

o Supervised: It answers discrete (classification) and continuous (regression) output values.

o Unsupervised: The output is Clustering or Feature transformation.

Evaluation Metrics:

o Models built with supervised learning are evaluated metrics like (and not limited to) accuracy, precision, recall or F1-score (classification) or RMSE, MAE and R^2 for the case of regression.

o There is no direct evaluation measure in unsupervised learning, and clustering scores (e.g., silhouette coefficient), visualization or domain relevance are used to judge the performance.

Examples:

o Supervised: Spam detection, predicting stock prices, medical diagnosis.

o Unsupervised: Segmentation of market, topic discovery and anomaly detection.

Complexity and Interpretability:

o Supervised models are easier to interpret and predict how well it would perform.

o Models that are not supervised tend to be more «exploratory» and should simply be used as suggestion and they can at times be harder to interpret, due to the heavy dependency on a knowledge of the given domain.

Data Requirements:

o Labeling requirement makes supervised learning data hungry.

o Unsupervised is a good idea to consider when in initial stages of exploring vast amounts of unlabeled data.

These differences could be applied to select the right learning strategy depending on the description of problem statement, available data and analysis goals.

8.2.4 When to Use Supervised vs Unsupervised Learning

Which between the supervised and unsupervised learning to apply, depends on various factors such as the nature of data set, availability (or absence) of labeled outcomes and what one wants to achieve with analysis.

Use Supervised Learning When:

• Textbook source includes the labeled outcomes (targets) of the data.

• Prediction or classi□cation is the main purpose.

• You need to determine the impact of various variables on an outcome.

• Examples: customer churn classification, loan default classification, product demand forecasting.

Use Unsupervised Learning When:

• There are no labeled output and the objective is to search for pattern or structure.

• You want to cluster data points, squash data together or find outliers.

• Examples: Customer segmentation, fraud detection and feature reduction.

Practical Considerations:

• Cost of Labeling: Unlabeled data may be available more cheaply or more easily than labeled data; in this case, we would ideally choose an unsupervised algorithm.

• Maturity of domain –for proven mature domains with plentiful labeled data, supervised methods can produce very accurate models.

• Unsupervised Learning: Unsupervised learning is helpful in exploratory data analysis where we are trying to discover some hidden groupings or trends.

Hybrid Scenarios:

• Both methods are, in some cases, combined. One such application is clustering which detects partitions of the data used to construct meaningfully targeted supervised models.

9 • Semi-supervised learning, an intermediate approach between the two, relies on a small amount of labeled data and a large pool of unlabeled data to direct learning.

In the final analysis, your decision between supervised and unsupervised learning will normal depend on the statement of the problem, nature of data as well as the final goals. A good feature selection enhances the model's performance and better suits to business or research purposes.

"Activity: Label or Discover?"

You are a data analyst at a retail firm and have been given access to customer transaction records. However, only 20% of the dataset includes labels indicating whether a customer responded to a previous campaign. Your team wants to build a system to understand customer behavior and predict responses to future campaigns. Discuss which learning approach—supervised, unsupervised, or a combination—would be most appropriate and justify your decision. Outline your strategy for modeling, data preparation, and performance evaluation, considering the partial labeling and business objectives.

8.3 Introduction to Classification and Regression Models

8.3.1 Classification Models – Overview

Classifiers are a type of supervised learning technique that involves training machines to predict discrete classes. The goal of classification is to assign input instances into predefined classes or categories, based on their attributes. These models are learned from datasets in which input features and their respective class labels have been associated. When it is trained, one can use the model to assign unseen data (from that same domain) into one of these categories.

The classification problem is binary (two classes) or multiclass if there are more than two classes. For instance, classifying a transaction as a fraud or non-fraud is a binary classification problem, and classifying types of fruits based on image features is a multiclass problem.

Commonly used classification algorithms include:

17

• Logistic Regression: It is a classification, not a regression. It uses a logistic (sigmoid) function to model the probability of the default class and is well suited for binary classification.

• Decision Trees : Data is divided into branches of the tree based on feature values. Leaves in tree stand for class labels. They are intuitive, easy to understand and don't require scaling of features.

• Random Forest: A collection of decision trees that increases the accuracy of predictions and compensates the overfitting. It sums up the predictions of several trees to achieve final classification.

• Support Vector Machines (SVM): they build a hyperplane which optimally separates data of the N-dimensional feature space into different classes. SVMs perform well in high-dimensional spaces and with the help of kernel tricks, it can model non-linear boundaries.

• K-Nearest Neighbors (KNN): A non-parametric approach to classify a data point according to the majority label of its nearest neighbors. It is straightforward and plausible, but computationally expensive for large crops.

Classification models have the following main features:

•Training and Prediction: A model is trained on labeled data where the relationships between features and class labels are learned. Predictions are obtained by plugging the learned function into new data.

• Probabilistic outputs: A number of models generate probabilities that a sample belongs to a certain class and can be used to adjust the threshold depending on business specific demands or risk.

• Decision Boundaries: Classification algorithms define either explicitly or implicitly decision boundaries in the feature space which separate one class from another.

The classification models are utilized in a wide variety of areas such as disease diagnosis, spamfiling fraud detection, VF image recognition sentiment analysis and credit scoring assessment. They perform very well when labeled data is provided and we have a need to make decisions according to what we are seeing or observing in the data.

8.3.2 Regression Models – Overview

It is a concept of regression models where we predict a continuous value as an output with the help of even more than one feature. The basic idea is to derive a mathematical relationship between the independent variables (predictors) and the dependent variable (response). In supervised learning, regression attempts to measure the effect that input variables have on a quantitative response variable.

The least complicated form of a regression technique is termed as linear regression, which can be expressed as:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_k X_k + \varepsilon$$

Where:

- Y is the predicted output
- β_0 is the intercept
- β_i are coefficients for predictor X_i
- ε is the error term

Linear regression assumes that the relationship between the DV and IV is linear. It computes the least-squares solution to optimally fit a line through the data.

Other popular regression models are the following:

- Polynomial Regression: Similar to linear regression but subset regression with extra polynomial terms X^2 and X^3 that allow the model to fit a straight line without being restricted.
- Decision Tree Regression: Segments the data to fit each feature threshold. And every terminal node predicts a unique constant.
- Random Forest Regression: An ensemble of multiple decision trees which averages their predictions. It addresses the non-linear relationships and works well for outliers.
- K-Nearest Neighbors Regression: It makes a prediction for a new input by averaging the outputs of its K nearest training data points.
- Gradient Boosting Regression: It constructs models in a stepwise manner such that the current model used is to predict residuals from the previous one. famous implementations include XGBoost and LightGBM.

