

Introdução à Ciência de Dados com Python

Preparação de Dados

Residência em TIC 09
Programa Prioritário da Lei de Informática



Agenda

Conceitos gerais sobre preparação de dados

Limpeza dos dados

Transformações

Dados organizados (tidy data)

Conceitos sobre preparação de dados

O que é preparação de dados?

Conjunto de atividades para aquisição e garantia de qualidade dos dados para análise subsequente

Etapa inicial que precede a análise e a modelagem

Interface com engenharia de dados para coleta de dados

Processo não linear

Vários autores utilizam nomes diferentes

Confusão de nomenclatura

Etapas de preparação de dados

Limpeza

Tratamento de dados ausentes, anomalias, erros, etc

Transformação

Formatação, agregação, filtragem, normalização, etc

Organização (tidy data)

Criação de um repositório válido e consistente a partir das etapas anteriores

Nosso foco

Neste curso, nosso foco será nas etapas de limpeza, transformação e organização

Estas duas etapas demandam maior tempo do analista de dados

Em alguns cenários, o analista de dados não realiza coleta nem integração

Limpeza de Dados

Dados Ausentes

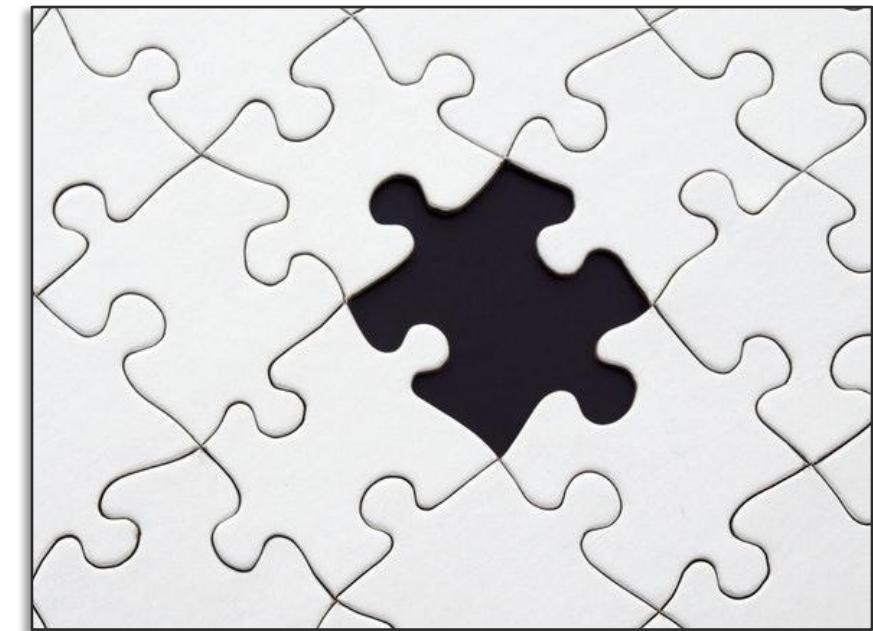
Por que os dados desaparecem?

Não há o que registrar (ex. formulários)

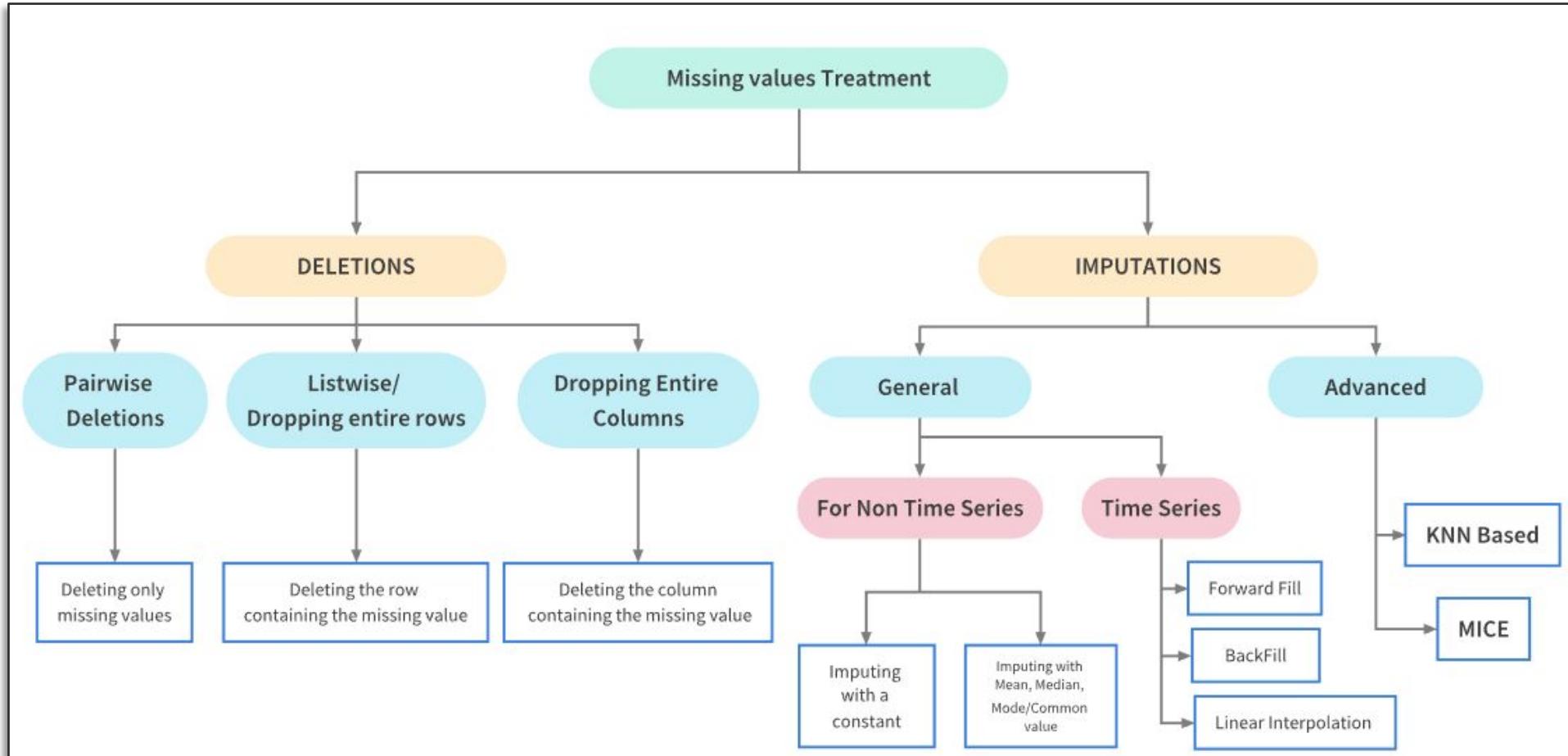
Problemas na coleta por erro humano,
sensores, etc

Remoção accidental

Resultado de join ou merge (left ou right)



Tratamento de dados ausentes



Diagnóstico de dados ausentes

Qual a porcentagem de dados ausentes em cada atributo?

Ausência de um atributo é correlacionado com ausência em algum outro?

Exemplo: dataset titanic

```
titanic_df = pd.read_csv('titanic.csv')
```

```
titanic_df.sample(n=10)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
579	580	1	3	Jussila, Mr. Eiriik	male	32.0	0	0	STON/O 2. 3101286	7.9250	NaN	S
777	778	1	3	Emanuel, Miss. Virginia Ethel	female	5.0	0	0	364516	12.4750	NaN	S
437	438	1	2	Richards, Mrs. Sidney (Emily Hocking)	female	24.0	2	3	29106	18.7500	NaN	S
405	406	0	2	Gale, Mr. Shadrach	male	34.0	1	0	28664	21.0000	NaN	S
557	558	0	1	Robbins, Mr. Victor	male	NaN	0	0	PC 17757	227.5250	NaN	C
774	775	1	2	Hocking, Mrs. Elizabeth (Eliza Needs)	female	54.0	1	3	29105	23.0000	NaN	S
205	206	0	3	Strom, Miss. Telma Matilda	female	2.0	0	1	347054	10.4625	G6	S
868	869	0	3	van Melkebeke, Mr. Philemon	male	NaN	0	0	345777	9.5000	NaN	S
550	551	1	1	Thayer, Mr. John Borland Jr	male	17.0	0	2	17421	110.8833	C70	C
419	420	0	3	Van Impe, Miss. Catharina	female	10.0	0	2	345773	24.1500	NaN	S

