Script 1 — `0_Utils.R` (documentação completa)

1) Gestão de caminhos, pastas e leitura do dataset

`.get_this_file()`

Finalidade: tenta descobrir o caminho absoluto do ficheiro atualmente em execução (útil em ambientes onde `sys.frame(1)$ofile` existe, por ex. quando se usa `source()`).

- **Input:** nenhum.
- **Output:** string com path normalizado (com `/`) ou `NA_character_` se não conseguir determinar.
- **Notas:** usa `tryCatch()` para ser robusta a contextos onde `ofile` não está disponível (ex.: execução interativa).

`ensure_dir(path)`

Finalidade: garantir que uma pasta existe.

- **Input:** `path` (string).
- **Ação:** se `path` não existir, cria com `recursive = TRUE`.
- **Output:** devolve `path` invisivelmente (`invisible(path)`), para permitir piping/uso silencioso.

Uso típico: criação de diretórios de outputs (capítulos 3–6).

`read_dataset_auto(path)`

Finalidade: ler o CSV com **deteção automática do separador**.

- **Input:** caminho `path`.
- **Pré-condição:** `file.exists(path)` (caso contrário, `stopifnot` falha logo).
- **Lógica:** lê a 1ª linha e decide:
  - se contiver `;` ⇒ usa `read.csv2()`
  - caso contrário ⇒ `read.csv()`
- **Output:** `data.frame` com `stringsAsFactors = FALSE`.

Motivação: compatibilidade com CSVs exportados de diferentes sistemas (pt/Excel costuma usar `;`).

---

2) Métricas de desempenho e pós-processamento de previsão

`rmse(y, yhat)`

Finalidade: calcular RMSE (Root Mean Squared Error), ignorando `NA`.

- **Output:** escalar numérico.

`mae(y, yhat)`

Finalidade: calcular MAE (Mean Absolute Error), ignorando `NA`.

`r2(y, yhat)`

Finalidade: calcular R^2 (coeficiente de determinação) via:

- `SS_res = Σ(y − yhat)²`
- `SS_tot = Σ(y − mean(y))²`
- `R² = 1 − SS_res/SS_tot`

`clip_1_5(x)`

Finalidade: limitar previsões ao intervalo **[1, 5]**, coerente com a escala do `score_review` no enunciado.

- **Uso:** avaliação adicional "com clipping" no Cap. 5 e diagnósticos no Cap. 6.

`within_tolerance(y, yhat, tol = 0.5)`

Finalidade: métrica de interpretação simples: percentagem de previsões com erro absoluto ≤ `tol`.

- Ex.: `tol = 0.5` mede "% de previsões a menos de meio ponto" na escala 1–5.

---

3) Funções auxiliares de IO (texto) e estatísticas simples

`save_lines(lines, path)`

Finalidade: escrever linhas de texto num ficheiro (logs/resumos).

`mode_value(x)`

Finalidade: obter a **moda** ignorando `NA`.

- Se não houver valores não-NA ⇒ devolve `NA`.
- Caso contrário devolve o valor mais frequente (`names(sort(table(...), decreasing=TRUE))[1]`).

Uso típico: imputação por moda para variáveis binárias/categóricas (Cap. 4).

---

4) Conversão e deteção de variáveis binárias 0/1

`to_binary01(x)`

Finalidade: converter automaticamente uma variável para codificação **0/1** quando fizer sentido.

- **Se `logical`:** TRUE→1, FALSE→0 (preserva NA).
- **Se `numeric`:**
  - se já for {0,1} devolve como está,
  - senão não força transformação (devolve original).
- **Se `character`:**
  - normaliza (`trimws`, `tolower`)
  - tenta mapear strings típicas para 1/0
  - calcula `ok_rate` (taxa de valores convertidos com sucesso, considerando NA como "ok")
  - só converte se `ok_rate > 0.7`, caso contrário devolve `x` original (evita conversões erradas).

Porque isto é importante: o dataset tem várias variáveis "sim/não" (amenities e flags), e esta função padroniza a codificação antes de modelação.

`is_binary_01(x)`

Finalidade: verificar se `x` é numérica e se os valores (não-NA) pertencem a {0,1} (no máximo 2 valores distintos).