Basic elements of the regression model are as follows:

- Model Assumptions: Linear regression makes several assumptions, including linearity, independence of errors, homoscedasticity (constant variance of errors) and residuals thatare normally distributed.
- Model Fit: Evaluated through error estimators like RMSE (Root Mean Squared Error), MAE (Mean Absolute Error) and R^2 (coefficient of determination).
- Interpretation: Linear regression receives high interpretability as it easy to understand the relations of variable, while tree-based methods receive low interpretation because it offers us with scores of importance.

Fluency in Regression It's also used commonly in professional applications- from sales forecasts, price estimates and demand projection to risk modeling. It is useful for both prediction and inferential purposes in the sense we can make future predictions but also understand which factors are driving outcomes.

8.3.3 Model Selection Criteria

Having the right model that reflects the task at hand is crucial to obtaining reliable, interpretable and efficient results with machine learning. Whether in classification or regression, model selection is based on a multitude of factors that describe the data type and characteristics, the computational setting, and corporate/ research aims.

Type of Target Variable:

• If the outcome is qualitative, we should use a classification model.

• If outcome is continuous, you can use regression models.

Interpretability:

• Simple models that are easy to interpret — such as linear regression or logistic regression.

• More sophisticated models, such as random forests or neural networks, have higher accuracy but less transparency.

Data Size and Dimensionality:

• On small datasets complex models can perform worse and overfit less than simpler ones.

• In high-dimensional data, it is often desirable to use algorithms that have regularization (e.g.Edgington,H.ystack the Ridge or Lasso) or dimension reduction (PCA).

Feature Relationships:

• Models in which f(x) is linear admit an additive, linear approximation.

• When the patterns are non-linear ones, tree-based models or kernel-based models hold better accuracy.

Treatment of Missing Values and Outliers:

• Some algorithms, such as decision trees, are able to handle missing values either natively or by imputing them.

• Robust models aren't influenced by the outliers but in this case linear regression is very much affected.

Computational Cost:

• Simpler models are computationally less expensive to evaluate and train.

• The more complex models may have better accuracy but they also need considerable time and resources.

Model Stability:

• Ensemble models generally are more robust and not so sensitive to slight variation in data.

• The variability in models such as single decision trees can influence reproducibility.

Business Requirements:

• If you need to explain decisions, use off-the-shelf models.

• If having correct predictions is most important to you (with understanding being secondary), choose a model that emphasizes performance.

Evaluation Results:

• Compare models using a consistent set of performance metrics through cross-validation.

• Think about overfitting by testing how well models generalize to new data.

Ultimately, model selection is iterative. Analysts generally take a series of baseline models, iterate through them and finally enhance those models (or depreciate as they see fit or if directed to find more advanced models and replace the older ones. This trade-off between the needs for accurate predictions, interpretability, limited resources, and feasible deployments.

8.3.4 Evaluation Metrics (Accuracy, RMSE, etc.)

When it comes to machine learning the devil is in the detail that's picking the right score to measure classifier performance. The selection of metric depends on the problem's nature (i.e., classification or regression) and the kind of dataset. Metrics inform model selection, tuning, and validation by measuring the quality of predictions with respect to actual outcomes.

For Classification Models:

8 Accuracy:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Computes the ratio of the number of true positives to the sum of false negatives and true positives. Ideal to play when sides are even.

Precision:

Precision = TP ÷ (TP + FP)

Tells you how many positives were claimed to be good and actually are.

Sensitivity (Recall): Recall = TP ÷ (TP + FN)

Evaluate how well the model is able to find all true positives.

22 F1 Score:

F1 = 2× (Precision × Recall) ÷ (Precision + Recall) A well balanced metric between precision and recall.

Confusion Matrix:

2 A table which shows true positives, false positives, true negatives and false negatives. It provides a comprehensive perspective of the model's performance.

ROC-AUC Score:

Evaluate how good the model is to distinguish between classes. AUC demonstrates higher AUC reflects better performance.

For Regression Models:

Mean Absolute Error (MAE): MAE = $\sum |y_i - \hat{y}_i| \div n$

This ignores the sign of errors.

Mean Squared Error (MSE) MSE = $\Sigma (y_i - \hat{y}_i)^2 \div n$

And punishes over-estimation more than under.

Root Mean Squared Error (RMSE):

RMSE = $\sqrt{MSE}$

Returns error in the same units as the target.

R-squared (R^2):

$R^2 = 1 - (SSR \div SST)$

24 Shows the percentage of variation explained by our model. Ranges from 0 to 1.

Adjusted R-squared:

The R^2adj.ajustado adjusts for number of predictors and enables a fair comparison between models with different fits.

Metrics are chosen according to what's relevant. Precision and recall are more practical than accuracy for imbalanced classification. For models in which large errors appear to be relatively more detrimental than smaller errors, MAE does not perform well.

Did You Know?

"In many business applications, a model with slightly lower accuracy but higher recall may be preferred, especially when the cost of missing a positive case is high, such as in fraud detection or disease diagnosis."

Choosing the right evaluation metric ensures that the model is not only statistically strong but also aligned with practical goals and real-world consequences.

8.4 Regression Analysis in R (Decision Tree, KNN)

8.4.1 Decision Tree Regression – Concept and Implementation in R

Decision Tree Regression Formulation of a decision tree regression: A tree method applies at each decision node t, for that sample i in partition R_f(t) where f is the region at node t.

Modification of the Decision Tree Regression Being one of non-parametric, tree-based approaches, Decision Tree Regression can be used to forecast continuous response by recursively partitioning intra-groups variance within feature space according to categorical splitting rule. The objective is to make predictions at the leaf/terminal nodes by averaging the target values within that region. At each node, the algorithm splitting point and feature that reduces variance (or some equivalent loss) as much as possible within each subset of features begin creating more pure subsets.

In R theformation of such a model is generally done by adopting rpart package here in which formula interface is used with desired outcome and predictor(s) while rpart() function is called. For example:

```
library(rpart)

model_tree <- rpart(y ~ x1 + x2 + x3, data = train_data, method = "anova")
```

Here it's by method=anova, which tells us that the outcome is continuous. Once the model has been trained, you can Ã‚Â evaluate it with printcp(model_tree) to see the complexity parameter table, or visuÃ‚Â alize it by typing plot(model_tree); text(model_tree).

Key aspects to consider:

• Splitting rule: Most often by minimising sum of squared errors (variance) in regression.

• Tree complexity: Trees have a tendency to grow very deep and overfit the training data; hence pruning or maximum depth constraints and minimum observations per node helps in generalization.

• Interpretability : Decision trees are very interpretable as the predictions follow simple branching rules.

• No feature scaling required: Conglomeration trees are insensitive to monotone transformation, and do not suffer from different scales of variables.

