

Aula 2 - Gerenciamento de Dados

Introdução ao aprendizado de máquina - UEMA 2025

Thiago S. F .Silva

2025-12-10

Aquecimento: dado vs informação?

Parte I: Aspectos Gerais

Gerenciamento de dados

- Usabilidade
- Reproducibilidade
- Transparência
- Praticidade

Quem é o seu principal colaborador?

Gerenciamento de dados

- Usabilidade
- Reproducibilidade
- Transparência
- Praticidade

Quem é o seu principal colaborador?

- Você mesmo daqui a 3 meses.

Dados estruturados vs. não-estruturados

- **Dados Estuturados:** planilhas, bancos de dados - dados organizados de maneira tabular.
- **Dados Não-Estruturados:** fotos, gravações, transcrições de entrevistas, páginas da web, etc.

Metadados

Metadado: O dado sobre o dado. Informação adicional / documentação explicativa que permite o uso correto e reproduzível dos dados.

- Especialmente importante para dados não estruturados.
- Ex: Data, hora, local, coletor, instrumento, condições ambientais, unidades, nomes de variáveis, nomenclatura dos arquivos, organização das pastas. etc.

Exemplo: [exemplo_metadado.txt](#)

Exemplo: <https://zenodo.org/records/13887881>

O gerenciamento de dados começa no campo/laboratório

- Protocolo de amostragem
- Caderno de laboratório
- Planilhas de campo

Boas práticas

1. Faça um **estudo piloto**
2. Prepare suas planilhas / caderno **antes** de começar.
3. Anote observações sobre **tudo** o que desviar do protocolo.
Não, você não vai lembrar depois.
4. **Backup, backup, backup:**

Boas práticas

1. Faça um **estudo piloto**
2. Prepare suas planilhas / caderno **antes** de começar.
3. Anote observações sobre **tudo** o que desviar do protocolo.
Não, você não vai lembrar depois.
4. **Backup, backup, backup:**
 - Dados analógicos:
 - Ao final de cada dia/sessão, **fotografe** seu caderno / planilha de campo.
 - Se houver acesso à internet, já sincronize com a nuvem.

Boas práticas

1. Faça um **estudo piloto**
2. Prepare suas planilhas / caderno **antes** de começar.
3. Anote observações sobre **tudo** o que desviar do protocolo.
Não, você não vai lembrar depois.
4. **Backup, backup, backup:**
 - Dados digitais:
 - Baixe com a maior frequencia possível.
 - Mantenha pelo menos duas cópias, em dois dispositivos diferentes.
 - Se houver acesso à internet, já sincronize com a nuvem.

Boas práticas

5. Tente digitar os dados o mais rápido possível - enquanto os detalhes do trabalho ainda estão frescos na sua cabeça.

Caderno/planilha vs. coletar dados direto no PC/tablet/telefone?

Parte II: Aspectos Práticos

Digitando dados

Qual software você usa pra digitar seus dados?

- Excel: cuidado com a formatação dos dados

Digitando dados

Qual software você usa pra digitar seus dados?

- Excel: cuidado com a formatação dos dados
- Datas e horas: formato ISO8601

Digitando dados

Qual software você usa pra digitar seus dados?

- Excel: cuidado com a formatação dos dados
- Datas e horas: formato ISO8601
- Separadores decimais e de milhar: vírgula ou ponto?

Digitando dados

Qual software você usa pra digitar seus dados?

- Excel: cuidado com a formatação dos dados
- Datas e horas: formato ISO8601
- Separadores decimais e de milhar: vírgula ou ponto?
- Indicadores de dado faltante (*missing data*)

Tipos de dados (*data types*):

Tipo de dado	nome R	nome Excel
Float	numeric	number
Integer	integer	number
Byte	numeric	number
Boolean	logical	-
String	character	text
Datetime	POSIX,	date / date, time

Demonstração em R?