```
titanic_df = pd.read_csv('titanic.csv')
```

```
titanic_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   PassengerId 891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object  
 4   Sex          891 non-null    object  
 5   Age          714 non-null    float64 
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object  
 9   Fare          891 non-null    float64 
 10  Cabin         204 non-null    object  
 11  Embarked     889 non-null    object  
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

isnull()

```
titanic_df = pd.read_csv('titanic.csv')
```

```
titanic_df.info()
```

```
titanic_df.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age             177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked        2
dtype: int64
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
--- 
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object  
 4   Sex          891 non-null    object  
 5   Age          714 non-null    float64 
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object  
 9   Fare         891 non-null    float64 
 10  Cabin        204 non-null    object  
 11  Embarked     889 non-null    object  
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

Percentagem

```
titanic_df = pd.read_csv('titanic.csv')
```

```
titanic_df.info()
```

```
titanic_df.isnull().sum()/len(titanic_df)*100
```

```
PassengerId      0.000000
Survived         0.000000
Pclass            0.000000
Name              0.000000
Sex               0.000000
Age              19.865320
SibSp             0.000000
Parch             0.000000
Ticket            0.000000
Fare              0.000000
Cabin            77.104377
Embarked          0.224467
dtype: float64
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype  
--- 
 0   PassengerId   891 non-null    int64  
 1   Survived      891 non-null    int64  
 2   Pclass         891 non-null    int64  
 3   Name           891 non-null    object  
 4   Sex            891 non-null    object  
 5   Age            714 non-null    float64 
 6   SibSp          891 non-null    int64  
 7   Parch          891 non-null    int64  
 8   Ticket         891 non-null    object  
 9   Fare           891 non-null    float64 
 10  Cabin          204 non-null    object  
 11  Embarked       889 non-null    object  
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

Função para dados ausentes

```
def missing_values_table(df):
    mis_val = df.isnull().sum()
    mis_val_percent = 100 * df.isnull().sum() / len(df)
    mis_val_df = pd.concat([mis_val, mis_val_percent], axis=1)
    mis_val_df.rename(columns = {0 : 'Ausentes',
                                 1 : 'Percentual'},
                      inplace=True)
    mis_val_df = mis_val_df[mis_val_df.iloc[:,1] != 0]
    mis_val_df = mis_val_df.sort_values('Percentual',
                                       ascending=False).round(1)
    return mis_val_df
```

```
missing_values_table(titanic_df)
```

	Ausentes	Percentual
Cabin	687	77.1
Age	177	19.9
Embarked	2	0.2

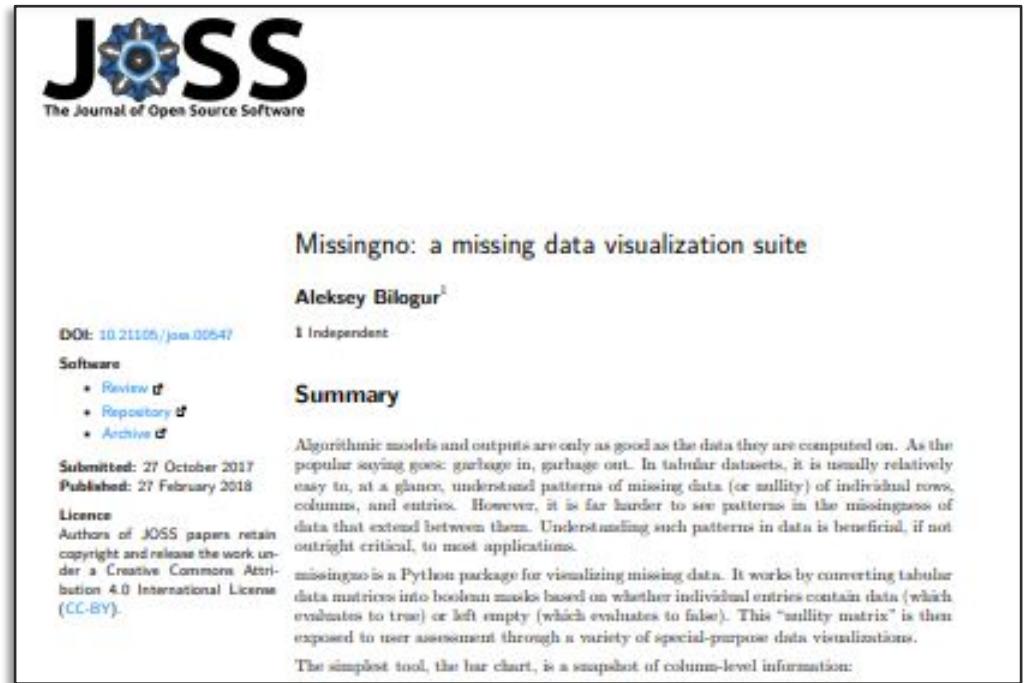
Biblioteca missingno

Provê visualizações que ajudam a identificar ocorrência de dados ausentes em um dataframe

Barplots

Matriz plot

Heatmaps



The screenshot shows a journal article from the Journal of Open Source Software (JOSS). The title is "Missingno: a missing data visualization suite" by Aleksey Bilogur. It includes links for DOI, software reviews, and repositories. The abstract discusses the challenges of identifying missing data patterns in tabular datasets and introduces missingno as a Python package for visualizing missing data through various plots like bar charts and heatmaps.

JOSS
The Journal of Open Source Software

DOI: [10.21105/joss.00547](https://doi.org/10.21105/joss.00547)

Aleksey Bilogur¹
¹ Independent

Software

- Review [d](#)
- Repository [d](#)
- Archive [d](#)

Submitted: 27 October 2017
Published: 27 February 2018

Licence
Authors of JOSS papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC-BY](#)).

Summary

Algorithmic models and outputs are only as good as the data they are computed on. As the popular saying goes, garbage in, garbage out. In tabular datasets, it is usually relatively easy to, at a glance, understand patterns of missing data (or nullity) of individual rows, columns, and entries. However, it is far harder to see patterns in the missingness of data that extend between them. Understanding such patterns in data is beneficial, if not outright critical, to most applications.

missingno is a Python package for visualizing missing data. It works by converting tabular data matrices into boolean masks based on whether individual entries contain data (which evaluates to true) or left empty (which evaluates to false). This “nullity matrix” is then exposed to user assessment through a variety of special-purpose data visualizations.

The simplest tool, the bar chart, is a snapshot of column-level information:

<https://www.theoj.org/joss-papers/joss.00547/10.21105.joss.00547.pdf>

<https://github.com/ResidentMario/missingno>

Biblioteca missingno

Instalação

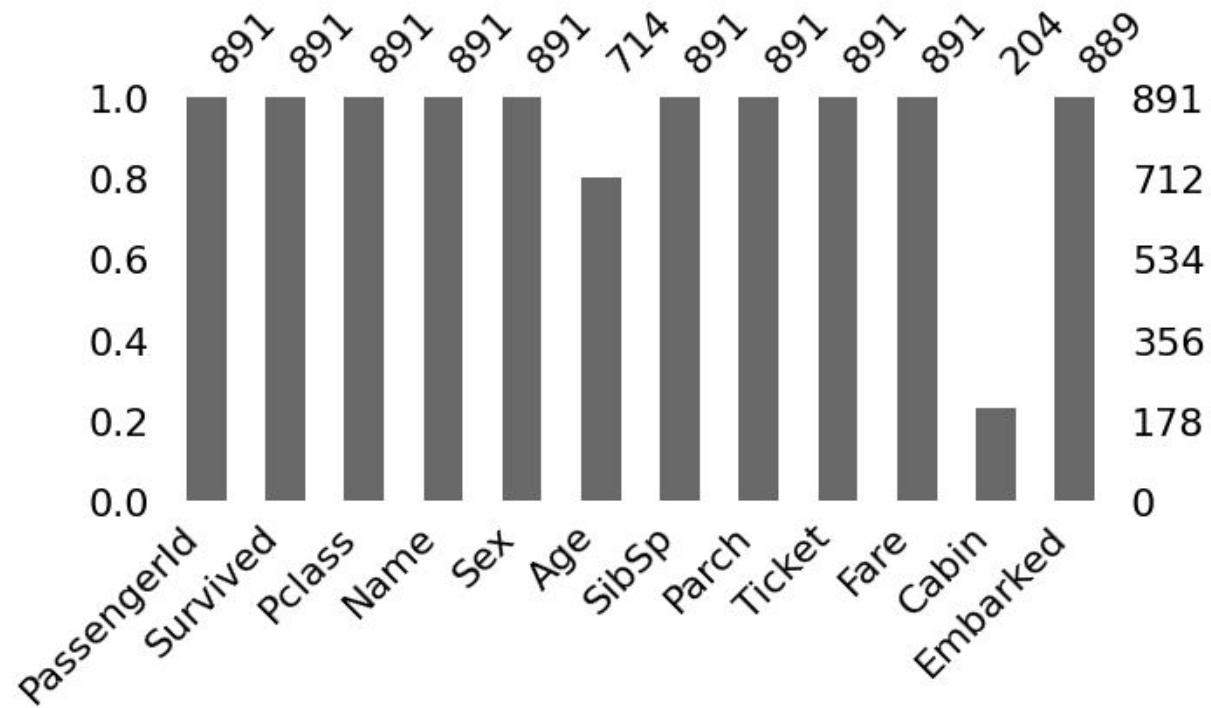
```
pip install missingno
```

Uso

```
import missingno as msno
```

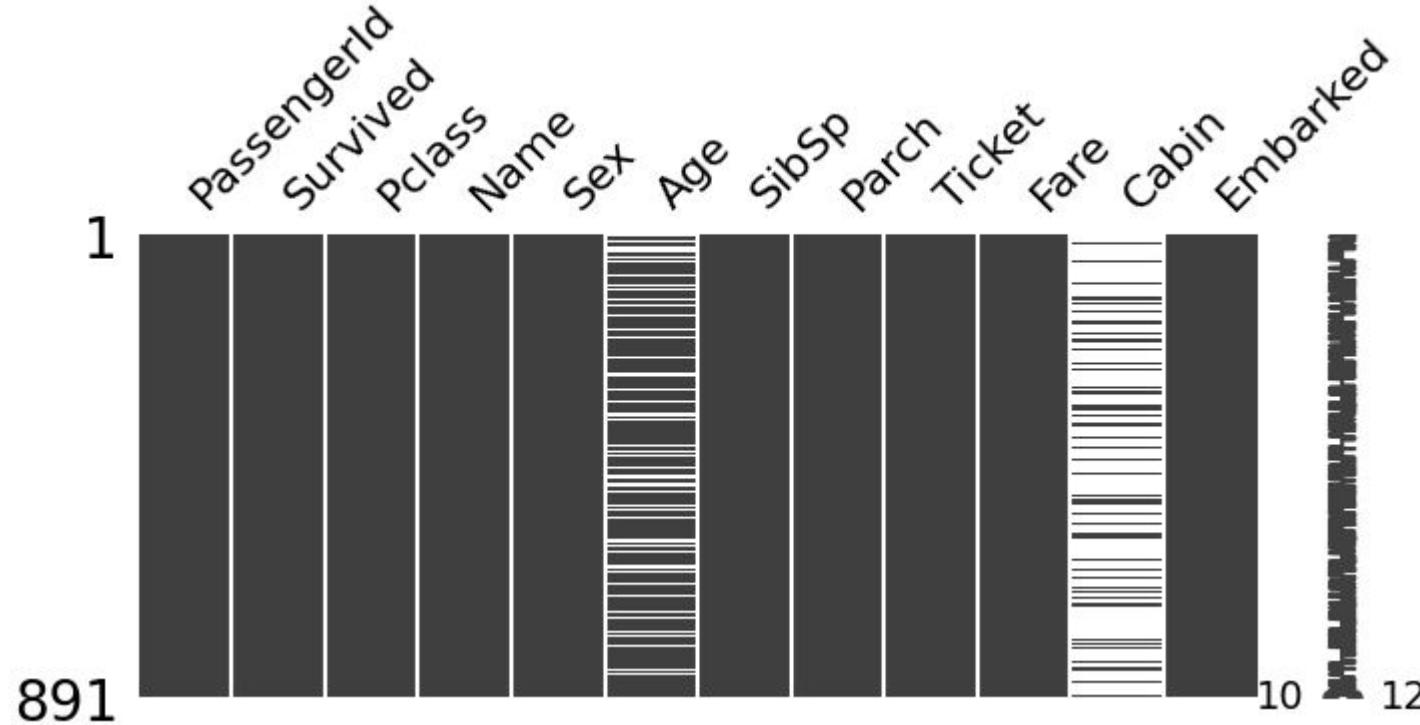
Missingno: barplot

```
msno.bar(titanic_df,figsize=(7,3))
```



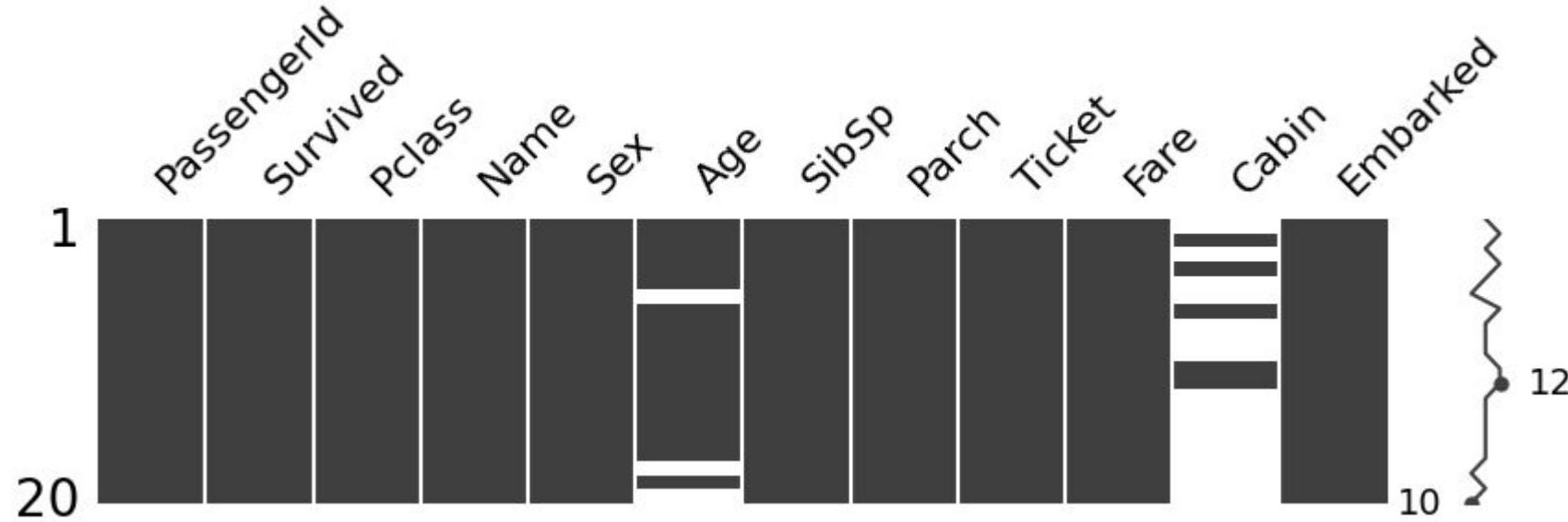
Missingno: matrix

```
msno.matrix(titanic_df, figsize=(8,3))
```



Missingno: detalhe (20 tuplas)

```
msno.matrix(titanic_df.head(20), figsize=(10, 2))
```