Decision trees are popular in business and science too, due to their intuitive nature and low data preprocessing demands. However, it may become fragile to noise in the data and overfit unless controlled.

8.4.2 K-Nearest Neighbors (KNN) Regression – Concept and Implementation in R

K-Nearest Neighbors Regression is a locally weighted learning technique, in the sense that the prediction at each point is based only on neighbor training points. For a new point xxx, its prediction is:

$\hat{y} = (1/K) \times \Sigma\ y_i$, iii among the K closest neighbors

Optionally, weights decreasing with the distance can also be used to give more importance to closer neighbors:

$\hat{y} = \Sigma (w_i \times y_i) / \Sigma w_i$, where $w_i = 1 / \text{dist}(x, x_i)$

1 In R, KNN regression can be done using the caret package or packages such as FNN: library(FNN)

predictions <- knn. reg (train = train_data[,predictors], test = test_data[,predictors],

y = train_data$y, k = 5)$pred

Important considerations:

• Distance metric: Popular choice is Euclidean distance, but scaling in case of different ranges of features is crucial.

• K : Small K can be too noisy, large K can smooth point but hide local structure. Cross-validation can be used to decide on the best K.

• Computational complexity: KNN needs to store the entire training set and compute distances at prediction time, so it is less efficient for large datasets.

• No real model training as such: I.e., the algorithm is “lazy”—it basically postpones all computation until prediction.

KNN regression is easy to do and useful for exploratory modeling, but it requires careful feature normalization and sometimes attention to computational efficiency.

8.4.3 Comparing Decision Tree and KNN Regression Models

Decision Tree Regression and KNN regression are both non-parametric, instance-based techniques Their structures, computational approach, how they're used in practice are completely different.

Model Structuring:

• Decision trees make a hierarchical division of the features space and so produce easy to understand decision rules.

• KNN relies solely on distance in feature space to make predictions, with no explicit structure imposed on the model.

Training vs Prediction:

• Training a decision tree is slow but prediction is fast because of the traversing through the tree.

• KNN Training cost is small but prediction costly in terms of computing all the distances to all training points.

Handling Feature Scales:

• Trees are invariant to scale.

• KNN needs to have features normalized because otherwise the feature with a higher range will weight more.

Interpretability:

• They are interpretable and the ‘why’ is shown.

• KNN provides no interpretable rules- it just gives you average of neighbors.

Hyperparameters:

• Tree performance was manipulated by depth, minimum samples per leaf and pruning complexity.

• Performance of KNN depends on value of K and the method to weight distances.

Sensitivity:

• Decision trees can overfit noisy data, but they can be pruned and regularized.

• KNN is able to capture local structure, but it is too sensitive to noise and irrelevant information so information selection or weighting are needed.

The choice between them depends on the data and application: if you want interpretable models with shaped cuts, then would recommend trees; if you want a form of local approximation where no training is needed try KNN.

8.4.4 Strengths and Limitations of Each Approach

It is important to know the pros and cons of Decision Tree Regression and KNN regression when using them in real practice.

Decision Tree Regression:

Strengths:

High interpretability—rules are easy to visualize.

• Easily handles mixed data types and missing values.

• No scaling or complicated preprocessing is required.

Limitations:

• Have the tendency to be overfitting, especially with deep trees – they need pruning or pre-pruning _CONVEX.

• Unreproducible—little data, minor changes in data can produce varying tree structure.

KNN Regression: Strengths:

• Very ease to understanding and execute.

•General—no constraints on distribution of data or model shape.

• If features are on similar scale, quite good for smooth non-linear functions.

Limitations:

• Expensive to compute at prediction time in case of large datasets.

• Data setting with high dimensional data (curse of dimensionality).

• Depends on careful scaling and selection of features to prevent irrelevant features from dominating.

A well informed application will select the model most appropriate for the data, interpretability requirements, and computational limitations.

Knowledge Check 1

Choose The Correct Options :

1. Decision Tree Regression splits data based on minimizing which measure?

a. RMSE

b. Variance

c. Distance

d. Correlation

2. In KNN regression, predictions are made by averaging what?

a. All data points

b. Cluster centers

c. K nearest neighbors

d. Tree leaves

3. Which method is insensitive to feature scaling?

a. KNN

b. Decision Tree

c. SVM

d. Linear Regression

4. KNN regression’s prediction cost is high due to what?

a. Training time

b. Tree depth

c. Matrix inversion

d. Distance computation

5. Which model offers clearer interpretability?

a. KNN

b. Neural Net

c. Decision Tree

d. SVM

8.5 Summary

ϖ Machine learning is an approach in which patterns are sought for systems to learn from data rather than being explicitly programmed to do so.

ϖ Labeled data is used for supervised learning to train models for classification or regression, while in unsupervised learning we work with unlabeled data to uncover patterns.

ϖ Typically, classification models predict discrete outcomes and these types of algorithms are logistic regression, decision trees and KNN (among others).

ϖ Regression models are used when the output is continuous, such as linear regression, decision trees for regression and KNN Regression.

19 ϖ Decision tree regression constructs a model by recursively partitioning the dataset into groups that minimize the variance and enable easy interpretation of decision paths.

ϖ KNN regression predicts based on the average of K neighboring data points, thus it needs feature normalization for performance effectiveness.

ϖ The choice for model selection depends on data type, interpretability requirements, computational resources and prediction goals.

21 ϖ Metrics such as accuracy, precision, recall and F1-score are used in classification instead of MAE, RMSE and R2.

ϖ In R, decision tree could be conducted using rpart package and KNN regression are usually performed via FNN or caret package.

ϖ Decision tree is highly interpretable and irrelevant feature resistant, however KNN is flexible but computationally expensive and sensitive to scaling of features.

ϖ Each approach has advantages and is most appropriate when their assumptions are satisfied by the characteristics of the dataset.

Machine learning models need to be assessed with suitable metrics and, tested in a held-out set of data for generalisation and out-of-sample validation to avoid overfitting.

8.6 Key Terms

Supervised Learning: The machine is given labeled data where inputs and expected outputs are provided.

Unsupervised Learning – Learns patterns or clusters from data without predefined labels.

Classification – Supervised learning where data is labeled into pre-specified categories.

Regresssion -A behaviour in which one tries to predict values, as well know as Continuous Values.

Decision Tree – A model that segments data into several subsets using a tree structure depending on their values for prediction.

K-Nearest Neighbors (KNN) – Is a type of algorithm in which provides values by considering the K nearest data points.

28 Accuracy – The ratio of the number of correct predictions to the total observations.

RMSE -Root Mean Squared Error, a measure of the average dualselection error of the regression models.predictions.