Prática

Preparando uma planilha Excel para receber dados

Prática

Identifique e corrija os problemas no arquivo
[planilha_do_capeta.xlsx](#)

Tidy data

Boa parte do trabalho de análise de dados consiste na organização / limpeza de dados.

Tidy Data: <https://www.jstatsoft.org/article/view/v059i10>

- Cada variável é uma coluna
- Cada observação é uma linha

Exemplo

Table 2: Tabela 1. Número de fatalidades em relação ao número de animais infectados, para três diferentes fazendas entre 2023-2025.

Local	2023	2024	2025
Fazenda A	12/50	13/45	9/52
Fazenda B	8/35	10/40	7/38
Fazenda C	8/42	5/37	3/51

Quantas variáveis tem essa tabela?

Exemplo

- Local/Fazenda

Exemplo

- Local/Fazenda
- Ano

Exemplo

- Local/Fazenda
- Ano
- Número de animais infectados

Exemplo

- Local/Fazenda
- Ano
- Número de animais infectados
- Número de fatalidades

Exemplo

- Local/Fazenda
- Ano
- Número de animais infectados
- Número de fatalidades

Exemplo

- Local/Fazenda
- Ano
- Número de animais infectados
- Número de fatalidades

Como organizar essa tabela de maneira *tidy*?

Exemplo

Local	ano	infetado	fatalidade
Fazenda A	2023	50	12
Fazenda A	2024	45	13
Fazenda A	2025	52	9
Fazenda B	2023	35	8
Fazenda B	2024	40	10
Fazenda B	2025	38	7
Fazenda C	2023	42	8
Fazenda C	2024	37	5
Fazenda C	2025	51	3

Encerramento Parte II - Discussão em pares

- Que dados você está coletado / planeja coletar?
- Como esses dados estão sendo organizados e armazenados?
- Como você acha que pode melhorar essa organização?
- Se o seu computador desaparecer agora, o que aconteceria com o seu trabalho?

Parte III - Gerenciando dados no R

Prática: Configurando o RStudio para trabalhar

Projetos no RStudio

Uma parte **essencial** da análise de dados é a estruturação de um *projeto*:

- Conjunto de pastas que organiza o trabalho
- Ajuda na auto-documentação do projeto
- Diferentes análises - diferentes projetos

Organização mínima de um projeto:

```
meu_projeto
|_____ src
|_____ data
|       |_____ raw
|       |_____ processed
|_____ outputs
|_____ docs
```

Organização mínima de um projeto:

```
meu_projeto
|_____ scripts
|_____ dados
|         |_____ brutos
|         |_____ processados
|_____ resultados
|_____ notas
```

Boas práticas de nomenclatura de pastas/arquivos

- sem letras maiúsculas
- sem acentos
- sem espaços
- usar somente – e _ como caracteres extra

Estilos de nomenclatura de pastas/arquivos/variáveis

- **Snake case:** meu_lindo_arquivo.txt
- **Kebab case:** meu-lindo-arquivo.txt
- **Camel case:** meuLindoArquivo.txt
- **Pascal case:** MeuLindoArquivo.txt

Pacotes do R às vezes usam o estilo nome.variavel para nomes de variáveis. Isso não é aconselhado, pois causa confusão com outras linguagens que usam o .. como um operador.

Escolha sua favorita - mas seja **consistente!**

Exemplo: criando um projeto no RStudio

Vamos criar um novo projeto no RStudio para a aula de hoje.

- Criar o projeto
- Criar a estrutura de pastas
- Baixar e organizar os arquivos de dados da Aula 1

Armazenando Dados

Qual o melhor formato para armazenamento de dados estruturados?

- Excel (xls, xlsx): incluem informações sobre o tipo de dado, mas não é um formato universal.

Armazenando Dados

Qual o melhor formato para armazenamento de dados estruturados?

