Manual de Formação em Ciência de Dados com R

Formador: JOÃO SOARES E SILVA

8 de setembro de 2025

Conteúdo

1 Módulo 1 – Data Collection and Preparation (10h)

1.1 Objetivos

Este módulo visa dotar os formandos das competências necessárias para recolher, limpar e preparar dados de forma eficiente e reprodutível. Ao final das 10 horas, os participantes deverão ser capazes de:

- Reconhecer a importância da fase de recolha e preparação de dados no ciclo de vida de um projeto de Ciência de Dados. *Exemplo prático:* Numa empresa de retalho, dados de vendas com erros de registo podem levar a previsões de stock incorretas.
- Importar dados de diversas fontes, incluindo ficheiros locais, APIs REST e bases de dados relacionais. *Exemplo prático:* Um analista de marketing importa dados do Google Ads via API e combina-os com ficheiros Excel do CRM.
- Aplicar técnicas de limpeza e transformação de dados com `tidyverse` e `janitor`. *Exemplo prático:* Num estudo de saúde pública, normalizar nomes de colunas e remover duplicados antes de calcular taxas.
- Implementar verificações de integridade e qualidade dos dados. *Exemplo prático:* Verificar se todos os códigos postais têm o formato correto e se não existem registos duplicados.
- Documentar todo o processo de recolha e preparação. *Exemplo prático:* Criar um relatório `RMarkdown` com o passo a passo do tratamento de dados.

1.2 Conteúdos Desenvolvidos

- **Introdução ao processo de *Data Collection* e *Data Preparation*** *Exemplo prático:* Numa fintech, dados de transações mal formatados podem gerar alertas falsos de fraude.
- **Leitura de dados:** CSV, Excel, APIs, bases de dados. *Exemplo prático:* Importar histórico de vendas de um POS.
- **Limpeza de dados:** normalização de nomes, tratamento de NAs e duplicados, conversão de tipos. *Exemplo prático:* Converter "01-03-2024" para objeto Date.
- **Transformação de dados:** filtragem, ordenação, criação de variáveis derivadas, reshaping. *Exemplo prático:* Calcular margem de lucro e transformar dados para formato longo.
- **Boas práticas:** organização de scripts, comentários claros, uso de `RMarkdown`.

1.3 Atividades Práticas Detalhadas

- Importar e limpar um dataset real do Kaggle ou API pública.
- Criar função personalizada para limpeza recorrente.
- Implementar verificações automáticas de integridade.
- Documentar o processo num `RMarkdown`.

1.4 Recursos e Ferramentas

- **Software:** R e RStudio.
- **Pacotes:** `tidyverse`, `janitor`, `httr`, `jsonlite`, `DBI`, `lubridate`.
- **Fontes de dados:** Kaggle, APIs públicas, bases de dados de teste.

1.5 Estudo de Caso – Pipeline Completo em R

Objetivo: Demonstrar todas as etapas do módulo num único fluxo de trabalho.

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

# 1. Importar dados
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. Limpeza inicial
dados <- dados %>%
  clean_names() %>%
  distinct() %>%
  mutate(data_venda = dmy(data_venda))

# 3. Tratamento de valores ausentes
dados <- dados %>%
  mutate(
    preco = if_else(is.na(preco), mean(preco, na.rm = TRUE), preco)
       ,
    quantidade = replace_na(quantidade, 0)
  )