Feature Scaling – During preprocessing, a normalization of the range of input variables is made.

Pruning -Procedure of cutting back a decision tree to help prevent overfitting.

Model Generalization – A model that does well on new data that it has not been trained to see before.

Cross-Validation – A method to determine how well a model's validation set results will generalize against other independent samples.

8.7 Descriptive Questions

Describe the main differences between supervised and unsupervised learning, suitable with examples.

How would you build a decision tree regression model in R and what are its benefits w.r.t interpretability?

Explain how KNN regression works and what factors are such that will lead to better performance.

Contrast regression trees and KNN in structure, scalability, and interpretation.

Which are the standard metrics to evaluate a classification/regression model? Explain their significance.

Explain the significance of feature scaling in KNN regression and what impact it has on prediction.

What are some pros and cons of decision tree regression models?

To what extent is the choice of model a function of data and objective?

8.8 References

1. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning. Springer.

2. Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling. Springer.

3. Lantz, B. (2019). Machine Learning with R: Expert Techniques for Predictive Modeling. Packt Publishing.

4. Han, J., Kamber, M., & Pei, J. (2011). Data Mining: Concepts and Techniques. Morgan Kaufmann.

5. Torgo, L. (2016). Data Mining with R: Learning with Case Studies. CRC Press.

6. Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow. O'Reilly

Media.

Answers to Knowledge Check

Correct options for Knowledge check 1 :

1. b. Variance
2. c. K nearest neighbors
3. b. Decision Tree
4. d. Distance computation
5. c. Decision Tree

8.9 Case Study / Practical Exercise

Predicting Housing Prices Using Regression Models in R

Background:

A real estate company needs a predictive model to estimate house prices in a metropolitan area. Their dataset includes the following features: square footage, number of bedrooms, age of the house, distance to the city center, and price. They

decide to implement two regressions and compare them: Decision Tree Regression, and K-Nearest Neighbors Regression. What were the specifics?

Problem Statement 1:

Build Decision Tree Regression for a house price prediction.

Solution: 1. Load the data and explore.

Split the data to train and testing sets.

Use rpart package to fit the model.

Visualize the tree and extract rules.

Evaluate the model using RMSE, and R^2.

5 library(rpart) model_dt <- rpart(price ~., data = train_data, method ="anova") pred_dt <- predict(model_dt, test_data) rmse_dt <- sqrt(mean((test_data$price - pred_dt)^2))

Problem Statement 2:

Implement KNN Regression and compare.

Solution: 1. Work with normalized dataset using standard scaling.

KNN regression using the FNN package.

Find the optimal K using cross-validation.

Predict and evaluate RMSE R^2.

1 library(FNN) knn_pred <- knn.reg(train = train_scaled, test = test_scaled, y = train_data$price, k = 5)$pred rmse_knn <- sqrt(mean((test_data$price - knn_pred)^2))

Problem Statement 3:

Compare and Interpret.

Solution: Find the lowest considering the RMSE, which approach has the better fit. Decide, which one is easier to interpret – a tree when every case could be explained and the KNN while there is a “black-box” in the middle. Compare the speed – which model is the hardest to implement on the big data. For the R^2, define which approach gives more variance account. Finally, choose the right approach. What is your understanding?

Reflective Questions:

Which one has the better result? Why? How does the standard scaling affect the KNN performance? When will you use tree approach in a similar situation? Did you experience the issue with the model tuning? How would the model behave with more features? With more data? Do you agree? Today’s Case Study illustrated two popular

regression models frequently used in R programming. While Decision Trees provided the best understandable and fast approach, the KNN showed the flexibility of use. The correctness of applied methods and models predetermination allowed providing an accurate forecast and actionable analytics tips required for whether to vote for one or another house or real estate.

BAR Unit 9 V3.docx

Business analytics using R_MBA_2

Business analytics using R_MBA_2

ATLAS SkillTech University

Document Details

Submission ID

trn:oid:::3618:127348551

Submission Date

Feb 2, 2026, 11:05 AM GMT+5:30

Download Date

Feb 2, 2026, 11:07 AM GMT+5:30

File Name

BAR Unit 9 V3.docx

File Size

86.1 KB

20 Pages

4,929 Words

30,271 Characters

1% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text
- Small Matches (less than 30 words)

Match Groups

2 Not Cited or Quoted 1%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

1% Internet sources
0% Publications
0% Submitted works (Student Papers)

Integrity Flags

1 Integrity Flag for Review

Hidden Text
222 suspect characters on 6 pages
Text is altered to blend into the white background of the document.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **2** Not Cited or Quoted 1%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

1% Internet sources
0% Publications
0% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1 Internet

medium.com <1%

2 Internet

kth.diva-portal.org <1%

Unit 9: Classification and Clustering in R (Part II)

Learning Outcomes:

1. Explain the foundational concepts of classification and clustering, distinguishing between supervised and unsupervised learning techniques.

2. Implement classification models in R, including decision tree classifiers, to predict categorical outcomes based on labeled data.

3. Interpret the output of classification models, including decision paths, class probabilities, and model accuracy metrics.

4. Describe the concept and applications of clustering, with a focus on unsupervised pattern discovery in unlabeled datasets.

5. Apply K-means clustering in R to group data into meaningful clusters, and evaluate cluster quality using internal validation metrics.

6. Differentiate between classification and clustering approaches, identifying appropriate use cases for each in business and research contexts.

7. Use R libraries and functions to build, visualize, and evaluate classification and clustering models, demonstrating proficiency in practical data science workflows.

Content:

9.0 Introductory Caselet

"Segmenting Success – A Tale of Customer Patterns."

FreshCart, a mid-sized online grocer delivery company, has exploded over the past couple years. With increasing number of customers and thousands of transactions every week, the executive team started having difficulty understanding how their customers were changing over time. Their time-honored tools of analysis -descriptive statistics, trend-based forecasting- had become inadequate to address difficult questions like:

- Who is most likely to be responsive to a sales promotion?
- What are common features across buyer groups?
- How to cluster customers for targeted marketing strategies?

To solve these problems, the analytics team decided to investigate classification and cluster analysis by utilizing a statistical language called R. They first attempted to classify customers who are likely to redeem a promotion coupon from those who do not redeem it with the information of previous buying behavior. This meant I had to create a classification model with the target being binary– coupon redeemed or not.

At the same time, the team sought to identify Natural groupings in customers based on frequency of purchase, product variety, basket size and response time. This got them thinking about unsupervised learning via K-means clustering which they used to categorize their user base without predetermined categories.

The results were striking. Classification approach was used to predict campaign performance, while clustering identified three primary customer segments: loyal patrons, price-conscious shoppers and browsers. These realizations led to the new marketing strategies, custom promotions and stock planning.