- Excel (xls, xlsx): incluem informações sobre o tipo de dado, mas não é um formato universal.
- Texto (csv, tsv): extremamente simples, universalmente acessíveis, mas não guardam informação sobre o tipo de dado.

Armazenando Dados

Qual o melhor formato para armazenamento de dados estruturados?

- Excel (xls, xlsx): incluem informações sobre o tipo de dado, mas não é um formato universal.
- Texto (csv, tsv): extremamente simples, universalmente acessíveis, mas não guardam informação sobre o tipo de dado.
- **Parquet**: formato aberto desenvolvido especialmente para armazenamento eficiente de dados. Recemente se tornou padrão em data science - mas já está sendo substituído.

Armazenando Dados

Arquivos RDS e RData

- Formato interno do R.
- RData pode armazenar múltiplos objetos, RDS somente um objeto.
- Formatos que só são entendidos pelo R.
- OK para salvar dados intermediários durante o processamento.

Armazenando Dados

E dados não-estruturados?

- Priorize formatos abertos

Armazenando Dados

E dados não-estruturados?

- Priorize formatos abertos
- Use texto simples para os metadados

Armazenando Dados

E dados não-estruturados?

- Priorize formatos abertos
- Use texto simples para os metadados
- Dados de texto não estruturados: JSON

Lendo dados no R

Exemplo prático: lendo diferentes tipos de arquivos para o R.

Passo 1: preparando o arquivo `plot_florestal.xlsx`

Excel

Pacote: `readxl()` (parte do `tidyverse`)

Funções: `read_xls()` e `read_xlsx()`

Help: https://readxl.tidyverse.org/reference/read_excel.html

Código

```
library(readxl)
plot_florestal <- read_xlsx('../data/plot_florestal.xlsx')
head(plot_florestal)
str(plot_florestal)
glimpse(plot_florestal)
```

Resultado

```
# A tibble: 6 x 16
  Espécie   Plot Tag Ramo Data `^Ano Recrutamento` V
  <chr>     <chr> <dbl> <chr> <dttm> <chr> <chr>
1 Eschweile~ 2b     247 <NA> 2022-10-01 00:00:00 NA
2 Pouteria ~ 2b     248 <NA> 2022-10-02 00:00:00 NA
3 Neea        2b     249 <NA> 2022-10-03 00:00:00 NA
4 Leonia gl~ 2b     250 <NA> 2022-10-04 00:00:00 NA
5 Pouteria ~ 2b     251 <NA> 2022-10-05 00:00:00 NA
6 Iryanther~ 2b     252 <NA> 2022-10-06 00:00:00 NA
# i 9 more variables: Família <chr>, Gênero <chr>, Mortalidade_2022 <dbl>
#   DAP_2017 <chr>, DAP_2022 <chr>, ALTURA_2017 <chr>, ALTURA_2022 <dbl>
#   pom_2017 <chr>, pom_2022 <chr>
```

Rows: 120

Columns: 16

```

$ Espécie           <chr> "Eschweilera albiflora", "Pouteria elegans",
$ Plot              <chr> "2b", "2b", "2b", "2b", "2b", "2b", "2b", "2b"
$ Tag               <dbl> 247, 248, 249, 250, 251, 252, 253, 254, 255,
$ Ramo              <chr> NA, NA,

```

**Desafio: como ler a segunda planilha
do arquivo excel?**

Texto Simples (CSV, TSV, etc)

- **R básico:** `read.csv()`,
`read.csv2()`,`read.delim()`,`read.table()`
- **Tidyverse** (pacote `readr`): `read_csv()`, `read_csv2()`,
`read_tsv()`, `read_delim()`

Código

```
library(readr)
turb_dados <- read_csv('../data/turbidez.csv')
head(turb_dados)
glimpse(turb_dados)
```