# 4. Vari veis derivadas
dados <- dados %>%
  mutate(receita = preco * quantidade)

# 5. Reshaping e agrega  o
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. Valida  o
stopifnot(all(dados$preco >= 0))
stopifnot(!any(is.na(dados$data_venda)))
```

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

Notas para o Formador:

- Este pipeline cobre importação, limpeza, transformação, validação e documentação.
- O dataset pode ser real ou sintético.
- Incentivar a adaptação para outros formatos de dados.

2 Módulo 2 – Exploratory Data Analysis (EDA) (10h)

2.1 Objetivos

Ao final deste módulo, os formandos deverão ser capazes de:

- Compreender o papel da Análise Exploratória de Dados (EDA) no ciclo de vida de um projeto de Ciência de Dados. *Exemplo prático:* Antes de criar um modelo de previsão de vendas, uma empresa analisa a distribuição das vendas por região para identificar padrões sazonais.
- Explorar dados utilizando estatísticas descritivas e visualizações gráficas para obter uma compreensão inicial do conjunto de dados. *Exemplo prático:* Um hospital calcula médias e desvios padrão de tempos de espera para identificar gargalos no atendimento.
- Identificar padrões, tendências, outliers e relações entre variáveis. *Exemplo prático:* Uma seguradora deteta clientes com valores de indemnização anormalmente altos, investigando possíveis fraudes.
- Formular hipóteses iniciais com base nas observações obtidas. *Exemplo prático:* Após observar que clientes mais jovens compram mais online, uma loja formula a hipótese de que campanhas digitais terão maior impacto nesse grupo.
- Comunicar de forma clara os principais achados da análise exploratória. *Exemplo prático:* Um analista apresenta à direção um dashboard com gráficos interativos mostrando tendências de vendas por produto e região.

2.2 Conteúdos Desenvolvidos

- **Introdução à EDA e sua importância** *Exemplo prático:* Numa startup de mobilidade, a EDA revelou que a maioria das viagens curtas ocorre em horários de pico.
- **Estatísticas descritivas:** média, mediana, moda, variância, desvio padrão, amplitude, distribuições de frequência e percentis. *Exemplo prático:* Calcular a mediana de salários para evitar distorções causadas por valores extremos.

- **Visualização de dados:** histogramas, boxplots, scatterplots, gráficos de barras com `ggplot2`; visualizações interativas com `plotly`; mapas de calor e matrizes de correlação com `corrplot`. *Exemplo prático:* Criar um histograma da idade dos clientes para segmentar campanhas.
- **Identificação de outliers:** métodos gráficos e estatísticos (IQR, Z-score). *Exemplo prático:* Detetar transações bancárias suspeitas com valores muito acima da média.
- **Análise de correlação:** Pearson, Spearman, Kendall e interpretação de coeficientes. *Exemplo prático:* Avaliar a relação entre temperatura e consumo de gelados.
- **Boas práticas na apresentação de resultados:** gráficos claros, evitar sobrecarga de informação, contextualizar achados. *Exemplo prático:* Apresentar apenas as 5 variáveis mais relevantes num relatório executivo.

2.3 Atividades Práticas Detalhadas

- Calcular estatísticas descritivas para um dataset real.
- Criar visualizações com `ggplot2` para explorar distribuições e relações.
- Utilizar `plotly` para criar gráficos interativos.
- Gerar e interpretar uma matriz de correlação com `corrplot`.
- Identificar e documentar padrões, outliers e correlações relevantes.

2.4 Recursos e Ferramentas

- **Software:** R e RStudio.
- **Pacotes:** `ggplot2`, `plotly`, `corrplot`, `dplyr`.
- **Datasets de apoio:** vendas online, dados de saúde pública, dados meteorológicos.

2.5 Estudo de Caso – EDA Completa em R

Objetivo: Demonstrar todas as etapas da EDA num único fluxo de trabalho.

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

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

# 2. Estat sticas descritivas
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)
)
```

```r
# 3. Visualizações básicas
ggplot(dados, aes(x = preco)) + geom_histogram(binwidth = 5)
ggplot(dados, aes(x = categoria, y = preco)) + geom_boxplot()

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

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

# 6. Identificação de outliers (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))
```

Notas para o Formador:

- O dataset `vendas.csv` pode ser real ou sintético, mas deve conter variáveis numéricas e categóricas.
- Incentivar os alunos a experimentar diferentes tipos de gráficos e parâmetros.
- Discutir como as descobertas da EDA podem influenciar a modelagem subsequente.

3 Módulo 3 – Model Development Support (12h)

3.1 Objetivos

Ao final deste módulo, os formandos deverão ser capazes de:

- Compreender o papel do desenvolvimento de modelos no ciclo de vida de um projeto de Ciência de Dados. *Exemplo prático:* Uma empresa de logística desenvolve um modelo para prever atrasos nas entregas com base em dados históricos e meteorológicos.
- Apoiar na criação de modelos preditivos supervisionados e não supervisionados.
- Realizar engenharia de variáveis (*feature engineering*) para melhorar a performance dos modelos.
- Selecionar e aplicar algoritmos adequados ao tipo de problema.
- Avaliar modelos utilizando métricas apropriadas e técnicas de validação.
- Documentar e comunicar o processo de modelagem e os resultados obtidos.

