

# DATA MINING & MACHINE LEARNING (I)

Thiago Marzagão



**Centro Universitário**

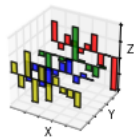
# mineração de dados



Excel não permite reprodutibilidade e facilita acidentes



**pandas**  
 $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



## str como int/float

```
import pandas as pd
data = pd.read_csv('municipios.csv')
data
data.columns
data['codigo']
```

- ▶ id ('codigo') deveria ser str (ou object), não int
- ▶ do contrário várias coisas podem dar errado:
- ▶ ...zeros à esquerda podem desaparecer
- ▶ ...id pode ser arredondado
- ▶ ...id pode ser truncado (ex.: mais de 15 dígitos no Excel)

## int/float como str

```
data['rural']  
data['rural'] + 5  
data['rural'] * 5  
"macarena" * 5
```

- ▶ 'rural' deveria ser float, não str ou object
- ▶ do contrário várias coisas podem dar errado:
- ▶ ...operações matemáticas vão dar erro
- ▶ ...o erro pode ser silencioso!

## solução (sort of)

```
data = pd.read_csv('municipios.csv',  
                  dtype = {'codigo': str,  
                          'rural': float})
```

- ▶ O parâmetro 'dtype' permite especificar o tipo de cada coluna

## solução (agora sim)

```
data = pd.read_csv('municipios.csv',  
                  dtype = {'codigo': str,  
                          'rural': float},  
                  decimal = ',')
```

- ▶ O parâmetro 'decimal' permite especificar o separador de decimais



## separador de campos != vírgula

- ▶ Nem sempre a vírgula é o separador de campo.
- ▶ O Excel em português, por exemplo, exporta p/ CSV separando c/ ponto-e-vírgula.
- ▶ Nesses casos é preciso informar explicitamente o separador:

```
data = pd.read_csv('nomedoarquivo.csv', sep = ';')
```

## CSV sem nomes de colunas

- ▶ Quando o CSV não tem nomes de colunas:

```
data = pd.read_csv('nomedoarquivo.csv', header = None)
```

## pulando linhas

```
data = pd.read_csv('municipios.csv', skiprows = 5)
```

- ▶ Mas cuidado: o header também é pulado.

## pulando columnas

```
data = pd.read_csv('municipios.csv',  
                  usecols = ['latitude',  
                             'longitud'])
```

## missing data

- ▶ vários “tipos” de missing data:
- ▶ None (em Python)
- ▶ NULL (SQL, outras linguagens)
- ▶ (errado! não façam)
- ▶ - (errado! não façam)
- ▶ 0 (errado! não façam)
- ▶ 01/01/1900 (errado! não façam)
- ▶ -9, -999, -1, etc (errado, errado, errado! não façam)

## missing data

- ▶ missing data propriamente identificado:

$$(50.5 + 42.1 + 30.4 + 60.7 + \text{None} + 45.8) / 6$$

- ▶ dá erro; e isso é bom!
- ▶ você *não* quer que essa média seja calculada
- ▶ missing data impropriamente identificado:

$$(50.5 + 42.1 + 30.4 + 60.7 + -999 + 45.8) / 6$$

$$(50.5 + 42.1 + 30.4 + 60.7 + 0 + 45.8) / 6$$

$$(50.5 + 42.1 + 30.4 + 60.7 + -1 + 45.8) / 6$$

- ▶ *não* dá erro; e isso é ruim!
- ▶ você *não* quer que essa média seja calculada

## missing data

- ▶ p/ encontrar possíveis problemas:

```
data['rural'].describe()
```

# duplicidades

- ▶ origem: por exemplo, diferentes grafias de um mesmo nome
- ▶ ...Mônica e Monica



## unidade inconsistente

- ▶ caso mais comum: confusão entre preço unitário e preço total

## datas em formatos inconsistentes

- ▶ caso mais comum: confusão entre DD/MM/AAAA e MM/DD/AAAA
- ▶ 3/4/2005? 3 de abril de 2005 ou 4 março de 2005?
- ▶ formato correto: AAAA-MM-DD (ISO 8601)

- ▶ dicionário de dados = arquivo contendo descrição detalhada de cada variável e origem dos dados