Resultado

```
# A tibble: 6 x 6
  id_amostra local data      temp_c ph turbidez_ntu
  <dbl> <chr> <chr>      <chr>  <chr> <chr>
1     1063 A02  2024-02-10 19.41   7    11.1
2     1083 B01  17/02/2024 24.59   7    5
3     1068 B01  15/02/2024 20.26   6.71  15.4
4     1065 A01  2024-01-21 29.09   7.19  4
5     1068 B01  23/01/2024 21.93  -999  11.1
6     1011 A02  25/01/2024 25.41   6.97  10.7
```

Rows: 120

Columns: 6

```
$ id_amostra <dbl> 1063, 1083, 1068, 1065, 1068, 1011, 10
$ local       <chr> "A02", "B01", "B01", "A01", "B01", "A01", "B01"
$ data        <chr> "2024-02-10", "17/02/2024", "15/02/2024", "2024-01-21", "23/01/2024", "25/01/2024", "2024-01-25"
$ temp_c      <chr> "19.41", "24.59", "20.26", "29.09", "21.93", "25.41", "25.07"
$ ph          <dbl> 7, 7, 6.71, 7.19, -999, 6.97, 7.01
$ turbidez_ntu <dbl> 11.1, 5, 15.4, 4, 11.1, 10.7, 10.33
```

Vantages e desvantagens

Excel: - Inclui informação sobre o *data type* de cada coluna -
Tamanho de arquivo é maior - Não é legível sem ter o Excel
instalado - Quem garante que daqui a 10 anos esse formato ainda
vai existir?

Texto: - Não inclui informação sobre os *data types* - Arquivos
menores - Qualquer software é capaz de abrir/ler

Um detalhe sobre texto

Qualquer arquivo digital é armazenado internamente como números (bits).

Como representar texto como números?

Um detalhe sobre texto

Qualquer arquivo digital é armazenado internamente como números (bits).

Como representar texto como números?

Encoding: padrão que define como cada caractere é representado por um número

ASCII vs UTF-8 vs outros

Os primeiros computadores tinham muito pouca memória/disco.

ASCII: encoding mais básico, só inclui 128 caracteres (7-bit encoding)

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	Space		64	40	100			96	60	140		
1	1	001	Start of Header	33	21	041		!	65	41	101			97	61	141		
2	2	002	Start of Text	34	22	042		“	66	42	102			98	62	142		
3	3	003	End of Text	35	23	043		#	67	43	103			99	63	143		
4	4	004	End of Transmission	36	24	044		\$	68	44	104			100	64	144		
5	5	005	Enquiry	37	25	045		%	69	45	105			101	65	145		
6	6	006	Acknowledgment	38	26	046		&	70	46	106			102	66	146		
7	7	007	Bell	39	27	047		‘	71	47	107			103	67	147		
8	8	010	Backspace	40	28	050		(72	48	108			104	68	150		
9	9	011	Horizontal Tab	41	29	051)	73	49	109			105	69	151		
10	A	012	Line feed	42	2A	052		*	74	4A	112			106	6A	152		
11	B	013	Vertical Tab	43	2B	053		+	75	4B	113			107	6B	153		
12	C	014	Form feed	44	2C	054		,	76	4C	114			108	6C	154		
13	D	015	Carriage return	45	2D	055		-	77	4D	115			109	6D	155		
14	E	016	Shift Out	46	2E	056		:	78	4E	116			110	6E	156		
15	F	017	Shift In	47	2F	057		/	79	4F	117			111	6F	157		
16	10	020	Data Link Escape	48	30	060		0	80	50	120			112	70	160		
17	II	021	Device Control 1	49	31	061		1	81	51	121			113	71	161		
18	I2	022	Device Control 2	50	32	062		2	82	52	122			114	72	162		
19	I3	023	Device Control 3	51	33	063		3	83	53	123			115	73	163		
20	I4	024	Device Control 4	52	34	064		4	84	54	124			116	74	164		
21	I5	025	Negative Ack.	53	35	065		5	85	55	125			117	75	165		
22	I6	026	Synchronous Idle	54	36	066		6	86	56	126			118	76	166		
23	I7	027	End of Trans. Block	55	37	067		7	87	57	127			119	77	167		
24	I8	030	Cancel	56	38	070		8	88	58	130			120	78	170		
25	I9	031	End of Medium	57	39	071		9	89	59	131			121	79	171		
26	I A	032	Substitute	58	3A	072		:	90	5A	132			122	7A	172		
27	I B	033	Escape	59	3B	073		:	91	5B	133			123	7B	173		
28	I C	034	File Separator	60	3C	074		<	92	5C	134			124	7C	174		
29	I D	035	Group Separator	61	3D	075		=	93	5D	135			125	7D	175		
30	I E	036	Record Separator	62	3E	076		>	94	5E	136			126	7E	176		
31	I F	037	Unit Separator	63	3F	077		?	95	5F	137			127	7F	177		
																	Del	