3.2 Conteúdos Desenvolvidos

- **Introdução ao fluxo de trabalho de modelagem preditiva** — preparação de dados, treino, validação, teste e implementação.
- **Preparação dos dados:** divisão treino/validação/teste; normalização e padronização; codificação de variáveis categóricas.
- **Engenharia de variáveis:** criação de novas variáveis; seleção de variáveis relevantes; redução de dimensionalidade (PCA).
- **Treino de modelos:** uso de `caret`, `tidymodels` e `recipes`; algoritmos comuns (regressão linear/logística, árvores, random forest, k-NN).
- **Avaliação de modelos:** métricas para regressão (RMSE, MAE, R^2) e classificação (Acurácia, Precisão, Recall, F1-score, AUC); validação cruzada.
- **Comparação e seleção de modelos**.

3.3 Atividades Práticas Detalhadas

- Criar um modelo de regressão para prever preços de imóveis utilizando `caret`.
- Desenvolver um modelo de classificação para prever churn de clientes com `tidymodels`.
- Implementar um `recipe` para normalizar dados, criar variáveis derivadas e codificar variáveis categóricas.
- Comparar o desempenho de pelo menos dois algoritmos diferentes para o mesmo problema.
- Documentar o processo de modelagem e apresentar os resultados com métricas e gráficos.

3.4 Recursos e Ferramentas

- **Software:** R e RStudio.
- **Pacotes:** `caret`, `tidymodels`, `recipes`, `ggplot2`, `dplyr`.
- **Datasets:** preços de imóveis, churn de clientes, datasets públicos do UCI Machine Learning Repository.

3.5 Estudo de Caso – Desenvolvimento de Modelo em R

Objetivo: Demonstrar todas as etapas do desenvolvimento de um modelo preditivo.

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

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

```r
# 2. Divis o treino/teste
set.seed(123)
divisao <- initial_split(dados, prop = 0.7)
treino <- training(divisao)
teste <- testing(divisao)

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

# 4. Modelo regress o linear
modelo_rl <- linear_reg() %>% set_engine("lm")

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

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

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

# 8. Comparar com 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)
```

Notas para o Formador:

- Este estudo de caso cobre preparação de dados, treino, avaliação e comparação de algoritmos.
- O dataset pode ser obtido de fontes abertas ou gerado sinteticamente.
- Incentivar a experimentação com diferentes algoritmos e hiperparâmetros.
- Discutir a escolha de métricas adequadas ao problema.

4 Módulo 4 – Reporting and Documentation (8h)

4.1 Objetivos

Ao final deste módulo, os formandos deverão ser capazes de:

- Compreender a importância da comunicação clara e estruturada dos resultados em projetos de Ciência de Dados.
- Criar relatórios técnicos e executivos que transmitam de forma eficaz as descobertas e conclusões.
- Desenvolver dashboards interativos para visualização e monitorização de métricas e indicadores.
- Adaptar a comunicação para diferentes públicos-alvo (técnico e não técnico).
- Documentar o código, processos e decisões para garantir reprodutibilidade e manutenção futura.

4.2 Conteúdos Desenvolvidos

- **Boas práticas de comunicação de resultados em Data Science** — estruturar mensagens de forma clara, objetiva e visualmente apelativa.
- **Relatórios com `RMarkdown`:** estrutura de um documento técnico; inclusão de código, tabelas e gráficos; exportação para HTML, PDF e Word.
- **Dashboards interativos:** criação com `flexdashboard`; aplicações web com `shiny`; integração de visualizações dinâmicas (`plotly`, `leaflet`).
- **Storytelling com dados:** estrutura narrativa para apresentação de insights; uso de visualizações para reforçar mensagens-chave.
- **Documentação de código e processos:** comentários claros e consistentes; ficheiros `README` e guias de utilização; versionamento com Git/GitHub.