---

5) Winsorização por IQR (controlo de outliers)

`winsorize_iqr_fit(x, k = 1.5)`

Finalidade: ajustar limites de winsorização com base em IQR:

- `lo = Q1 - k*IQR`
- `hi = Q3 + k*IQR`
- **Input:** vetor numérico
- **Output:** lista `{lo, hi, k}` ou `NULL` se `x` não for numérico.

`winsorize_apply(x, limits)`

Finalidade: aplicar winsorização: valores abaixo de `lo` são elevados para `lo`, acima de `hi` descem para `hi`.

Nota metodológica (anti-leakage): no pipeline correto, os limites devem ser **calculados no treino** e aplicados a val/test com os mesmos parâmetros (é isso que o Cap. 4 faz).

---

6) Reamostragem estratificada: folds (CV) e split 3-way

`make_folds_stratified(y, k = 5, seed = 1)`

Finalidade: criar folds de CV com **estratificação** pela variável `y` (tratada como fator).

- Para cada nível de `y`:
  - obtém índices,
  - embaralha,
  - distribui ciclicamente pelos `k` folds,
  - combina todos os níveis para manter proporções semelhantes por fold.
- **Output:** lista com `k` vetores de índices (ordenados).

Uso: Cap. 5 para CV no treino sem distorcer a distribuição do `score_review` (discreto).

`stratified_split_3way(df, target="score_review", train_frac=0.7, val_frac=0.1, seed=1)`

Finalidade: dividir o dataset em **train/val/test** com estratificação por níveis do target.

- **Pré-condições:**
  - `train_frac > 0`
  - `val_frac >= 0`
  - `train_frac + val_frac < 1` (para existir test)
- **Lógica por nível do target:**
  - embaralha índices desse nível
  - calcula n_train e n_val (com `floor`)
  - garante pelo menos 1 elemento no treino se existir dados nesse nível
  - o resto vai para teste
- **Segurança extra:** remove sobreposições entre conjuntos.

Uso: Cap. 4 para garantir comparabilidade entre conjuntos e reduzir variância da avaliação.

---

7) Preparação de matriz de desenho (model.matrix)

`mmatrix(df, target = "score_review")`

Finalidade: criar uma **matriz de design** (`x`) com `model.matrix` para modelos que exigem input matricial (ex.: `glmnet`).

- Cria fórmula: `target ~ .`
- Constrói `x` e remove intercepto (`[, -1]`)
- Extrai `y = df[[target]]`
- **Output:** lista `{x, y}`.

Uso: Ridge/Lasso no Cap. 5.

---

8) Produção de outputs de diagnóstico (texto e tabelas)

`write_text_snapshot(df, path)`

Finalidade: guardar um "snapshot" textual do dataset:

- dimensões (`dim`)
- nomes das variáveis
- `str(df)`

Uso: Cap. 3 para documentar estrutura original.

`missing_summary_df(df)`

Finalidade: tabela com:

- `variavel`
- `n_missing`
- `pct_missing`

Ordena por `n_missing` desc.

Uso: Cap. 3 para quantificar missingness e justificar imputação.

`coherence_messages(df, target="score_review", target_min=1, target_max=5, numeric_var="logavaliacoes")`

Finalidade: gerar mensagens de coerência (QC):

- range do target
- nº de valores fora do intervalo [1,5]
- resumo de uma variável numérica importante (`logavaliacoes`)
- avisa se variáveis não existirem

Uso: Cap. 3 como verificação rápida de qualidade e coerência com o enunciado (target 1–5).

`unique_counts_df(df)`

Finalidade: número de valores únicos (não-NA) por variável, ordenado crescente.

Uso: identificar colunas quase constantes / potencialmente irrelevantes.

`write_summary_txt(df, path)`

Finalidade: gravar `summary(df)` num txt, para documentação do relatório.

---

9) Funções de visualização (ggplot2)

Estas funções produzem objetos `ggplot` para guardar com `ggsave`.

`plot_target_bar(df, target="score_review")`

Barplot da distribuição do target.

`plot_hist_numeric(df, var, bins=30)`

Histograma de uma variável numérica.

`plot_box_numeric(df, var)`

Boxplot da variável numérica.

`plot_scatter(df, xvar, yvar, alpha=0.5)`

Scatter para análise bivariada.

