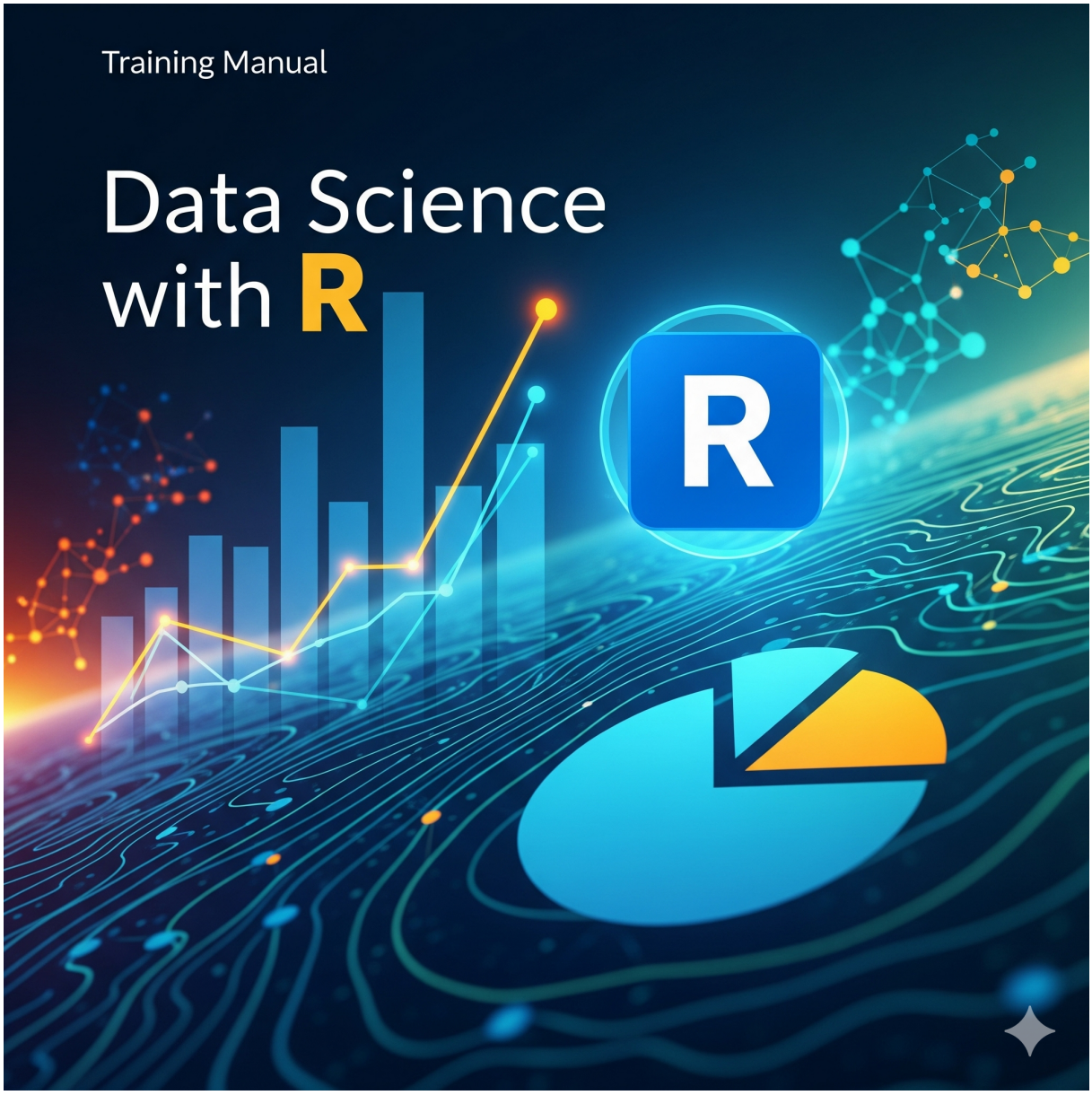
Training Manual



Instructor: JOÃO SOARES E SILVA

September 11, 2025

Data Science with R
Training Manual

Instructor: JOÃO SOARES E SILVA

September 11, 2025

Contents

1 Module 1 Data Collection and Preparation (10h)

1.1 Objectives

This module aims to equip students with the necessary skills to collect, clean, and prepare data in an efficient and reproducible manner. At the end of the 10 hours, participants should be able to:

- Acknowledge the importance of the data collection and preparation phase in the life cycle of a Data Science project. *Practical example:* In a retail company, sales data with registration errors can lead to incorrect stock forecasts.
- Import data from various sources, including local files, REST APIs, and relational databases. *Practical example:* A marketing analyst imports data from Google Ads via API and combines it with Excel files from the CRM.
- Apply data cleaning and transformation techniques with `tidyverse` and `janitor`. *Practical example:* In a public health study, normalize column names and remove duplicates before calculating rates.
- Implement data integrity and quality checks. *Practical example:* Verify that all postal codes have the correct format and that there are no duplicate records.
- Document the entire collection and preparation process. *Practical example:* Create an `RMarkdown` report with a step-by-step guide for data handling.