4.3 Atividades Práticas Detalhadas

- Criar um relatório técnico em `RMarkdown` com análise exploratória e resultados de um modelo preditivo.
- Desenvolver um dashboard interativo com `flexdashboard` ou `shiny`.
- Preparar uma apresentação executiva para um público não técnico.
- Documentar todo o processo de análise, incluindo código, decisões e fontes de dados.

4.4 Recursos e Ferramentas

- **Software:** R e RStudio.
- **Pacotes:** `rmarkdown`, `flexdashboard`, `shiny`, `plotly`, `leaflet`.
- **Ferramentas de versionamento:** Git e GitHub.
- **Datasets:** conjuntos de dados utilizados nos módulos anteriores.

4.5 Estudo de Caso – Relatório e Dashboard em R

Objetivo: Demonstrar a criação de um relatório técnico e de um dashboard interativo.

```
# Relat rio RMarkdown (chunk de c digo)
library(tidyverse)
dados <- read_csv("vendas.csv")

ggplot(dados, aes(x = mes, y = vendas, fill = regiao)) +
  geom_col(position = "dodge") +
  labs(title = "Vendas Mensais por Regi o",
       x = "M s", y = "Total de Vendas")
```

Exemplo de cabeçalho e código para dashboard com flexdashboard:

```
---
title: "Dashboard de Vendas"
output: flexdashboard::flex_dashboard
---

## Vendas por Regi o
```{r}
library(flexdashboard)
library(plotly)

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

Notas para o Formador:

- O código do relatório e do dashboard deve ser colocado em ficheiros `.Rmd` separados e executado no RStudio.
- No manual, o código é apenas exibido como exemplo.
- Incentivar a personalização do layout, cores e tipos de visualização.
- Demonstrar como publicar o dashboard no `shinyapps.io` ou partilhar o HTML gerado.

5 Módulo 5 – Research and Learning (6h)

5.1 Objetivos

Ao final deste módulo, os formandos deverão ser capazes de:

- Desenvolver competências de pesquisa contínua para se manterem atualizados nas áreas de Ciência de Dados, Machine Learning e Inteligência Artificial.
- Avaliar criticamente novas bibliotecas, técnicas e metodologias antes de as adotar.
- Integrar novas ferramentas e abordagens nos fluxos de trabalho existentes.
- Partilhar conhecimento com a comunidade e com a equipa de trabalho.
```

5.2 Conteúdos Desenvolvidos

- **Fontes de informação e atualização:** repositórios de pacotes (CRAN, Bioconductor), blogs e comunidades (`R-bloggers`, Stack Overflow, Posit Community), publicações científicas (arXiv, IEEE, Nature Machine Intelligence).
- **Avaliação de novas ferramentas:** critérios de avaliação (desempenho, documentação, comunidade, manutenção), testes de integração em projetos piloto.
- **Integração no fluxo de trabalho:** adaptação de código e pipelines, automação de tarefas repetitivas.
- **Partilha de conhecimento:** documentação interna e externa, contribuição para projetos open-source.

5.3 Atividades Práticas Detalhadas

- Pesquisar e apresentar um pacote R recente, explicando as suas funcionalidades e casos de uso.
- Implementar um exemplo prático com uma técnica ou pacote recém-descoberto.
- Criar um documento comparativo entre duas abordagens para o mesmo problema.
- Publicar um pequeno tutorial ou exemplo de código num repositório GitHub.

5.4 Recursos e Ferramentas

- **Software:** R e RStudio.
- **Pacotes:** variáveis conforme a pesquisa (ex.: `tidyverse`, `data.table`, `lightgbm`).
- **Fontes de dados:** CRAN, GitHub, arXiv, APIs públicas.

5.5 Estudo de Caso – Pesquisa e Integração de um Novo Pacote

Objetivo: Demonstrar como identificar, testar e integrar um novo pacote R num fluxo de trabalho.