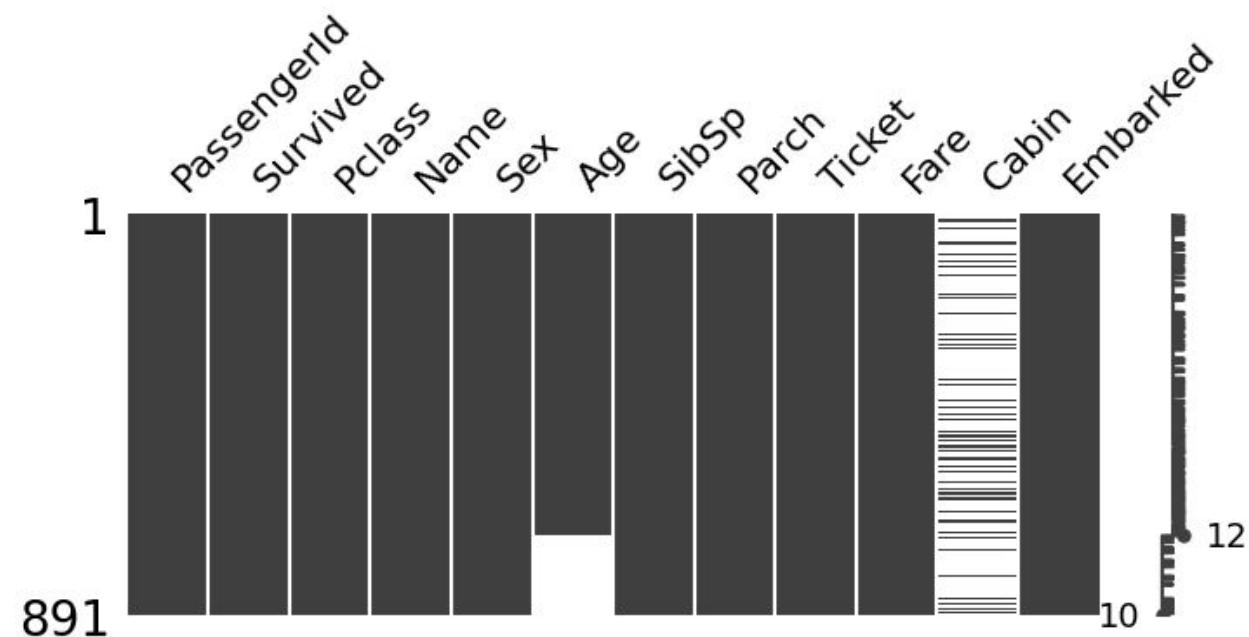
Missingno: matrix com dados ordenados

Há correlações entre as ausências de dados?

Ordenar os dados pode ajudar a visualizar isso

Exemplo: ausência de **idade e cabine** são relacionadas?

```
sorted = titanic_df.sort_values('Age')
msno.matrix(sorted, figsize=(8,3))
```



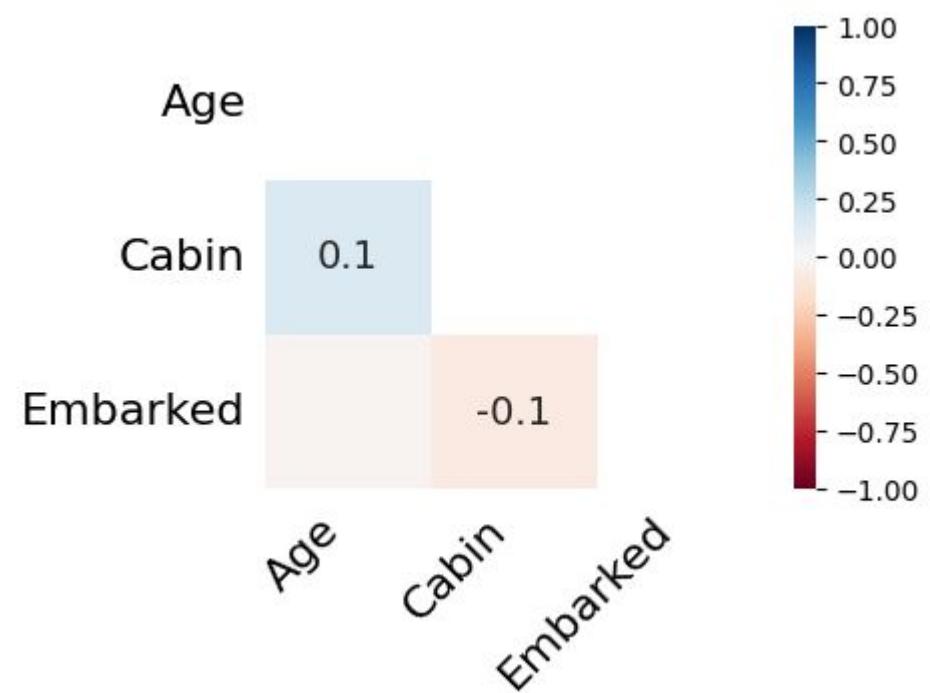
Missingno: heatmap de correlações

Há correlações entre as ausências de dados?

O heatmap mostra correlação entre ausências dos vários atributos

Neste caso fica claro que **não há correlação** entre ausência em Age, Embarked ou Cabin

```
msno.heatmap(titanic_df,figsize=(4,3))
```



Missingno: dendrogram

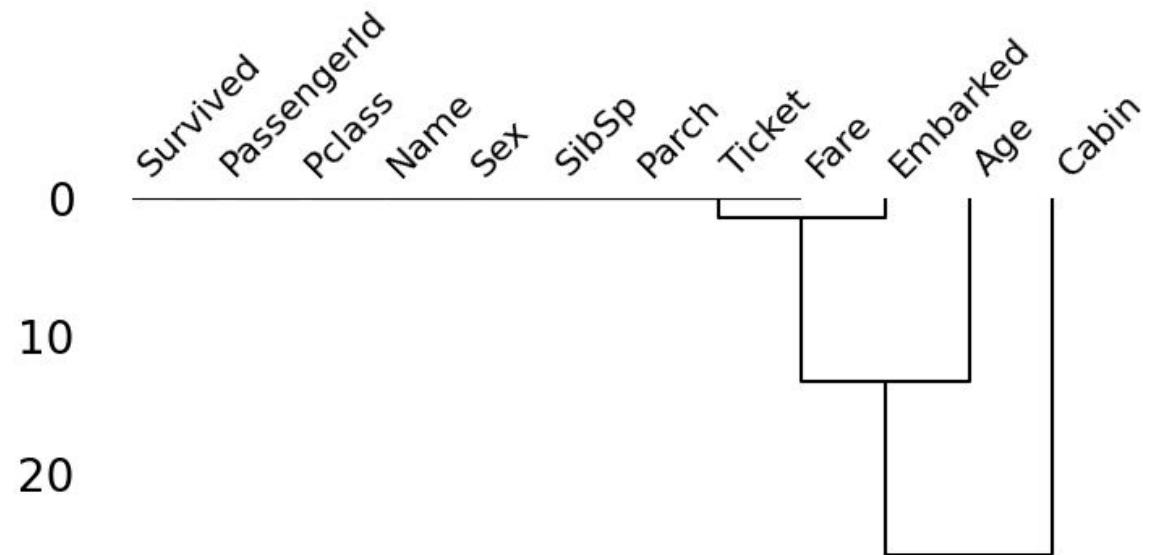
O dendrogram agrupa atributos similares em termos de número de dados ausentes

Embarked é similar a todos os outros atributos sem nulos

Age é mais similar à Embarked que Cabin

Cabin se mostra sem similaridades neste dataset em termos de dados ausentes

```
msno.dendrogram(titanic_df,figsize=(8,3))
```



Tratamento de valores ausentes

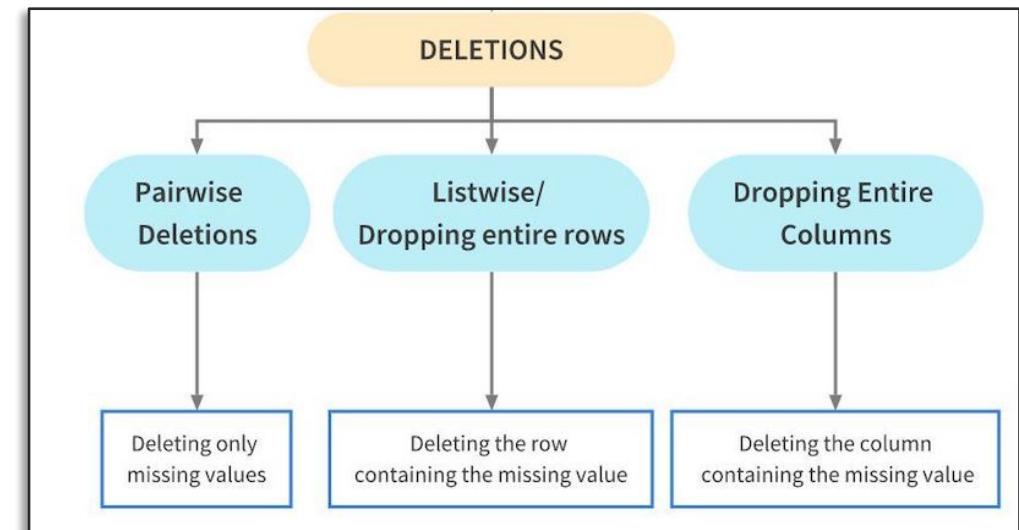
Ignorar linhas com nulos

Ex: funções de cálculo estatístico

Remover linhas com nulos

`dropna()`

Remover colunas



Ignorar linhas com nulos

Pandas ignora valores nulos por default

Ex: sum(), mean(), std(), skew(), quantile(), etc.

```
titanic_df.mean(numeric_only=True)
```

```
PassengerId      446.000000
Survived          0.383838
Pclass            2.308642
Age               29.699118
SibSp             0.523008
Parch             0.381594
Fare              32.204208
dtype: float64
```

Remover linhas com nulos: dropna()

Este procedimento **perde informações!**

```
titanic_1 = titanic_df.copy()
titanic_1.dropna(subset=[ 'Age' ],how='any',inplace=True)
titanic_1[ 'Age' ].isnull().sum()
```

0

Remover colunas

Se a coluna possuir muitos valores nulos,
considere removê-la.