1.2 Developed Contents

- **Introduction to the *Data Collection* and *Data Preparation* process** *Practical example:* In a fintech company, poorly formatted transaction data can generate false fraud alerts.
- **Reading data:** CSV, Excel, APIs, databases. *Practical example:* Importing sales history from a POS.
- **Data cleaning:** name normalization, handling NAs and duplicates, type conversion. *Practical example:* Convert 01-03-2024 to a Date object.
- **Data transformation:** filtering, sorting, creating derived variables, reshaping. *Practical example:* Calculate profit margin and transform data to a long format.
- **Best practices:** organizing scripts, clear comments, use of `RMarkdown`.

1.3 Detailed Practical Activities

- Import and clean a real dataset from Kaggle or a public API.
- Create a custom function for recurring cleaning.
- Implement automatic integrity checks.
- Document the process in an `RMarkdown` file.

1.4 Resources and Tools

- **Software:** R and RStudio.
- **Packages:** `tidyverse`, `janitor`, `httr`, `jsonlite`, `DBI`, `lubridate`.
- **Data sources:** Kaggle, public APIs, test databases.

1.5 Case Study Complete Pipeline in R

Objective: Demonstrate all steps of the module in a single workflow.

```
library(tidyverse)
library(janitor)
library(lubridate)
library(httr)
library(jsonlite)

# 1. Import data
vendas <- read_csv("dados_vendas.csv")
resposta <- GET("https://api.exemplo.com/vendas")
dados_api <- fromJSON(content(resposta, "text"))
dados <- bind_rows(vendas, dados_api)

# 2. Initial cleaning
dados <- dados %>%
  clean_names() %>%
  distinct() %>%
  mutate(data_venda = dmy(data_venda))

# 3. Handling missing values
dados <- dados %>%
  mutate(
    preco = if_else(is.na(preco), mean(preco, na.rm = TRUE),
       preco),
    quantidade = replace_na(quantidade, 0)
  )

# 4. Derived variables
dados <- dados %>%
  mutate(receita = preco * quantidade)

# 5. Reshaping and aggregation
dados_mensal <- dados %>%
  mutate(mes = floor_date(data_venda, "month")) %>%
  group_by(mes, produto) %>%
  summarise(receita_total = sum(receita, na.rm = TRUE), .groups =
      "drop")

# 6. Validation
stopifnot(all(dados$preco >= 0))
stopifnot(!any(is.na(dados$data_venda)))
```

```r
# 7. Export
write_csv(dados, "dados_vendas_limpos.csv")
write_csv(dados_mensal, "dados_vendas_mensal.csv")
```

Notes for the Instructor:

- This pipeline covers import, cleaning, transformation, validation, and documentation.
- The dataset can be real or synthetic.
- Encourage adaptation for other data formats.

```r
library(tidyverse)
library(plotly)
library(corrplot)

# 1. Import dataset
dados <- read_csv("vendas.csv")

# 2. Descriptive statistics
summary(dados)
dados %>% summarise(
  media_preco = mean(preco, na.rm = TRUE),
  mediana_preco = median(preco, na.rm = TRUE),
  desvio_padrao_preco = sd(preco, na.rm = TRUE)
)

# 3. Basic visualizations
ggplot(dados, aes(x = preco)) + geom_histogram(binwidth = 5)
ggplot(dados, aes(x = categoria, y = preco)) + geom_boxplot()

# 4. Interactive visualization
grafico_interativo <- ggplot(dados, aes(x = preco, y = quantidade
   , color = categoria)) +
  geom_point()
ggplotly(grafico_interativo)

# 5. Correlation matrix
matriz <- cor(dados %>% select_if(is.numeric), use = "complete.
   obs")
corrplot(matriz, method = "color", type = "upper", tl.col = "
   black")

# 6. Outlier identification (IQR)
Q1 <- quantile(dados$preco, 0.25, na.rm = TRUE)
Q3 <- quantile(dados$preco, 0.75, na.rm = TRUE)
IQR <- Q3 - Q1
outliers <- dados %>% filter(preco < (Q1 - 1.5*IQR) | preco > (Q3
    + 1.5*IQR))
```