```
# 1. Pesquisa de pacote no CRAN
# Exemplo: pacote 'data.table' para manipula  o eficiente de
   dados

# 2. Instala  o e carregamento
install.packages("data.table")
library(data.table)

# 3. Compara  o de desempenho com dplyr
library(tidyverse)
dados <- as.data.frame(matrix(runif(1e6), ncol = 10))

# Usando dplyr
system.time({
```

```
  df_dplyr <- as_tibble(dados) %>%
    summarise(across(everything(), mean))
})

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

# 4. Integra  o no fluxo de trabalho
# Substituir opera  es lentas por equivalentes em data.table
# Documentar altera  es e ganhos de desempenho

# 5. Partilha de resultados
# Criar um README com instru  es e benchmarks
```

Notas para o Formador:

- Incentivar os alunos a escolher pacotes relevantes para os seus interesses ou área de trabalho.
- Discutir critérios objetivos para adoção de novas ferramentas.
- Mostrar como documentar e comunicar resultados de testes à equipa.

==========

6 Módulo 6 – Ad-hoc Analysis (8h)

6.1 Objetivos

Ao final deste módulo, os formandos deverão ser capazes de:

- Compreender o papel das análises *ad-hoc* na resposta rápida a questões específicas de negócio ou investigação.
- Formular hipóteses e estruturar análises direcionadas para responder a perguntas concretas.
- Selecionar e aplicar métodos estatísticos e de visualização adequados ao problema.
- Comunicar resultados de forma clara e objetiva, focando nas conclusões mais relevantes.

6.2 Conteúdos Desenvolvidos

- **Definição e contexto de análises *ad-hoc*** — diferença entre análises planeadas e *ad-hoc*; quando e porquê utilizá-las.
- **Formulação de hipóteses e perguntas de negócio** — transformar questões vagas em hipóteses testáveis.

- **Seleção de métodos e ferramentas** — escolha de técnicas estatísticas, filtros e visualizações adequadas.
- **Execução rápida e validação de resultados** — garantir que a rapidez não compromete a qualidade.
- **Comunicação e entrega** — formatos de entrega: relatório breve, e-mail, apresentação curta.

6.3 Atividades Práticas Detalhadas

- Receber uma questão de negócio simulada e desenvolver uma análise *ad-hoc* para respondê-la.
- Criar visualizações simples e eficazes para comunicar resultados.
- Aplicar um teste estatístico adequado ao problema apresentado.
- Documentar a análise num formato conciso e claro.

6.4 Recursos e Ferramentas

- **Software:** R e RStudio.
- **Pacotes:** `tidyverse`, `lubridate`, `ggplot2`, `broom`.
- **Fontes de dados:** datasets internos ou públicos simulando cenários reais.

6.5 Estudo de Caso – Análise *Ad-hoc* em R

Objetivo: Demonstrar como responder rapidamente a uma questão específica com dados disponíveis.

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

# Cen rio: O gestor quer saber se a m dia de vendas do  ltimo
   m s
# foi significativamente diferente do m s anterior.

# 1. Importar dados
dados <- read_csv("vendas.csv") %>%
  mutate(data = dmy(data))

# 2. Filtrar  ltimos  dois meses
ultimo_mes <- max(month(dados$data))
dados_filtrados <- dados %>%
  filter(month(data) %in% c(ultimo_mes, ultimo_mes - 1))

# 3. Estat sticas descritivas
dados_filtrados %>%
  group_by(mes = month(data)) %>%
```

```
  summarise(media_vendas = mean(vendas, na.rm = TRUE),
            sd_vendas = sd(vendas, na.rm = TRUE))

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

# 5. Visualiza  o r pida
ggplot(dados_filtrados, aes(x = factor(month(data)), y = vendas)) +
  geom_boxplot(fill = "skyblue") +
  labs(x = "M  s", y = "Vendas", title = "Compara  o de Vendas
     entre Meses")

# 6. Conclus o
# Se p-valor < 0.05, h  diferen a estatisticamente significativa.
```

Notas para o Formador:

- Este exemplo mostra como estruturar rapidamente uma análise desde a importação de dados até à conclusão.
- Incentivar os alunos a adaptar o código a outros tipos de questões *ad-hoc*.
- Discutir a importância de comunicar apenas o essencial em análises urgentes.

7 Módulo 7 – Tooling and Infrastructure (6h)

7.1 Objetivos

Ao final deste módulo, os formandos deverão ser capazes de:

- Configurar e manter ambientes de desenvolvimento eficientes para projetos de Ciência de Dados.
- Automatizar tarefas e fluxos de trabalho para aumentar a produtividade.
- Utilizar ferramentas de controlo de versões para gerir código e colaborar em equipa.
- Integrar pipelines de dados e modelos em ambientes de produção.