Ex: acima de 70% de nulos

```
ausentes = titanic_1.isnull().sum()/len(titanic_1)
print(ausentes)
```

PassengerId	0.000000
Survived	0.000000
Pclass	0.000000
Name	0.000000
Sex	0.000000
Age	0.000000
SibSp	0.000000
Parch	0.000000
Ticket	0.000000
Fare	0.000000
Cabin	0.740896
Embarked	0.002801
dtype:	float64

Remover colunas

Se a coluna possuir muitos valores nulos,
considere removê-la.

Ex: acima de 70% de nulos

```
cols_remover = ausentes[ausentes > 0.7].index.to_list()
titanic_1.drop(cols_remover, axis=1, inplace=True)
```

```
titanic_1.isnull().sum() / len(titanic_1)
```

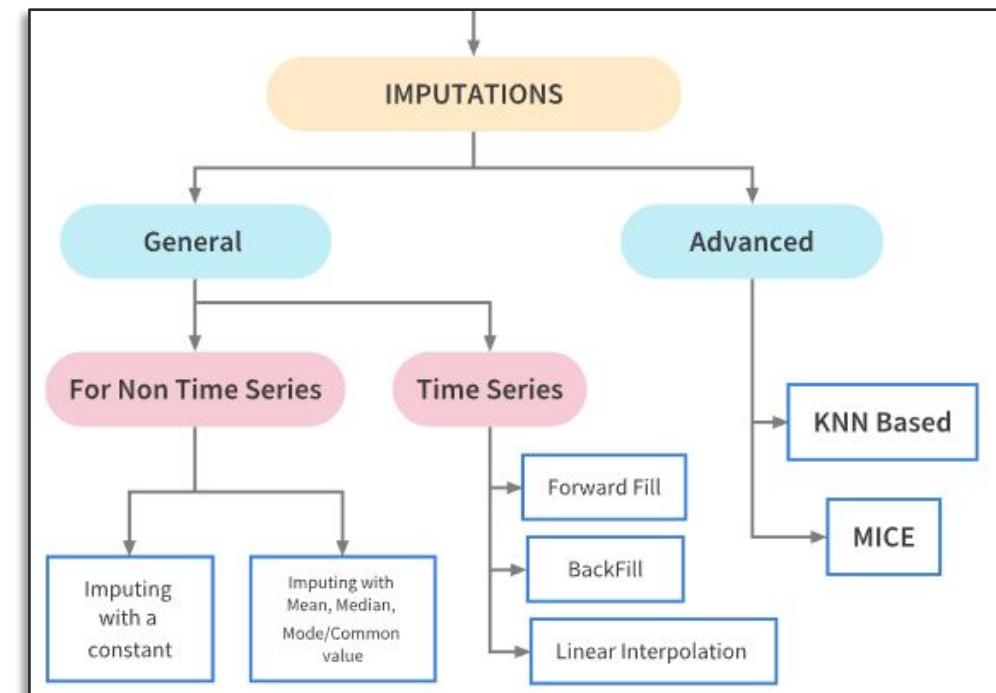
```
PassengerId      0.000000
Survived         0.000000
Pclass           0.000000
Name             0.000000
Sex              0.000000
Age              0.000000
SibSp            0.000000
Parch            0.000000
Ticket           0.000000
Fare             0.000000
Embarked         0.002801
dtype: float64
```

Tratamento de valores ausentes

Imputação de dados

Substituir o valor ausente por um valor pré-definido: constante, média, moda, etc.

Vamos discutir apenas técnicas gerais para dados tabulares



Use uma constante

A estratégia mais simples é marcar valores ausentes de modo explícito.

Ex: “Missing Value”, -99, “Não respondido”, etc.

Método pandas **fillna()** substitui NA por um valor informado

```
titanic_2 = titanic_df.copy()
titanic_2.fillna('missing', inplace=True)
titanic_2[['Fare', 'Cabin', 'Embarked']].head(5)
```

	Fare	Cabin	Embarked
0	7.2500	missing	S
1	71.2833	C85	C
2	7.9250	missing	S
3	53.1000	C123	S
4	8.0500	missing	S

Use uma constante

A estratégia mais simples é marcar valores ausentes de modo explícito.

Ex: "Missing Val."

```
df.fillna( value, method, axis, inplace)  
  
value : scalar, dict, Series, or DataFrame  
method : {'backfill', 'bfill', 'pad', 'ffill', None}, default None  
axis : {0 or 'index', 1 or 'columns'}  
inplace : bool, default False  
  
[1 rows x 4 columns]  
[1 rows x 4 columns] .head(5)
```

2	7.9250	missing	S
3	53.1000	C123	S
4	8.0500	missing	S

Use a média, mediana

Usar a média, mediana, com valores numéricos preserva as informações estatísticas da amostra

```
titanic_3 = titanic_df.copy()
m_age = titanic_df['Age'].mean()
titanic_3['Age'].fillna(m_age,inplace=True)
titanic_3[['Age','Fare','Cabin','Embarked']]
```

	Age	Fare	Cabin	Embarked
0	22.000000	7.2500	NaN	S
1	38.000000	71.2833	C85	C
2	26.000000	7.9250	NaN	S
3	35.000000	53.1000	C123	S
4	35.000000	8.0500	NaN	S
...
886	27.000000	13.0000	NaN	S
887	19.000000	30.0000	B42	S
888	29.699118	23.4500	NaN	S
889	26.000000	30.0000	C148	C
890	32.000000	7.7500	NaN	Q

Use a média, mediana

Usar a média, mediana, com valores numéricos preserva as informações estatísticas da amostra

```
titanic_3 = titanic_df.copy()
m_age = titanic_df['Age'].mean()
titanic_3['Age'].fillna(m_age,inplace=True)
titanic_3[['Age','Fare','Cabin','Embarked']]
```

```
titanic_3.isnull().sum()/len(titanic_3)
```

PassengerId	0.000000
Survived	0.000000
Pclass	0.000000
Name	0.000000
Sex	0.000000
Age	0.000000
SibSp	0.000000
Parch	0.000000
Ticket	0.000000
Fare	0.000000
Cabin	0.771044
Embarked	0.002245
	dtype: float64

Use a moda

Moda é o valor mais frequente

```
titanic_df['Cabin'].value_counts()[:4]
```

```
B96 B98      4
G6           4
C23 C25 C27  4
C22 C26      3
Name: Cabin, dtype: int64
```

```
titanic_df['Cabin'].mode()
```

```
0      B96 B98
1    C23 C25 C27
2          G6
Name: Cabin, dtype: object
```

```
titanic_4 = titanic_df.copy()
m_cabin = titanic_df['Cabin'].mode()[0]
titanic_4['Cabin'].fillna(m_cabin,inplace=True)
titanic_4[['Age','Fare','Cabin','Embarked']].head()
```

	Age	Fare	Cabin	Embarked
0	22.0	7.2500	B96 B98	S
1	38.0	71.2833	C85	C
2	26.0	7.9250	B96 B98	S
3	35.0	53.1000	C123	S
4	35.0	8.0500	B96 B98	S

SimpleImputer do sklearn

SimpleImputer é uma classe flexível para imputação de dados

- Constante
- Média
- Mais frequente

```
from sklearn.impute import SimpleImputer
```

```
titanic_5 = titanic_df.copy()
mean_imputer = SimpleImputer(strategy='constant')
titanic_5.iloc[:, :] = mean_imputer.fit_transform(titanic_5)
titanic_5[['Age', 'Fare', 'Cabin', 'Embarked']].sample(5)
```