Notes for the Instructor: * The `vendas.csv` dataset can be real or synthetic, but it should contain both numerical and categorical variables. * Encourage students to

experiment with different chart types and parameters. * Discuss how EDA findings can influence subsequent modeling.

```
library(tidyverse)
library(caret)
library(tidymodels)

# 1. Import dataset
dados <- read_csv("precos_imoveis.csv")

# 2. Train/test split
set.seed(123)
divisao <- initial_split(dados, prop = 0.7)
treino <- training(divisao)
teste <- testing(divisao)

# 3. Preprocessing recipe
receita <- recipe(preco ~ ., data = treino) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_dummy(all_nominal_predictors())

# 4. Linear regression model
modelo_rl <- linear_reg() %>% set_engine("lm")

# 5. Workflow
workflow_rl <- workflow() %>%
  add_recipe(receita) %>%
  add_model(modelo_rl)

# 6. Train
modelo_treinado <- fit(workflow_rl, data = treino)

# 7. Evaluate
predicoes <- predict(modelo_treinado, teste) %>%
  bind_cols(teste)
metrics(predicoes, truth = preco, estimate = .pred)

# 8. Compare with Random Forest
modelo_rf <- rand_forest() %>%
  set_engine("ranger") %>%
  set_mode("regression")
workflow_rf <- workflow() %>%
  add_recipe(receita) %>%
  add_model(modelo_rf)
modelo_rf_treinado <- fit(workflow_rf, data = treino)
predicoes_rf <- predict(modelo_rf_treinado, teste) %>%
  bind_cols(teste)
metrics(predicoes_rf, truth = preco, estimate = .pred)
```

Notes for the Instructor: * This case study covers data preparation, training, evaluation, and algorithm comparison. * The dataset can be obtained from open sources or generated synthetically. * Encourage experimentation with different algorithms and

hyperparameters. * Discuss the choice of appropriate metrics for the problem.

2 Module 4 Reporting and Documentation (8h)

2.1 Objectives

By the end of this module, students should be able to:

- Understand the importance of clear and structured communication of results in Data Science projects.
- Create technical and executive reports that effectively convey findings and conclusions.
- Develop interactive dashboards for visualizing and monitoring metrics and indicators.
- Adapt communication for different target audiences (technical and non-technical).
- Document code, processes, and decisions to ensure reproducibility and future maintenance.

2.2 Developed Contents

- **Best practices for communicating results in Data Science** structuring messages in a clear, objective, and visually appealing manner.
- **Reports with `RMarkdown`:** technical document structure; including code, tables, and charts; exporting to HTML, PDF, and Word.
- **Interactive dashboards:** creation with `flexdashboard`; web applications with `shiny`; integration of dynamic visualizations (`plotly`, `leaflet`).
- **Storytelling with data:** narrative structure for presenting insights; using visualizations to reinforce key messages.
- **Code and process documentation:** clear and consistent comments; `README` files and user guides; version control with Git/GitHub.

2.3 Detailed Practical Activities

- Create a technical report in `RMarkdown` with exploratory analysis and predictive model results.
- Develop an interactive dashboard with `flexdashboard` or `shiny`.
- Prepare an executive presentation for a non-technical audience.
- Document the entire analysis process, including code, decisions, and data sources.

2.4 Resources and Tools

- **Software:** R and RStudio.
- **Packages:** `rmarkdown`, `flexdashboard`, `shiny`, `plotly`, `leaflet`.
- **Version control tools:** Git and GitHub.
- **Datasets:** datasets used in previous modules.

2.5 Case Study Report and Dashboard in R

Objective: Demonstrate the creation of a technical report and an interactive dashboard.

```
# RMarkdown Report (code chunk)
library(tidyverse)
dados <- read_csv("vendas.csv")

ggplot(dados, aes(x = mes, y = vendas, fill = regiao)) +
  geom_col(position = "dodge") +
  labs(title = "Monthly Sales by Region",
       x = "Month", y = "Total Sales")
```