With plans to grow FreshCart, the team plans to build and scale models and automate these workflows while integrating analytics into day-to-day operations.

Critical Thinking Question:

How can fresh cart have a competitive edge over other businesses(few mentioned above) by mixing both classification and clustering models? Especially when they want to launch into new market or they want to create a new category of products.

9.1 Classification in R

9.1.1 Concept of Classification in Machine Learning

Machine learning has classification as an essential element of supervised learning, which attempts to predict discrete labels or categories instead of continuous values. What we want to do is to learn a function f(X) = Y where X is your input features and Y is your categorical result. The model is trained on such a labeled dataset. Most of the time, classification problems are binary (two classes) or multi-class (more than two).

In classification, a model is created that separates the features space into regions for various classes. The purpose of the training process is to optimise models to minimize its classification error, where loss functions such as misclassification rate or cross-entropy are employed. The way algorithms do this vary:

• Logistic Regression applies a sigmoid function to transform linear combinations of features into class probabilities, which is particularly suitable for binary classification.

• Decision Trees simply splits the data over and over again on every feature to minimize impurity in sub-samples.

• Random Forests constructs multiple trees to obtain predictions and aggregates them to enhance prediction accuracy, and also avoid overfitting.

• SVM: The SVMs seek separating hyperplanes for classes.

• k-Nearest Neighbors(KNN) uses the majority class of nearest points to assign a class.

Classification is key in many real-world applications: between spam and non-spam emails, diseases, fraud detection tasks, sentiment analysis and images. The performance of classification approaches depends on data quality, feature set selection, class imbalance, and the type of algorithm. Performance of classifiers are estimated using measurements such as accuracy, precision, recall and visualized using tools like confusion matrix or Roc curve. In R, if one wishes to carry out classification, run-time libraries (e.g., caret, rpart, randomForest, e1071 and nnet) can readily facilitate development of training evaluation and deployment workflows.

9.1.2 Model Evaluation – Confusion Matrix, Accuracy, and ROC Curve

Performance of classification models is related with evaluation metrics that provide quantitative measure. Important tools are the Confusion Matrix, Accuracy and the ROC Curve with AUC.

Confusion Matrix

A confusion matrix is a 2x2 table summarizing how often a classifier is right/wrong, and for what topic:

Predicted Positive Predicted Negative Actual Positive TP FN

Actual Negative FP TN

• TP (True Positive): positive correct prediction.

• FN (False Negative): positive, but predicted negative.

• FP (Fake Positive): in fact-negative, but predicted as positive.

• TN (True Negative): negative prediction is the correct one.

This matrix gives detailed view of where the classifier is wrong or strong. It becomes a foundation for some crucial measures:

• Accuracy = (TP + TN) ÷ (TP + TN + FP + FN) Overall, how often is the classifier correct? The ratio of correct predictions to the total number of predictions but, it can be misleading if one class dominates the others.

• Recall = TP ÷ (TP + FN) is the fraction of positives that were successfully identified.

• Remember (Sensitivity) = TP ÷ (TP + FN) represents how well the model captures the true positive cases.

• F_1 Score (the harmonic mean of precision and recall) provides a balance between false positives and false negatives.

ROC Curve and AUC

The ROC (Receiver Operating Characteristic) curve has the True Positive Rate (TPR) on the xlabel and.

False Positive Rate (FPR) with different classification thresholds.

• TPR = TP ÷ (TP + FN)

• FPR = FP ÷ (FP + TN)

The curve demonstrates the compromise between sensitivity and specificity as threshold varies. The classifier discrimination power can be quantified using the Area Under the ROC Curve (AUC). An AUC of 1 indicates perfect classification; an AUC of 0.5 represents no better than random guessing.

ROC analysis is particularly helpful when the classes are balanced or the cost of FP is about equal to that of FN. For extremely imbalanced examples, precision-recall curves may be more informative.

In R we have used this in a model-evaluation context when something like the following applies:

• Generating predictions and actual labels.

• Constructing of a confusion matrix using table() or functions in packages such as caret.

• Measurements: Accuracy, precision, recall and F_1 score.

• Plotting ROC curves with software tools like pROC or ROCR.

• AUC computing to summarize model discriminability.

Good evaluation allows you to tune classification models, as well adjust thresholds between precision and recall; it also offers insights into model performance beyond naive accuracy.

Did You Know?

"A model can achieve high accuracy yet perform poorly if one class dominates; this is known as the accuracy paradox, and tools like ROC curves and F_1 scores provide a more nuanced performance picture."

9.2 Classification using Decision Tree

9.2.1 Decision Tree for Classification – Concept

A classification decision tree can be represented in the form of a binary tree, i.e., a tree in which each node splits into two branches (Safavian and Landgrebe 1991; Quinlan 1993).

A classification decision tree is a tree-like structure that captures decisions and their outcomes: chance event realizations, resources expended and utility gained. It is nonparametric technique of supervised learning, which is used for both regression and classification. In classification, the goal is to assign data items to a specific category based on input variables.

A decision tree is comprised of nodes, branches and leaves:

• Main Node: It is the root node that contains first test split and it represents the whole dataset.

• Internal Nodes: These represent decisions on the inputs.

• Branches: They are used to link nodes and express tests outcomes.

• Leaf Nodes (Terminal Nodes): Display class predictions at the end.

The algorithm is such that it takes the dataset and divides it initially based on whatever feature provides the most efficient separation of classes. Splitting conditions are criteria such as Gini impurity or information gain (entropy based). At a given iteration, the algorithm selects the feature that leads to greatest class purity.

Key concepts include:

• Impurity measures Statistics that measure how impure (or mixed) the classes are in the node. A lower impurity indicates a more homogeneous nodes.

• Split: At each node, for each feature find a threshold that best split the data into classes.

• Pruning: Once the full tree has grown, pruning removes branches that have little power in predicting response to prevent overfitting.

• Overfitting: When the tree fits training data perfectly while generalizing poorly to new data.

Decision trees are popular in areas, where result clarity is important (in health sector, for example), because the rules and resulting visualization can be easily understood even by a layperson. They are particularly valuable when rapid and interpretable decisions are required and the data is a mix of numerical and categorical features.

9.2.2 Building Decision Tree Classifiers in R (rpart, caret)

Creating decision tree classifiers in R is easy and can be done as you wish. You can use the well-known package rpart for simple modeling just as you'd call other basic functions and caret, with which you can perform nearly all types of your modeling workflows in combination including training, tuning, and validation.

With rpart, you would do something like:

library(rpart)

model 10, then Class = A

• Importance of Features: Some are providing ranking that depict contribution of each feature in reducing impurity over all splits. Can tell you which factors have the strongest influence on your predictions.