ASCII vs UTF-8 vs outros

UTF-8: usa um método diferente (variable length encoding) e por isso é capaz de armazenar até 1.1 milhão de caracteres. - Multiplas línguas - Símbolos - Emojis - Etc...

Existem muitos outros...

ASCII vs UTF-8 vs outros

Se você tenta ler um arquivo criado com um certo encoding, especificado um encoding diferente, o resultado pode ser estranho:

```
# Gravando um arquivo em UTF-8
con <- file('exemplo_utf8.txt', open = 'w', encoding = 'UTF-8')
writeLines('Maçã, café, romã, pêra', con)
close(con)
```

ASCII vs UTF-8 vs outros

```
# Lendo como UTF-8
con <- file('exemplo_utf8.txt', open = 'r', encoding = 'UTF-8')
content <- readLines(con, warn = FALSE)
close(con)
content
```

[1] "Maçã, café, romã, pêra"

ASCII vs UTF-8 vs outros

```
# Lendo como ASCII
con <- file('exemplo_utf8.txt', open = 'r', encoding = 'ASCI
content <- readLines(con, warn = FALSE)
close(con)
content
```

[1] "MaC'C#, cafC), romC#, pC*ra"

Exemplo de arquivo Parquet

```
install.packages('arrow')
library(arrow)

dados <- read_parquet('../data/flights-1m.parquet')
head(dados)
glimpse(dados)

write_csv(dados, '../junk/flights-1m.csv')

saveRDS(dados, '../junk/flights-1m.rds')

# Usando o rad.csv do R base
testcsv <- read.csv('../junk/flights-1m.csv')

glimpse(testcsv)
```

Parte III - Organizando dados

Problemas comuns

- Erros de importação
- Dados faltantes
- Erros de digitação
- Nomes de variável impróprios
- Tipos de dado incorretos
- Tabela organizada de maneira ‘não-*tidy*’

Pré-processamento dos dados

O pré-processamento inclui todos os ajustes necessários **antes** de se iniciar a análise.

Boa Prática: *Nunca modifique seu arquivo original.*

- Todas as correções devem ser feitas via código
- Isso torna a análise reproduzível e transparente, e serve como auto-documentação.

Pré-processamento

Os pacotes do tidyverse mais importantes para preparo dos dados são:

- `tidyr`: <https://tidyr.tidyverse.org/>
- `dplyr`: <https://dplyr.tidyverse.org/>
- `lubridate`:<https://lubridate.tidyverse.org/>
- `forcats`: <https://forcats.tidyverse.org/>
- `stringr`:<https://stringr.tidyverse.org/>

Exemplo - `tidyR`

A principal função utilizada é `pivot_longer`, para desmembrar variáveis.

Tabela formato *largo(wide)*: geralmente quebram a expectativa de uma variável por coluna

Tabela formato *longo(long)*: formato esperado com uma variável por coluna.