Example of header and code for a flexdashboard:

````
---
title: "Sales Dashboard"
output: flexdashboard::flex_dashboard
---

## Sales by Region
```{r}
library(flexdashboard)
library(plotly)

grafico <- ggplot(dados, aes(x = regiao, y = vendas, fill =
   regiao)) +
  geom_bar(stat = "identity")
ggplotly(grafico)
````

Notes for the Instructor:

- The report and dashboard code should be placed in separate `.Rmd` files and run in RStudio.
- In the manual, the code is only shown as an example.
- Encourage personalization of the layout, colors, and visualization types.
- Demonstrate how to publish the dashboard on `shinyapps.io` or share the generated HTML.
````

3 Module 5 Research and Learning (6h)

3.1 Objectives

By the end of this module, students should be able to:

- Develop continuous research skills to stay updated in the fields of Data Science, Machine Learning, and Artificial Intelligence.
- Critically evaluate new libraries, techniques, and methodologies before adopting them.
- Integrate new tools and approaches into existing workflows.
- Share knowledge with the community and the work team.

3.2 Developed Contents

- **Information and update sources:** package repositories (CRAN, Bioconductor), blogs and communities (`R-bloggers`, Stack Overflow, Posit Community), scientific publications (arXiv, IEEE, Nature Machine Intelligence).
- **Evaluation of new tools:** evaluation criteria (performance, documentation, community, maintenance), integration tests in pilot projects.
- **Workflow integration:** adapting code and pipelines, automating repetitive tasks.
- **Knowledge sharing:** internal and external documentation, contributing to open-source projects.

3.3 Detailed Practical Activities

- Research and present a recent R package, explaining its functionalities and use cases.
- Implement a practical example with a newly discovered technique or package.
- Create a comparative document between two approaches for the same problem.
- Publish a small tutorial or code example in a GitHub repository.

3.4 Resources and Tools

Software: R and RStudio. **Packages:** variable according to research (e.g., `tidyverse`, `data.table`, `lightgbm`). **Data sources:** CRAN, GitHub, arXiv, public APIs.

3.5 Case Study Research and Integration of a New Package

Objective: Demonstrate how to identify, test, and integrate a new R package into a workflow.

```
# 1. Package research on CRAN
# Example: 'data.table' package for efficient data manipulation

# 2. Installation and loading
install.packages("data.table")
library(data.table)

# 3. Performance comparison with dplyr
library(tidyverse)
dados <- as.data.frame(matrix(runif(1e6), ncol = 10))

# Using dplyr
system.time({
  df_dplyr <- as_tibble(dados) %>%
    summarise(across(everything(), mean))
})

# Using data.table
system.time({
  dt <- as.data.table(dados)
  dt_means <- dt[, lapply(.SD, mean)]
})

# 4. Workflow integration
# Replace slow operations with data.table equivalents
# Document changes and performance gains

# 5. Sharing results
# Create a README with instructions and benchmarks
```

Notes for the Instructor:

- Encourage students to choose packages relevant to their interests or work area.
- Discuss objective criteria for adopting new tools.
- Show how to document and communicate test results to the team.

4 Module 6 Ad-hoc Analysis (8h)

4.1 Objectives

By the end of this module, students should be able to:

- Understand the role of **ad-hoc** analyses in quickly responding to specific business or research questions.
- Formulate hypotheses and structure targeted analyses to answer concrete questions.

- Select and apply statistical and visualization methods suitable for the problem.
- Communicate results clearly and objectively, focusing on the most relevant conclusions.

4.2 Developed Contents

- **Definition and context of ad-hoc analyses** difference between planned and ad-hoc analyses; when and why to use them.
- **Formulation of hypotheses and business questions** transforming vague questions into testable hypotheses.
- **Selection of methods and tools** choosing appropriate statistical techniques, filters, and visualizations.
- **Rapid execution and validation of results** ensuring that speed does not compromise quality.
- **Communication and delivery** delivery formats: brief report, email, short presentation.

4.3 Detailed Practical Activities

- Receive a simulated business question and develop an ad-hoc analysis to answer it.
- Create simple and effective visualizations to communicate results.
- Apply a statistical test suitable for the presented problem.
- Document the analysis in a concise and clear format.

4.4 Resources and Tools

- **Software:** R and RStudio.
- **Packages:** `tidyverse`, `lubridate`, `ggplot2`, `broom`.
- **Data sources:** internal or public datasets simulating real-world scenarios.