• Confusion Matrix: Measures how well the model predicted the test data, by comparing its predictions against actual labels (It gives you the number of true positives, false positives, false negatives and true negatives). Derived metrics such as accuracy, precision, recall and F_1 score can also be computed.

• Cross-Validation Results: Since decision tree has high tendency to overfit, cross validation performance is useful for generalization check.

• Pruning effects: Comparison of full vs pruned trees illustrates how simplification impacts predictive power and interpretability. Often, pruned trees generalize better.

As with anything, apply the lens of business: a rule way raise an area that you want to target. Be careful of splits based on noisy or collinear features.

9.2.4 Advantages and Limitations of Decision Trees

• Interpretable results: The findings are transparent, and can be communicated easily.

• Data mix mode: Trees can handle categorical and numerical data without much of prepossessing.

• No scaling: Intrinsically invariant against feature scale.

• Automatic feature selection: Splits will automatically give higher ranks to more informative variables.

Limitations:

• Overfitting: Full deep trees can model noise, cutter them or control the parameters.

• Uncertainty: The structures and splits can vary if the data is perturbed.

• Bias toward high-cardinal variables: A variable with high cardinality can contain the majority of information and lead to biased splits.

• Limited predictive performance: Single-tree models tend to provide lower accuracy than ensemble models such as RF or GBM.

“Activity 1: Predicting Loan Approval with Decision Trees”

You are an analyst at a bank aiming to develop a fair and interpretable model to predict whether loan applications will be approved. Using customer data—including credit score, income, loan amount, and employment status—build a decision tree classifier in R. Train the model, visualize it, and extract decision rules. Then, evaluate its performance using a confusion matrix and key metrics. Present your findings to non-technical stakeholders, emphasizing how the model’s transparency supports fairness and explainability in lending decisions.

9.3 Introduction to Clustering

9.3.1 Concept and Applications of Clustering

Clustering is a fundamental unsupervised machine learning method of grouping similar data points according to some notion of similarity. Clustering is unsupervised, that means it does not require a labeled output. It does not fit the content of data into some predefined pattern, instead it fits to data patterns and new structure is discovered by forming an arbitrary representation of multidimensional space consisting of clusters.

The general concept behind clustering is the calculation of similarity or distance between data points and determining how closely associated they are using measurements like Euclidean distance, Manhattan distance or cosine similarity. Accordingly, clustering algorithms group data points such that intra-cluster similarity is maximized and inter-cluster similarity minimized.

There are numerous applications to clustering:

• Customer Segmentation: In the marketing world, clustering can be adopted to recognize groups of customers with common purchasing behavior, demographics or level of engagement and promote targeted campaigns.

• Image Segmentation: Segmentation of images involves grouping together pixels by color or texture to form objects or regions.

• Document Clustering: Documents can be grouped into topics they are about, without prior knowledge of the topic labels, which is helpful in news aggregation or academic research.

• Anomaly Detection: Points not belonging well to any cluster may be considered as outliers or anomaly, helpful in fraud detection or surveillance.

• Biological Data Analysis: In the domain of genomics, clustering has helped researchers to aggregate together genes with similar expression profiles, thus assisting in disease research.

• If applied to urban planning, Clustering is used to categorize areas with similar density of traffic or usage of resources.

Well-known clustering algorithms include K-means, hierarchical clustering, DBSCAN and Gaussian mixture models. In R, many functions such as kmeans(), hclust() and cutree() are commonly utilized, which make use of visualization tools like cluster plots and dendrograms to increase interpretability.

Clustering is an essential tool in exploratory data analysis. By disclosing obscured structure, it can support supervised modeling tasks or enable business strategy and scientific discovery.

9.3.3 Hierarchical vs Non-Hierarchical Clustering Approaches

There are two principal approaches to the design of cluster models: hierarchical and non-hierarchical.

An overview of classification and clustering Classification Alloy sorting methods Clustering algorithms can be divided into two main types: (1) hierarchical and (2) non-hierarchical or partitioning. Each type of merging adheres to different logic when combining information, and they are suitable for different types of tasks.

Use of Hierarchical Clustering yields a tree based on the clusters formed called as dendrogram. It works without need of pre defining number of clusters. There are two major types:

• Agglomerative (bottom-up): It begins with each observation as a separate cluster and merges them together in the end based on similarity.

• Divisive (top-down): All observations start in one cluster, and splits of clusters are performed recursively.

Different methods of linkage (single, complete and average) will define how the distance between clusters is measured when clustering or unclustering.

Key features of hierarchical clustering:

• No assumption of the number of clusters in advance.

• Description of the structure of data groupings.

• Appropriate for medium size datasets.

• Visualized using a dendrogram.

• Less computationally efficient as a result of pair-wise distance computations.

Non-hierarchical Clustering: K-means In k -means clustering, we partition the data into a fixed number of clusters (k) such that each point belongs to the cluster whose mean it is closest to, minimising within-cluster variation. It is efficient and iterative over large datasets.

K-means clustering works as follows:

Initialize k centroids randomly.

Associate each point with the closest centroid.

OK new centroids based on the current assignment of elements to clusters.

Repeat until there is no more change in assignments.

Key features of non-hierarchical clustering:

• It is efficient and scalable to large datasets.

- Need to know the end number of clusters (k) in prior.
- Sensitive to initial cluster assignment.
- Spheric clusters and clusters all of the same size are assumed.

Comparison Overview:

Attribute Hierarchical Clustering Non-Hierarchical Clustering (e.g., K-means)

Number of clustersPost-hocDetermined in advance

Structure Tree-like (nested clusters) Flat (non-overlapping clusters)

Scalability Not efficient for large data So very scalable

Flexibility Enables different linkage options Distance from centroid is commonly applied

Visualization Dendrogram Cluster plot or silhouette analysis

Whether to use one or the other depends on dataset size, desired interpretability, data type and purpose of analysis. The hierarchical strategy is used because of its depth into the data and better visualization, but we chose K-means due to being faster and easier to implement.

9.3.4 Evaluation of Clustering Results

Assessing clusters goodness plays an essential role in knowing how a clustering method is done. Compared to classification, for which we could easily measure performance against known labels, clustering frequently does not have a ground truth, particularly in unsupervised scenarios. Therefore, the internal validation measures are applied to evaluate

cohesion (how close the points of a cluster are to one another) and separation (how distinct a cluster is from other clusters).

Here are the key metrics and approaches to assess clustering results:

Silhouette Score

The measure compares similarity between each data point and (its own cluster) and other clusters. The silhouette score for a point:

$s(i) = (bi - ai)/max\{ai, bi\}$

Where:

2

- a(i) is the average distance of point i from all other points in the same cluster.
- b(i) is the minimum mean distance of point i from all points in other clusters.