		Age	Fare	Cabin	Embarked
636	32.0	7.9250	missing_value	S	
676	24.5	8.0500	missing_value	S	
375	missing_value	82.1708	missing_value	C	
172	1.0	11.1333	missing_value	S	
339	45.0	35.5000	T		S

SimpleImputer do sklearn

SimpleImputer é uma classe flexível para imputação de dados univariados

- Constante
- Média
- Mediana
- Mais frequente

```
titanic_6 = titanic_df.copy()
freq_imputer = SimpleImputer(strategy='most_frequent')
titanic_6.iloc[:, :] = freq_imputer.fit_transform(titanic_6)
titanic_6[['Age', 'Fare', 'Cabin', 'Embarked']]
```

	Age	Fare	Cabin	Embarked
886	27.0	13.00	B96 B98	S
887	19.0	30.00	B42	S
888	24.0	23.45	B96 B98	S
889	26.0	30.00	C148	C
890	32.0	7.75	B96 B98	Q

Técnicas avançadas de imputação

Além das técnicas básicas pode-se utilizar aprendizagem de máquina para escolher um valor adequado

KNN (média dos k vizinhos mais próximos)

MICE

```
from sklearn.impute import KNNImputer

titanic_knn = titanic_df.copy(deep=True)

knn_imputer = KNNImputer(n_neighbors=2, weights="uniform")
titanic_knn['Age'] = knn_imputer.fit_transform(titanic_knn[['Age']])
```

Técnicas avançadas de imputação

Além das técnicas básicas pode-se utilizar aprendizagem de máquina para escolher um valor adequado

KNN (média dos k vizinhos mais próximos)

MICE

```
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer

titanic_mice = titanic_df.copy(deep=True)

mice_imputer = IterativeImputer()
titanic_mice['Age'] = mice_imputer.fit_transform(titanic_mice[['Age']])
```

Verifique se não há valores nulos

```
titanic_6.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin            0
Embarked         0
dtype: int64
```

Dados anômalos (ouliers)



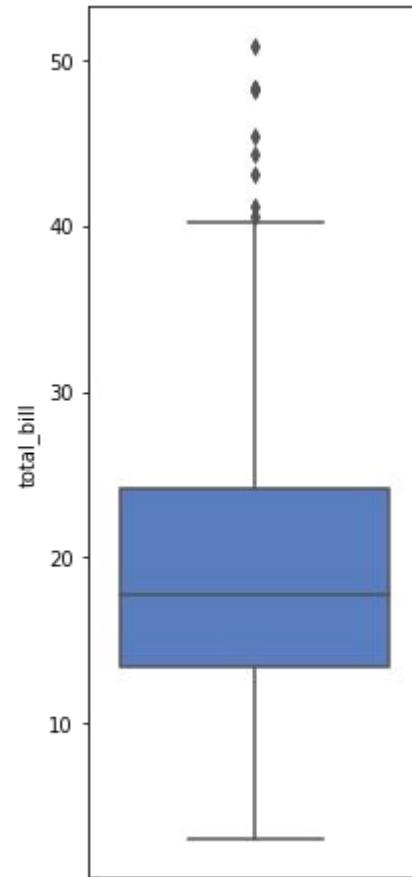
imgflip.com

Dados anômalos (outliers)

Outliers podem distorcer a descrição estatística

Podem indicar:

- erros no processo de aquisição
- Anomalias ou casos atípicos
- mudança na população (*data drift*)



Dados anômalos (outliers)

Diagnóstico

Regra empírica (3 desvios)

Inter Quantile Range (IQR)

Z-score

Tratamento

Desconsiderar as linhas com valores extremos

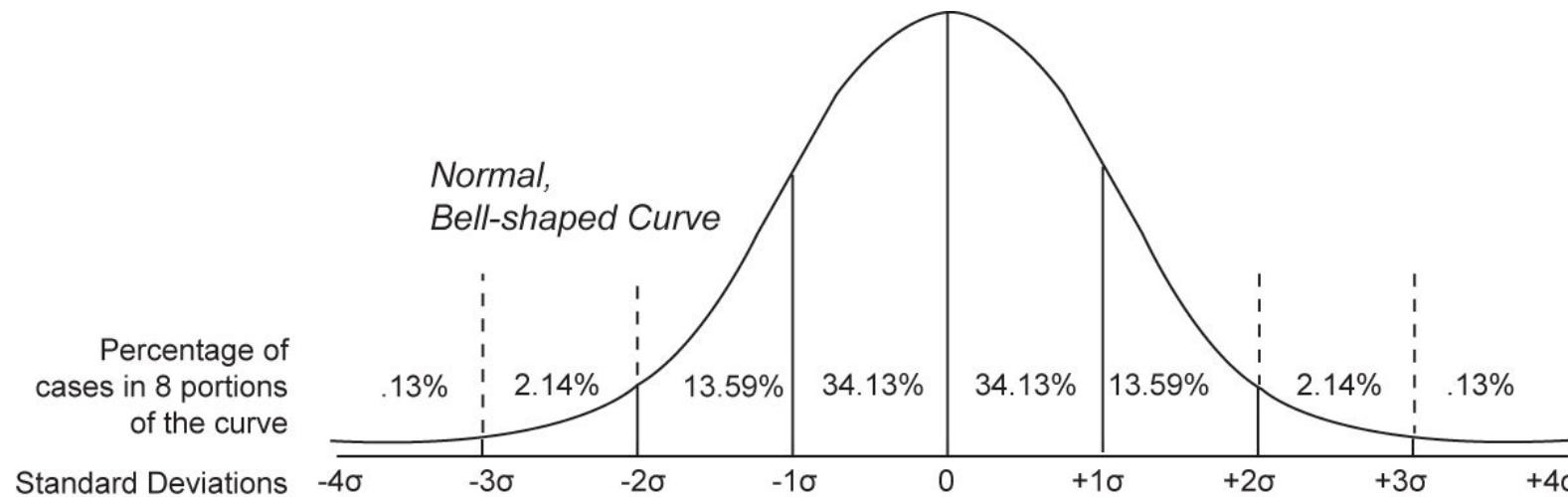
Winsorization (limitação de percentil)