4.5 Case Study Ad-hoc Analysis in R

Objective: Demonstrate how to quickly answer a specific question with available data.

```
library(tidyverse)
library(lubridate)
library(broom)

# Scenario: The manager wants to know if the average sales of the
    last month
# were significantly different from the previous month.

# 1. Import data
```

```r
dados <- read_csv("vendas.csv") %>%
  mutate(data = dmy(data))

# 2. Filter last two months
ultimo_mes <- max(month(dados$data))
dados_filtrados <- dados %>%
  filter(month(data) %in% c(ultimo_mes, ultimo_mes - 1))

# 3. Descriptive statistics
dados_filtrados %>%
  group_by(mes = month(data)) %>%
  summarise(media_vendas = mean(vendas, na.rm = TRUE),
            sd_vendas = sd(vendas, na.rm = TRUE))

# 4. Statistical test (t-test)
resultado_t <- t.test(vendas ~ month(data), data =
   dados_filtrados)
tidy(resultado_t)

# 5. Quick visualization
ggplot(dados_filtrados, aes(x = factor(month(data)), y = vendas))
    +
  geom_boxplot(fill = "skyblue") +
  labs(x = "Month", y = "Sales", title = "Sales Comparison
     Between Months")

# 6. Conclusion
# If p-value < 0.05, there is a statistically significant
   difference.
```

Notes for the Instructor:

- This example shows how to quickly structure an analysis from data import to conclusion.
- Encourage students to adapt the code to other types of ad-hoc questions.
- Discuss the importance of communicating only the essentials in urgent analyses.

5 Module 7 Tooling and Infrastructure (6h)

5.1 Objectives

By the end of this module, students should be able to:

- Configure and maintain efficient development environments for Data Science projects.
- Automate tasks and workflows to increase productivity.
- Use version control tools to manage code and collaborate in a team.
- Integrate data and model pipelines into production environments.

5.2 Developed Contents

- **Project organization:** folder and file structure; consistent naming of files and scripts.
- **Dependency management:** use of `renv` to isolate environments; `DESCRIPTION` and `requirements.txt` files.
- **Version control:** Basic Git (commits, branches, merges); collaboration platforms (GitHub, GitLab).
- **Automation and pipelines:** scheduled scripts (cron jobs, tasks); continuous integration (CI) and continuous delivery (CD).

5.3 Detailed Practical Activities

- Create a Data Science project structure with organized folders and scripts.
- Configure an isolated environment with `renv` and install necessary packages.
- Create a Git repository and practice basic operations (commit, branch, merge).
- Configure a script to run automatically (scheduling).

5.4 Resources and Tools

- **Software:** R, RStudio, Git.
- **Packages:** `renv`, `targets`, `usethis`.
- **Platforms:** GitHub, GitLab.

5.5 Case Study Project Structuring and Automation

Objective: Demonstrate how to create an organized project, manage dependencies, and automate tasks.

```
# 1. Create project structure
usethis::create_project("sales_project")
dir.create("data_raw")
dir.create("data_processed")
dir.create("scripts")
dir.create("reports")

# 2. Initiate version control
usethis::use_git()

# 3. Configure isolated environment
install.packages("renv")
renv::init()

# 4. Import and cleaning script (scripts/01_importation.R)
library(tidyverse)
```

```r
dados <- read_csv("data_raw/sales.csv") %>%
  janitor::clean_names() %>%
  filter(!is.na(sales_value))
write_csv(dados, "data_processed/clean_sales.csv")

# 5. Automate daily execution (example on Unix system with cron)
# Open crontab: crontab -e
# Add line to run script at 2 a.m.:
# 0 2 * * * Rscript /path/to/sales_project/scripts/01_importation
   .R

# 6. Version changes and sync with GitHub
# git add .
# git commit -m "Initial structure and import script"
# git push origin main
```

Notes for the Instructor:

- This case study shows how to combine organization, dependency management, version control, and automation.
- Encourage students to adapt the structure and scripts to their own project needs.
- Discuss the importance of keeping a README updated with execution instructions.

6 Module 8 Model Deployment and Maintenance (6h)

6.1 Objectives

By the end of this module, students should be able to:

- Understand the process of making Machine Learning models available in production environments.
- Implement APIs to serve model predictions in real-time or in batch.
- Monitor the performance of models in production and detect **data drift** or performance degradation.
- Perform maintenance and periodic re-training of models to ensure relevance and accuracy.
- Document and version models for traceability and auditing.

6.2 Developed Contents

- **Deployment concepts:** difference between development, testing, and production environments; deployment modes (batch, real-time, **edge deployment**).
- **Serving models via API:** use of the `plumber` package to create REST APIs in R; API security and authentication.