The score is between −1 and +1. A higher value represents better clustering. A value near zero implies two clusters overlap; a negative can mean misallocation.

Calinski–Harabasz Index

Also called the variance ratio criterion, which is calculated by the ratio of between-cluster and within-cluster dispersion. A larger index indicates denser and better separated clusters.

Davies–Bouldin Index

This criterion measures the extent to which clusters are similar within themselves and dissimilar from one another, averaged over all of the cluster pairs. Small values correspond to better clusters.

Dunn Index

This ratio measures the disparity between short distances separating points in different clusters, and longer distance that are internal to each cluster. Higher the Dunn index better the well-formed, compact clusters.

Visual Evaluation

Visualizations, including cluster scatter plots and silhouette plots, assist in interpretation of the quality of clusters. Dendrograms are for hierarchical clustering, PCA-based plots for high-dimensional data.

Did You Know?

"The silhouette score, introduced in 1987, remains one of the most effective ways to simultaneously evaluate the cohesion and separation of clusters, even when the ground truth is unknown."

Evaluating clustering is not just about numbers. It involves comparing different algorithms, varying the number of clusters, and ensuring that the results make sense in the context of the data. It's essential to balance statistical validity with interpretability and practical relevance.

9.4 K-means Clustering

9.4.1 Concept and Algorithm of K-means Clustering

K-means clustering is an unsupervised learning algorithm that partitions a set of n samples into K clusters, where each sample belongs to the group whose mean is nearest it. The purpose of the algorithm is to minimize WCSS which in other words means that all points in a cluster should be similar.

The algorithm is composed of the iterated process:

Initialization: Choose K starting centroids—these can either be randomly selected from the data set, or could be selected using a heuristic (e.g., k-means++).

Assignment Step: Assign each point to the closest centroid by distance (typically Euclidian).

Update Step: Calculate the new centroid for each cluster, using the mean of all points belonging to that cluster.

Convergence Test: Continue to steps 2 and 3 until no significant change in centroids or the maximum number of iteration reached.

From the mathematical point of view, we want to find:

WCSS = $\Sigma_i \Sigma_{x \in C_i}$ distance(x, μ_i)**2

Here C_i denotes cluster i, and μ_i the center of that cluster. K-mean creates convex, Voronoi tesselations and the clusters are likely to be of comparable spatial scale.

Key Characteristics:

• Simple and efficient: Suitable for large dataset with quick convergence.

• Centroid-biased partitioning: All points are members of a single cluster with no soft assignments.

• Initialization sensitivity: Varied initial centers may yield local minima rather than global ones.

• Spherical clusters and isotropic cluster size: Best performance when variances for data clusters do not significantly different.

Clustering is widely used for market segmentation, image quantization, novelty detection and customer profiling. An understanding of the composition of clusters can inform strategic decisions if the clustering has meaning in business terms.

9.4.2 Implementing K-means in R (kmeans() function)

In R, K-means clustering is implemented via the kmeans() function that is relatively simple and flexible. Its basic usage is:

set.seed(123)

result 0 print("running k-means alg. on test data and help func") return (clust1.vector) } Keep in mind that the best thing is to start with a single cluster, then increasing the number of clusters and averages over optimal point in each cluster reached by groups obtained. max = 100)

• data_matrix: Numeric matrix or data frame with the features.

• centers: The number of clusters or the initial centroids.

• nstart: Number of random initializations; higher value reduces effect of suboptimal starts.

• iter. max: The maximum number of iterations per run.

The output (result) includes:

• cluster: node array Indicating which cluster each point belongs to.

• centers: The x,y-Coordinates of the cluster cetroids for the last iteration.

• tot. withinss: Total within sum of squares.

• totss: Total sum of squares.

• betweenss: Between-cluster sum of squares; the sum of squared distance between cluster means and the overall mean.

Example in practice:

Prepare the data: rescale features to have same scale using scale().

Run K-means with various starts: Set nstart = 25 or greater for more stable clustering.

Assess cluster output: Inspect sizes, centroid values, WCSS vs total variance.

Visualise clusters: Reduce dimensionality with PCA and plot on color-coded scatter plots. scaled_data <- scale(data_matrix)

km 1,]$cluster table(km$cluster)

Best practices:

• Scale down the data to prevent domination by variables with higher range of numbers.

• Use multiple runs (nstart) to address initialization issues.

• Check cluster composition for interpretability, not just mathematical correctness.

9.4.3 Choosing the Number of Clusters (Elbow Method, Silhouette Score)

Estimation of the number of clusters (K) can be an important step in K-means. This selection is guided by two commonly used techniques:

Elbow Method:

• Plot WCSS against the values of K.

• Identify the "elbow" point which would be a good value for k.

• The elbow point indicates the optimum K is over this, the enhancement will be tending to stabilize.

Silhouette Score:

• Measures similarity of data point to its own cluster vs nearest other cluster.

• For each point the "silhouette value" s(i) is calculated:

1

a(i) = average distance of i to other points in the same cluster

b(i) = minimum mean distance of points to the points in nearest cluster

s(i) = (b(i) − a(i)) / max{a(i), b(i)}

• The average Silhouette score over all the points, varies between −1 and +1. The larger the values of H k and D k are, the better their cohesion and separation will be.

Both methods have strengths:

• Elbow gives an indication of compactness but it can be subjective.

• Silhouette metric measures the cluster quality but is more computationally expensive.

Their joint use results in much more confidant decisions about K which trade off compactness, interpretability and cluster separation.

9.4.4 Practical Applications of K-means in Business Analytics

K-means is used in many applications, like the following:

• Customer Segmentation: Segment customers based on their buying activity for personalized marketing and targeted offers.

• Product Packaging: Aggregate demographic data points and behaviors to craft marketing personas.

• Retail Inventory Optimization – Cluster the store types based on sales patterns to optimize stocking strategy or promotional scheduling.

• Website Visitor Profiling: Profile visitors based on engagement metrics to customize site experiences or recommend content.

• Cluster social media posts by similarity of content or sentiment to determine trends and customer behaviour on the online sphere.

• Operational Machine Performance: Cluster the similar machine performance or operation parameters together to filter anomalies or requirements for maintenance.

For instance, a retail chain could group stores by daily sales volume, foot traffic and preferences of customers to designate them as “high-volume flagship,” “regional hubs” or “seasonal outlets” and adjust staffing, supply lines or promotional budgets accordingly.

Due to the simplicity, speed, and scalability of k-means clustering it is an exploratory analysis technique often applied at initial stages. Such clusters are usually used to inform other modeling — maybe a classification, forecasting, or regression strategy.