Tratamento de outliers

Regra empírica

99% dos dados ocorre entre -3std e $+3\text{std}$

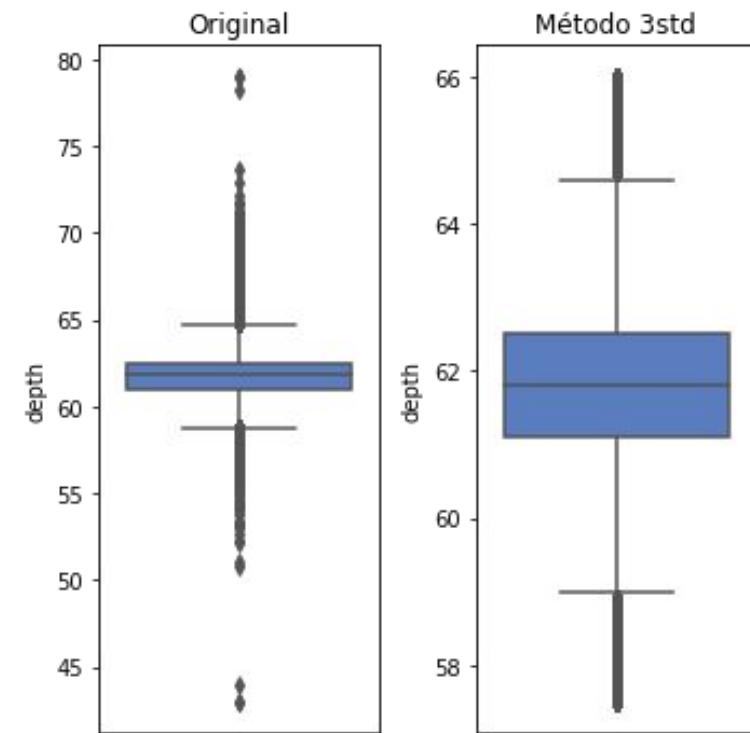
Descartar os dados fora dessa faixa



Tratamento de outliers

Regra empírica

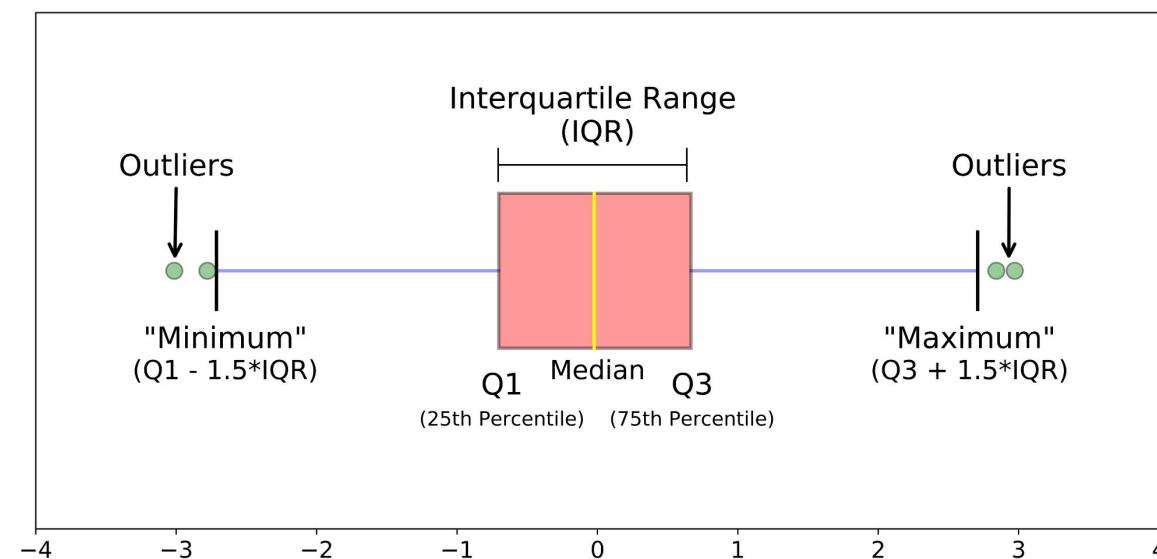
```
def drop_outliers_3std(df, field_name):
    df2 = df.copy()
    m = df2[field_name].mean()
    sd = df2[field_name].std()
    upper = m + 3*sd
    lower = m - 3*sd
    i_up = df2[df2[field_name] > upper].index
    i_lo = df2[df2[field_name] < lower].index
    df2.drop(i_up, axis=0, inplace=True)
    df2.drop(i_lo, axis=0, inplace=True)
    return df2
```



Tratamento de outliers

Inter Quantile Range (IQR)

Desconsiderar valores menores que $Q1 - 1.5 * IQR$ e os maiores que $Q3 + 1.5 * IQR$



Tratamento de outliers

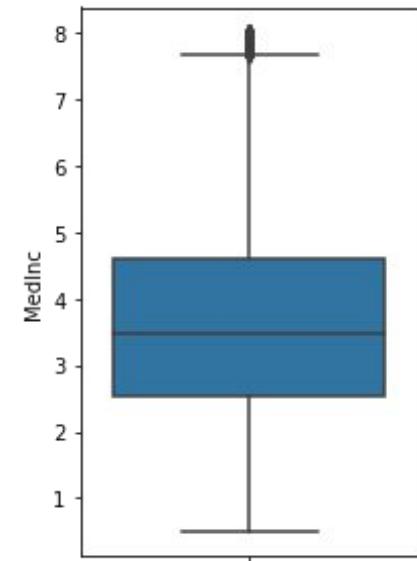
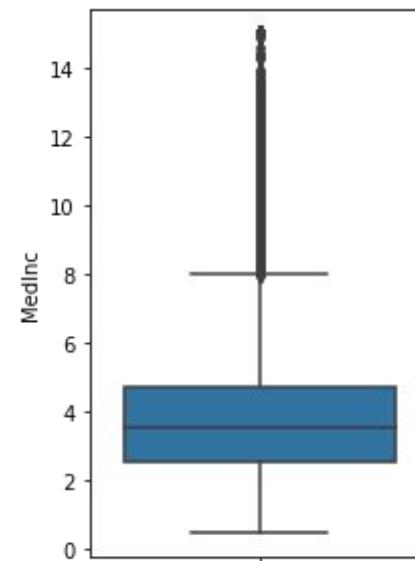
Inter Quantile Range (IQR)

```
def drop_outliers_iqr(df, field_name):
    df2 = df.copy()
    q1 = df2[field_name].quantile(.25)
    q3 = df2[field_name].quantile(.75)
    iqr = q3 - q1
    upper = q3 + 1.5*iqr
    lower = q1 - 1.5*iqr
    i_up = df2[df2[field_name] > upper].index
    i_lo = df2[df2[field_name] < lower].index
    df2.drop(i_up, axis=0, inplace=True)
    df2.drop(i_lo, axis=0, inplace=True)
    return df2
```

```
from sklearn.datasets import fetch_california_housing
housing = fetch_california_housing(as_frame=True)
hous_df= housing['data']
```

```
hous_df2 = drop_outliers(hous_df, 'MedInc')
```

```
f,ax = plt.subplots(figsize=(3,5))
sns.boxplot(data=hous_df2,y='MedInc',ax=ax)
```



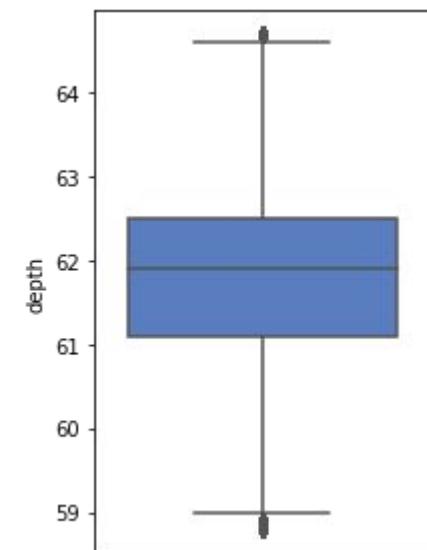
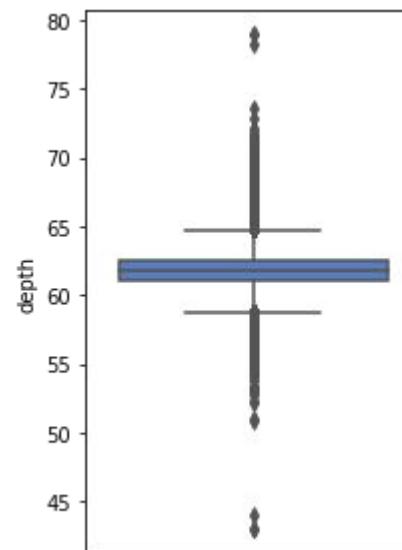
Tratamento de outliers

Inter Quantile Range (IQR)

```
def drop_outliers_iqr(df, field_name):
    df2 = df.copy()
    q1 = df2[field_name].quantile(.25)
    q3 = df2[field_name].quantile(.75)
    iqr = q3 - q1
    upper = q3 + 1.5*iqr
    lower = q1 - 1.5*iqr
    i_up = df2[df2[field_name] > upper].index
    i_lo = df2[df2[field_name] < lower].index
    df2.drop(i_up, axis=0, inplace=True)
    df2.drop(i_lo, axis=0, inplace=True)
    return df2
```