`plot_binary_props(props_df)`

Barplot horizontal com proporção de 1 em variáveis binárias.

Uso principal: Cap. 3 (EDA).

---

10) Apoio à análise de variáveis binárias e correlações

`coerce_all_to_numeric_safely(df)`

Finalidade: tentar converter todas as colunas para numérico com `suppressWarnings(as.numeric())`.

Uso: permitir identificar binárias 0/1 mesmo quando vieram como texto.

`binary_props_df(df_num, thr_nzv = 0.02)`

Finalidade: sobre colunas binárias 0/1:

- lista `bin_cols`
- `props_df`: proporção de 1 por variável (ordenado desc)
- `nzv_df`: subset das binárias "raras" (min(prop,1-prop) < thr_nzv)

Isto é uma aproximação simples a "near zero variance" para binárias.

`valid_numeric_cols(df_num)`

Finalidade: manter apenas colunas numéricas com variância > 0 e pelo menos 2 valores não-NA.

- remove colunas constantes e “degeneradas”.

`correlation_outputs(df_num_only, target="score_review")`

Finalidade: calcular:

- matriz de correlação Pearson (`cor_mat`) com `pairwise.complete.obs`
- tabela de correlação de cada variável com o target (se o target existir na matriz)

Se houver poucas colunas válidas, devolve `cor_mat=NULL` e uma mensagem explicativa.

Uso: Cap. 3 para a etapa de correlações.

---

11) Diagnóstico de previsões (erros e indicadores)

`add_pred_diagnostics(df_preds, y_true_col="y_true", y_pred_col="y_pred", y_pred_clip_col="y_pred_clipped", tol=0.5)`

Finalidade: adicionar colunas úteis para avaliação:

- `residuo = y_true - y_pred`
- `residuo_abs`
- `residuo_clip = y_true - y_pred_clipped`
- flags booleanas:
  - `acc_0_5` (|erro| ≤ tol)
  - `acc_0_5_clip`

Uso: Cap. 6 ao carregar previsões finais e produzir gráficos e indicadores.

---

12) Guardar plots e gráficos padrão de avaliação

`save_plot(plot_obj, path, width=7, height=5, dpi=150)`

Wrapper de `ggsave` que devolve `path` invisivelmente.

`plot_obs_vs_pred(df, y_pred_col="y_pred", title, subtitle)`

Scatter observado vs previsto com linha y=x (bom para ver viés e dispersão).

`plot_resid_vs_pred(df, y_pred_col="y_pred", resid_col="residuo", title, subtitle)`

Resíduos vs previsto com linha horizontal em 0 (deteção de heterocedasticidade/padrões).

`plot_hist(df, x_col, bins=30, title, subtitle, xlab)`

Histograma genérico (usado para resíduos e |resíduos|).

`plot_rmse_bar_cv(metrics_cv)`

Barplot do RMSE por modelo (ordenado), útil para comunicar escolha do melhor.

Uso: Cap. 6 para relatório final (figuras de avaliação).

---

13) Validação cruzada genérica (motor comum do Cap. 5)

`cv_evaluate(folds, y, predict_fun)`

Finalidade: framework genérico para avaliar um modelo em CV, sem duplicar código.

- **Inputs:**
  - `folds`: lista de índices de validação por fold
  - `y`: vetor target alinhado com os índices
  - `predict_fun(tr_idx, val_idx)`: função que treina no sub-treino e devolve:
    - `list(y_true=..., y_pred=...)`
- **Outputs:** vetor nomeado com médias:
  - `RMSE`, `MAE`, `R2`, `PCT_0_5`

Vantagem: qualquer modelo (lm, rf, gbm, glmnet, etc.) entra no mesmo "contrato" e fica comparável.

`baseline_predict(y_train, n)`

Finalidade: baseline simples: prever sempre a média do treino.

- Serve como referência mínima no benchmarking (Cap. 5).

`save_preds_generic(filename, y_true, y_pred, out_dir)`

Finalidade: guardar previsões padronizadas (CSV) com:

- `y_true`
- `y_pred`
- `y_pred_clipped` (via `clip_1_5`)

Uso: Cap. 5 guarda previsões do teste final do melhor modelo; Cap. 6 lê e acrescenta diagnósticos.