7.2 Conteúdos Desenvolvidos

- **Organização de projetos:** estrutura de pastas e ficheiros; nomeação consistente de ficheiros e scripts.
- **Gestão de dependências:** uso do `renv` para isolar ambientes; ficheiros `DESCRIPTION` e `requirements.txt`.
- **Controlo de versões:** Git básico (commits, branches, merges); plataformas de colaboração (GitHub, GitLab).
- **Automação e pipelines:** scripts agendados (cron jobs, tasks); integração contínua (CI) e entrega contínua (CD).

7.3 Atividades Práticas Detalhadas

- Criar a estrutura de um projeto de Ciência de Dados com pastas e scripts organizados.
- Configurar um ambiente isolado com `renv` e instalar pacotes necessários.
- Criar um repositório Git e praticar operações básicas (commit, branch, merge).
- Configurar um script para ser executado automaticamente (agendamento).

7.4 Recursos e Ferramentas

- **Software:** R, RStudio, Git.
- **Pacotes:** `renv`, `targets`, `usethis`.
- **Plataformas:** GitHub, GitLab.

7.5 Estudo de Caso – Estruturação e Automação de um Projeto

Objetivo: Demonstrar como criar um projeto organizado, gerir dependências e automatizar tarefas.

```
# 1. Criar estrutura de projeto
usethis::create_project("projeto_vendas")
dir.create("data_raw")
dir.create("data_processed")
dir.create("scripts")
dir.create("reports")

# 2. Iniciar controlo de vers es
usethis::use_git()

# 3. Configurar ambiente isolado
install.packages("renv")
renv::init()

# 4. Script de importa  o e limpeza (scripts/01_importacao.R)
library(tidyverse)
dados <- read_csv("data_raw/vendas.csv") %>%
  janitor::clean_names() %>%
  filter(!is.na(valor_venda))
write_csv(dados, "data_processed/vendas_limpo.csv")

# 5. Automatizar execu  o di ria (exemplo em sistema Unix com
   cron)
# Abrir crontab: crontab -e
# Adicionar linha para executar script  s  2h da manh :
# 0 2 * * * Rscript /caminho/projeto_vendas/scripts/01_importacao.R

# 6. Versionar altera  es e sincronizar com GitHub
# git add .
```

```
# git commit -m "Estrutura inicial e script de importa  o"
# git push origin main
```

Notas para o Formador:

- Este estudo de caso mostra como combinar organização, gestão de dependências, controlo de versões e automação.
- Incentivar os alunos a adaptar a estrutura e scripts às necessidades dos seus próprios projetos.
- Discutir a importância de manter um README atualizado com instruções de execução.

8 Módulo 8 – Deployment e Manutenção de Modelos (6h)

8.1 Objetivos

Ao final deste módulo, os formandos deverão ser capazes de:

- Compreender o processo de disponibilização de modelos de Machine Learning em ambientes de produção.
- Implementar APIs para servir previsões de modelos em tempo real ou em lote.
- Monitorizar o desempenho de modelos em produção e detetar *data drift* ou degradação de performance.
- Realizar manutenção e re-treino periódico de modelos para garantir relevância e precisão.
- Documentar e versionar modelos para rastreabilidade e auditoria.

8.2 Conteúdos Desenvolvidos

- **Conceitos de deployment:** diferença entre ambientes de desenvolvimento, teste e produção; modos de disponibilização (batch, tempo real, *edge deployment*).
- **Servir modelos via API:** uso do pacote `plumber` para criar APIs REST em R; segurança e autenticação de APIs.
- **Monitorização de modelos:** métricas de desempenho em produção; deteção de *data drift* e *concept drift*.
- **Manutenção e re-treino:** estratégias de re-treino (agendado, baseado em eventos); automação de pipelines de re-treino.
- **Versionamento e documentação de modelos:** armazenamento de modelos com `pins` ou `modeltime`; registo de alterações e auditoria.

8.3 Atividades Práticas Detalhadas

- Criar uma API com `plumber` para servir previsões de um modelo treinado.
- Implementar um script de monitorização que regista métricas de desempenho diariamente.
- Simular *data drift* e executar re-treino automático.
- Documentar o processo de deployment e manutenção num relatório técnico.