Exemplo - `tidyverse`

Dataset `relig_income`: pesquisa sobre religião e renda.

```
library(tidyverse)
head(relig_income)
glimpse(relig_income)
```

Quais as variáveis desse dataset?

A organização está *tidy*?

Exemplo 1 - tidyverse

```
library(tidyverse)
relig_tidy <- relig_income %>%
  pivot_longer(cols = !religion,
               names_to = 'renda',
               values_to = 'freq')

head(relig_tidy)
glimpse(relig_tidy)
```

Exemplo 1 - tidyverse

```
relig_tidy <- relig_income %>%
  pivot_longer(cols = !religion,
               names_to = 'renda',
               names_transform = as.factor,
               values_to = 'freq')

head(relig_tidy)
glimpse(relig_tidy)
```

Exemplo 2 - tidyverse

Dataset billboard: ranking das musicas no topo da lista billboard ao longo das semanas no ano de 2000.

```
head(billboard)  
glimpse(billboard)
```

Quantas variáveis nesse dataset?

Exemplo 2 - tidyverse

```
bill_tidy <- billboard %>%
  pivot_longer(cols = starts_with('wk'),
               names_to = 'week',
               values_to = 'rank',
               values_drop_na = TRUE)

summary(bill_tidy)
head(bill_tidy)
levels(bill_tidy)
```

Exemplo 3 - tidyverse

Dataset `who`: dados sobre pacientes de tuberculose.

```
head(who)  
glimpse(who)
```

Quantas variáveis?

Exemplo 3 - tidyverse

country: país

iso2 e *iso3*: siglas padronizadas para cada país

year: ano da medição

new_/: indica que são novos casos.

sp/rel/ep: método de diagnóstico do caso

m/f: gênero do paciente

014/1524/2535/3544/4554/65: intervalos de idade

Exemplo 3 - tidyverse

```
who_tidy <- who %>%  
  pivot_longer(cols = starts_with('new'),  
               names_to = 'temp_names',  
               values_to = 'freq',  
               values_drop_na = TRUE)  
  
dim(who_tidy)  
head(who_tidy)  
levels()
```

dplyr - o canivete suíço para manipular dados

Principais funções:

- `mutate()`: transformação de variáveis
- `select()`: seleção de variáveis (colunas)
- `filter()`: seleção de observações (linhas)
- `arrange()`: ordena as linhas com base em variáveis
- `group_by()`: agrupa observações com base em variáveis
- `summarise()`: calcula sumários de grupos de observações

Exemplo 1 - dplyr

```
glimpse(iris)
```

Exemplos para `mutate()`,`select()`, `filter()`, `arrange()`,
`group_by()`,`summarise()`.

stringr - maniulação de *strings*

<https://stringr.tidyverse.org/index.html>

forcats - manipulação de fatores

<https://forcats.tidyverse.org/reference/index.html>

Combinando stringr, dplyr, tidyverse e forcats

```
who_tidy <- who %>%  
  pivot_longer(cols = starts_with('new'),  
               names_to = 'temp_names',  
               values_to = 'freq',  
               values_drop_na = TRUE) %>%  
  mutate(prefix = str_sub(temp_names, 1, 3),  
         diag = str_sub(temp_names, 4, 6),  
         gender = str_sub(temp_names, 8, 8),  
         age_range = str_sub(temp_names, 9)) %>%  
  mutate_if(is.character, as_factor) %>%  
  mutate(diag = fct_recode(  
    diag,  
    'sp'='_sp', 'sn'='_sn', 'ep'='_ep'  
  )) %>%  
  mutate(age_range = fct_recode(
```

Importante

NUNCA substitua os dados originais!

Os dados ‘limpos’ devem ser um produto automatizado dos dados originais, salvos em um arquivo diferente

Isso garante a integridade e transparência.

Prática Interativa

Organizando os datasets anteriores:

- tabela_do_capeta.xlsx
- plot_florestal.xlsx
- turbidez.csv