Knowledge Check 1

1. K-means partitions data into clusters by minimizing what?

a. Total distance

b. WCSS

c. SSE

d. Centroid error

2. In R’s kmeans() function, what does nstart control?

a. Iterations

b. Scaling

c. Initialization runs

d. Cluster count

3. The Elbow Method helps determine K by plotting WCSS vs K. Where is the optimal K?

a. Maximum point

b. Minimum point

c. Elbow point

d. Zero point

4. A high silhouette score indicates clusters are:

a. Arbitrary

b. Well-separated

c. Overlapping

d. Undefined

5. K-means clustering is suitable for:

a. Labeled data

b. Unsupervised segmentation

c. Time series

d. Regression only

9.5 Summary

ϖ Clustering is an unsupervised learning technique for grouping similar data points without a priori category information.

ϖ Classification is placing of the data points into pre- defined classes using supervised learning algorithms.

ϖ Hierarchical clustering forms a binary tree of clusters using either the agglomerative or divisive approach.

ϖ Flat clustering algorithms, where the most famous one is K-means, divide data into given numbers of flat clusters.

ϖ Clustering quality is evaluated using evaluation metrics such as silhouette value, Davies– Bouldin index and Calinski–Harabasz index.

ϖ K-means clustering, a process that attempt to minimise the within-cluster sum of squares, can be used on large data.

ϖ The following will generate clustering in R using kmeans() function by first iteratively assigning points to the nearby centroid.

ϖ It's important to determine the appropriate number of clusters based on the elbow method or silhouette value.

ϖ For classification purposes, decision trees can split data based on the most useful features.

ϖ The R packages rpart and caret, which are typically used to create and evaluate the decision tree models.

• Both classification and clustering forms of machine learning form the rough groundset for customer segmentation and pattern recognition.

• Application of these techniques include market segmentation, fraud detection and recommendations systems.

9.6 Key Terms

Clustering: Dividing similar data points without known outcomes.

K-means: A partitioning algorithm that groups data into K clusters by how close they are to the centroids.

Classification: Supervised learning method where a test example is assigned to one of the several predetermined categories.

Centroid: The average of all points in cluster.

Silhouette Score: The silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation).

Elbow Method: A graphical method of finding the correct number of cluster by plotting WCSS.

Hierarchical Clustering: A clustering algorithm that creates a tree-like structure of clusters in which items; men and women, are nested within these clusters.

rpart: A package for the R statistical programming environment which is implemented in C and designed to produce decision trees either from a classification or regression perspective.

Confusion Matrix: A testing performance summary table for models that classify.

ROC Curve: A graphical representation of the diagnostic ability of a binary classifier.

Dendrogram: A branching structure diagram that represents the clusters created through hierarchical clustering.

caret: A package for train and assessing machine learning models in R.

9.7 Descriptive Questions

Differentiate between classification and clustering with examples.

Explain the algorithm for K-means clustirng?

How do you measure the quality of clusters in unsupervised learning?

What are the major Pros and Cons of decision trees for classification?

Compare and contrast between hierarchical and non-hierarchical clustering techniques.

Explain the computation and interpretation of the silhouette score.

Discuss the procedure and steps to perform K-means clustering in R?

Allude to the kinds of problems classification and clustering can solve in business analysis.

9.8 References

1. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning. Springer.

2. Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling. Springer.

3. Han, J., Kamber, M., & Pei, J. (2011). Data Mining: Concepts and Techniques. Morgan Kaufmann.

4. Lantz, B. (2019). Machine Learning with R: Expert techniques for predictive modeling. Packt Publishing.

5. Grolemund, G., & Wickham, H. (2017). R for Data Science. O'Reilly Media.

6. Tan, P., Steinbach, M., & Kumar, V. (2018). Introduction to Data Mining. Pearson.

Answers to Knowledge Check

Correct Options For Knowledge Check 1 :

1. b. WCSS
2. c. Initialization runs
3. c. Elbow point
4. b. Well-separated
5. b. Unsupervised segmentation

9.9 Case Study / Practical Exercise

Enhancing Business Strategy through Customer Clustering and

Background

FreshMart is mid-sized internet grocery platform which is under a steady growth trend. The company tracks a variety of favorable customer transactions, such as how often

they purchase, the size and contents of their baskets, most favored categories, and the payments used. But no clear customer segments and this means the marketing team has a tough time personalizing campaigns. Also, FreshMart hopes to identify potential customers who might take advantage of seasonal promotions.

In order to meet both requirements, we will employ clustering and classification algorithms through R using the analytics team.

Problem 1: Clustering Matching Customers to Segments Using K-means

The idea is to cluster your customers together into three distinct groups based on metrics such as: AOV – Average Order Value Frequency of purchase Number of product categories purchased from.

Solution

• Scale the data using scale().

• Use the function kmeans() with a particular number of clusters (K).

• Do the elbow method for finding optimal K.

• See clusters on PCA or cluster plots.

• Analyze results: Cluster 1 may be loyal high-value customers, Cluster 2 might be cost-aware yet infrequent buyers and Cluster 3- new/inactive customers.

This division assists in customizing our marketing approach that is unique to each group.

Problem Statement 2: Coupon Redemption Prediction Using Decision Trees

The company is hosting a new coupon campaign and it wants to predict which customers are likely to redeem the offer.

Solution

• Use data with target variable as Coupon_Redeemed (Yes/No).

• Expect a decision tree will be created using rpart() in R.

• Divide your data into training and test set.

• Assess with a confusion matrix and an ROC curve.

• Visual the decision tree and extract comprehensible rules.

The outcome enables the team to send coupons to only the likely responders, thus increasing ROI.

Problem Statement 3: Clustering Quality Estimation

The clustering result generated is doubtful to the management. You should verify that the clussters are well formed.

Solution

• Estimate the silhouette score for this clustering result.

• Plot silhouette widths as a tool to identify mis-assigned points.

• Re-factor using a variety of K's and compare scores.

• Apply internal validation metrics such as Calinski–Harabasz Index to compare.

This will make sure the clusters we select are based on meaningful customer differences, and not random.

Reflective Questions

How does classification and clustering help each other in customer analytics?

What are the dangers of using K-means without validating cluster quality?

What is need of data scaling before running k means?

What is the tension between accuracy and interpretability of decision trees?

Where do you use these outside of marketing, in operations or product development?

Conclusion

This use case demonstrates the value of clustering and classification in helping businesses become more informed and actionable about customer behavior. K-means clustering exposes data's latent structure, which helps with segmentation and personalization. Decision tree classification allows predictive targeting to maximise marketing expenditure. Collectively, these tools support decision making based on data in many areas of business strategy.