- **Model monitoring:** performance metrics in production; detection of **data drift** and **concept drift**.
- **Maintenance and re-training:** re-training strategies (scheduled, event-based); automation of re-training pipelines.
- **Model versioning and documentation:** storing models with `pins` or `modeltime`; logging changes and auditing.

6.3 Detailed Practical Activities

- Create an API with `plumber` to serve predictions from a trained model.
- Implement a monitoring script that logs performance metrics daily.
- Simulate **data drift** and perform automatic re-training.
- Document the deployment and maintenance process in a technical report.

6.4 Resources and Tools

Software: R and RStudio. **Packages:** `plumber`, `pins`, `modeltime`, `tidyverse`. **Platforms:** Local servers, `shinyapps.io`, Docker.

6.5 Case Study Prediction API with `plumber`

Objective: Demonstrate how to make a trained model available via a REST API.

```
# 1. Train and save model
library(tidyverse)
library(caret)
dados <- read_csv("dados_clientes.csv")

modelo <- train(churn ~ ., data = dados, method = "glm", family =
    "binomial")
saveRDS(modelo, "modelo_churn.rds")

# 2. Create API with plumber (file: api.R)
library(plumber)

#* @apiTitle Churn Prediction API
#* @param idade:int Client age
#* @param rendimento:double Annual income
#* @post /prever
function(idade, rendimento){
  modelo <- readRDS("modelo_churn.rds")
  novo_dado <- data.frame(idade = as.integer(idade),
                          rendimento = as.numeric(rendimento))
  prob <- predict(modelo, novo_dado, type = "prob")[,2]
  list(probabilidade_churn = prob)
}
```

```
# 3. Run API
# plumber::pr("api.R") %>% pr_run(port = 8000)

# 4. Test API
# POST to http://localhost:8000/prever with JSON:
# {"idade": 35, "rendimento": 45000}
```

Notes for the Instructor:

- This example shows a simple deployment flow with `plumber`.
- Encourage students to add authentication and logging to the API.
- Discuss how to integrate this API into a larger system (e.g., dashboard, web application).

7 Module 9 Evaluation (4h)

7.1 Objectives

By the end of this module, students should be able to:

- Define evaluation criteria and instruments suitable for the course objectives.
- Apply different types of evaluation (diagnostic, formative, and summative) throughout the course.
- Provide constructive feedback aimed at continuous improvement.
- Use digital tools to collect, record, and analyze evaluation results.
- Document the evaluation process to ensure transparency and traceability.

7.2 Developed Contents

- **Types of evaluation:** diagnostic, formative, and summative.
- **Performance criteria and indicators:** clarity, objectivity, and alignment with course objectives.
- **Evaluation instruments:** evaluation grids, rubrics, tests, presentations; digital tools (Google Forms, Microsoft Forms, Moodle).
- **Constructive feedback:** "strengths areas for improvement suggestions" structure; assertive communication techniques.
- **Recording and documentation:** secure storage of results; evaluation reports and feedback meeting minutes.

7.3 Detailed Practical Activities

- Create an evaluation grid for a final project, with defined criteria and weights.
- Apply the grid to a sample work and calculate the final grade.
- Prepare a feedback report based on the results.
- Simulate the use of a digital tool to collect and analyze results.

7.4 Resources and Tools

Software: R and RStudio, Google Forms, Microsoft Forms, Moodle. **R Packages:** `dplyr`, `readr`, `knitr`. **Documents:** templates for grids and rubrics.

7.5 Case Study Evaluation Grid in R

Objective: Demonstrate how to create and apply an evaluation grid with automatic final grade calculation.

```r
library(dplyr)

# 1. Define criteria and weights
criterios <- data.frame(
  criterio = c("Technical Quality", "Creativity", "Communication"
     , "Documentation"),
  peso = c(0.4, 0.2, 0.2, 0.2),
  pontuacao = c(4, 5, 4, 3) # example evaluation (1 to 5)
)

# 2. Calculate final grade (0-20 scale)
criterios <- criterios %>%
  mutate(resultado = peso * pontuacao)

nota_final <- sum(criterios$resultado) / sum(criterios$peso) * 4
   # 5 points = 20 values

# 3. Show results
criterios
nota_final

# 4. Export simple report
library(knitr)
kable(criterios, col.names = c("Criterion", "Weight", "Score", "
   Result"))
cat("Final Grade:", round(nota_final, 2), "/ 20")
```

Notes for the Instructor:

- This example shows how to use R to automate grade calculation based on a grid.
- Encourage students to adapt criteria and weights to the type of project.
- Discuss the importance of keeping clear and accessible records for auditing.