8.4 Recursos e Ferramentas

- **Software:** R e RStudio.
- **Pacotes:** `plumber`, `pins`, `modeltime`, `tidyverse`.
- **Plataformas:** Servidores locais, `shinyapps.io`, Docker.

8.5 Estudo de Caso – API de Previsões com `plumber`

Objetivo: Demonstrar como disponibilizar um modelo treinado via API REST.

```
# 1. Treinar e guardar modelo
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. Criar API com plumber (ficheiro: api.R)
library(plumber)

#* @apiTitle API de Previs o de Churn
#* @param idade:int Idade do cliente
#* @param rendimento:double Rendimento anual
#* @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. Executar API
# plumber::pr("api.R") %>% pr_run(port = 8000)

# 4. Testar API
# POST para http://localhost:8000/prever com JSON:
# {"idade": 35, "rendimento": 45000}
```

Notas para o Formador:

- Este exemplo mostra um fluxo simples de deployment com `plumber`.
- Incentivar os alunos a adicionar autenticação e logging à API.
- Discutir como integrar esta API num sistema maior (ex.: dashboard, aplicação web).

9 Módulo 9 – Avaliação (4h)

9.1 Objetivos

Ao final deste módulo, os formandos deverão ser capazes de:

- Definir critérios e instrumentos de avaliação adequados aos objetivos da formação.
- Aplicar diferentes tipos de avaliação (diagnóstica, formativa e sumativa) ao longo do curso.
- Fornecer feedback construtivo e orientado à melhoria contínua.
- Utilizar ferramentas digitais para recolher, registar e analisar resultados de avaliação.
- Documentar o processo de avaliação para garantir transparência e rastreabilidade.

9.2 Conteúdos Desenvolvidos

- **Tipos de avaliação:** diagnóstica, formativa e sumativa.
- **Critérios e indicadores de desempenho:** clareza, objetividade e alinhamento com os objetivos do curso.
- **Instrumentos de avaliação:** grelhas de avaliação, rubricas, testes, apresentações; ferramentas digitais (Google Forms, Microsoft Forms, Moodle).
- **Feedback construtivo:** estrutura "pontos fortes – pontos a melhorar – sugestões"; técnicas de comunicação assertiva.
- **Registo e documentação:** armazenamento seguro de resultados; relatórios de avaliação e atas de reuniões de feedback.

9.3 Atividades Práticas Detalhadas

- Criar uma grelha de avaliação para um projeto final, com critérios e pesos definidos.
- Aplicar a grelha a um trabalho exemplo e calcular a nota final.
- Elaborar um relatório de feedback com base nos resultados.
- Simular a utilização de uma ferramenta digital para recolher e analisar resultados.

9.4 Recursos e Ferramentas

- **Software:** R e RStudio, Google Forms, Microsoft Forms, Moodle.
- **Pacotes R:** `dplyr`, `readr`, `knitr`.
- **Documentos:** modelos de grelhas e rubricas.

9.5 Estudo de Caso – Grelha de Avaliação em R

Objetivo: Demonstrar como criar e aplicar uma grelha de avaliação com cálculo automático da nota final.

```
library(dplyr)

# 1. Definir crit rios e pesos
criterios <- data.frame(
  criterio = c("Qualidade T cnica", "Criatividade", "Comunica  o
     ", "Documenta  o"),
  peso = c(0.4, 0.2, 0.2, 0.2),
  pontuacao = c(4, 5, 4, 3) # exemplo de avalia  o (1 a 5)
)

# 2. Calcular nota final (escala 0-20)
criterios <- criterios %>%
  mutate(resultado = peso * pontuacao)

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

# 3. Mostrar resultados
criterios
nota_final

# 4. Exportar relat rio simples
library(knitr)
kable(criterios, col.names = c("Crit rio", "Peso", "Pontua  o",
   "Resultado"))
cat("Nota Final:", round(nota_final, 2), "/ 20")
```

Notas para o Formador:

- Este exemplo mostra como usar R para automatizar o cálculo de notas com base numa grelha.
- Incentivar os alunos a adaptar critérios e pesos ao tipo de projeto.
- Discutir a importância de manter registos claros e acessíveis para auditoria.