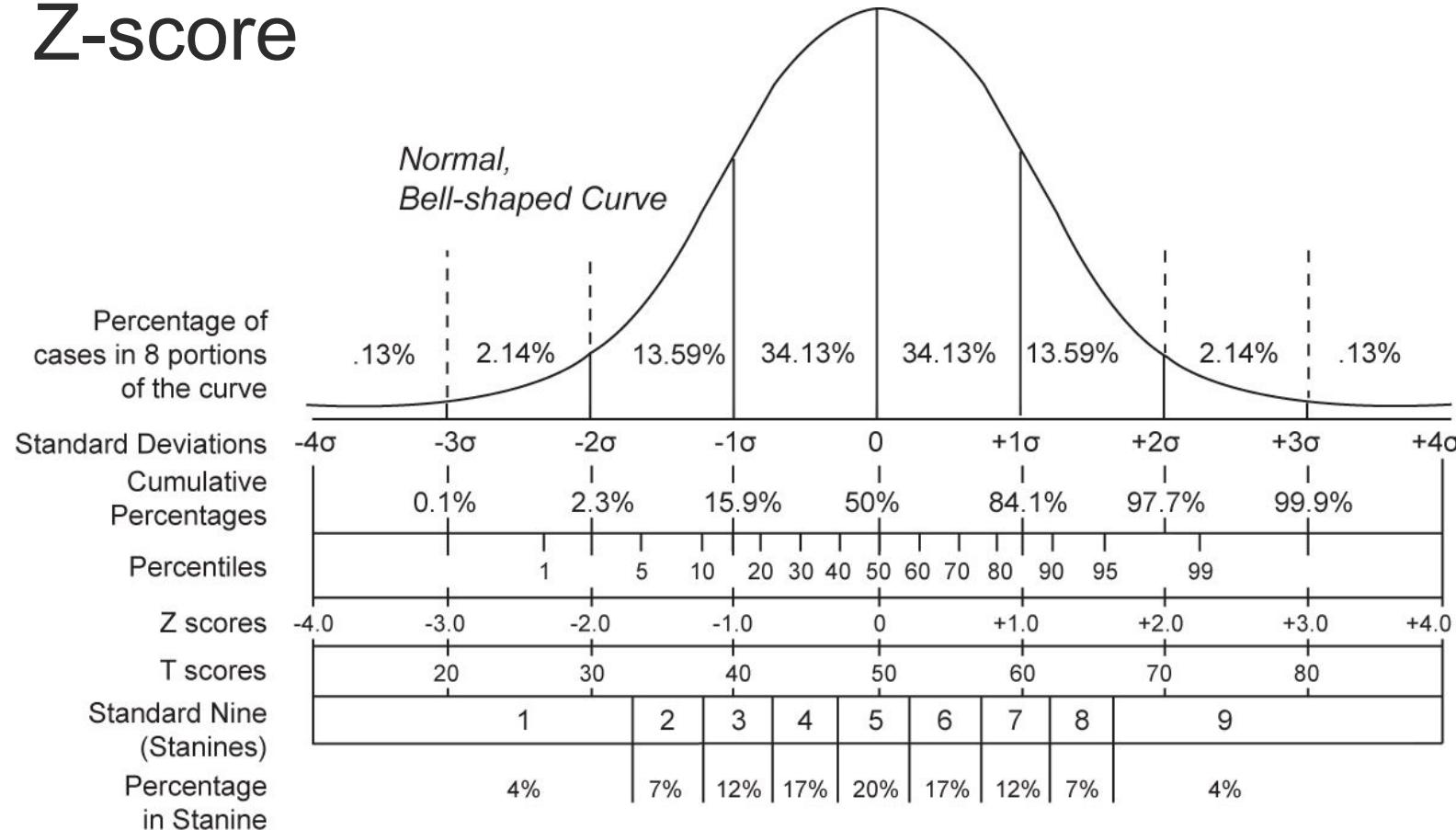
```
diam_df = sns.load_dataset("diamonds")
diam_df2 = drop_outliers(diam_df, 'depth')
```

```
fig, ax = plt.subplots(figsize=(3,5))
sns.boxplot(data=diam_df2,y='depth',palette="muted")
plt.show()
```



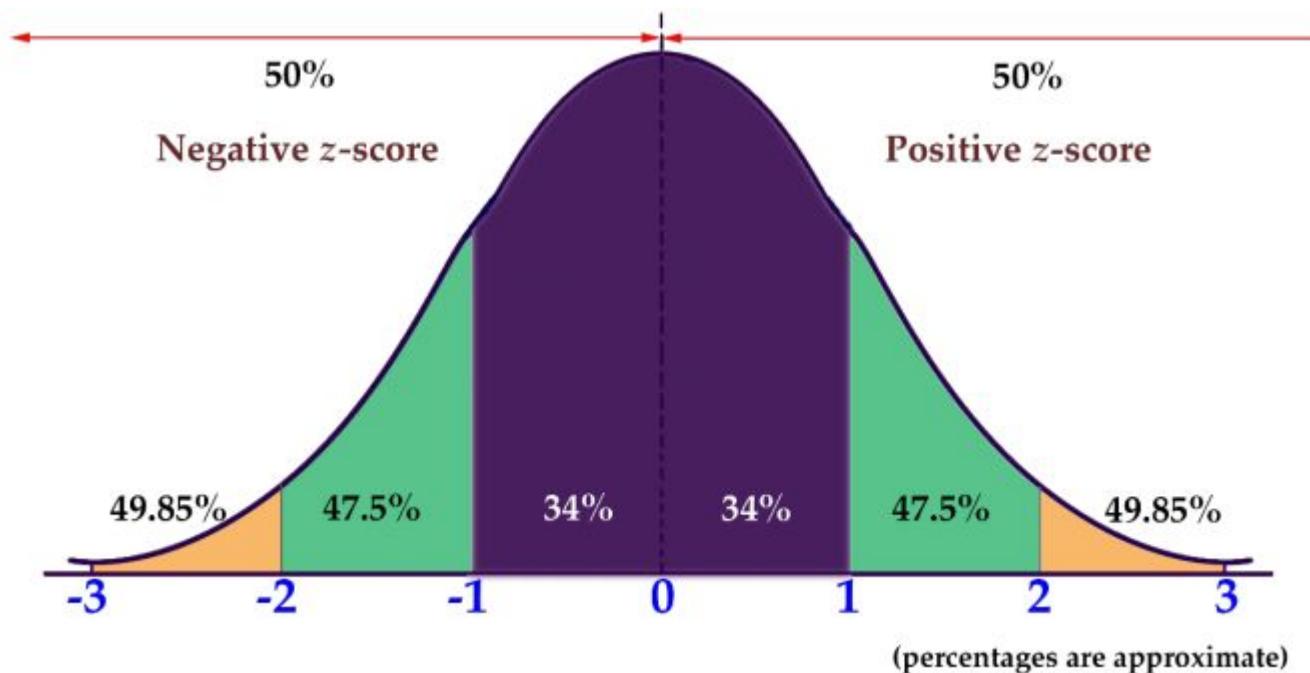
Tratamento de outliers

Z-score



Tratamento de outliers

Z-score



Tratamento de outliers

Transforme os valores originais em z-score e escolha a faixa de interesse

Z-score > 3 abrange 99% dos dados

Z-score >2 abrange 95% dos dados

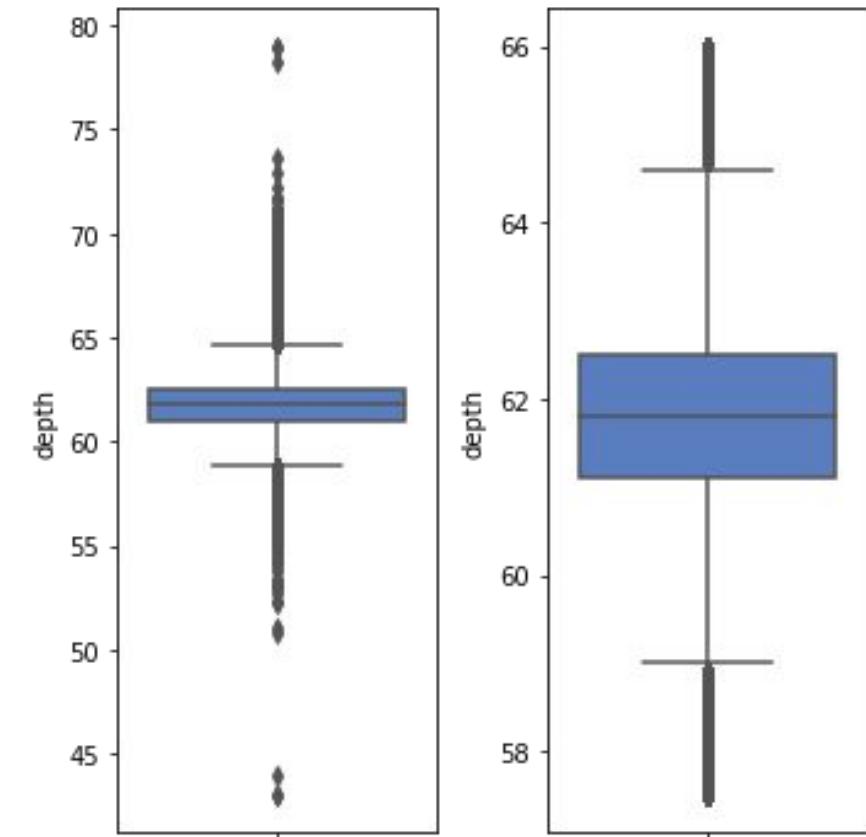
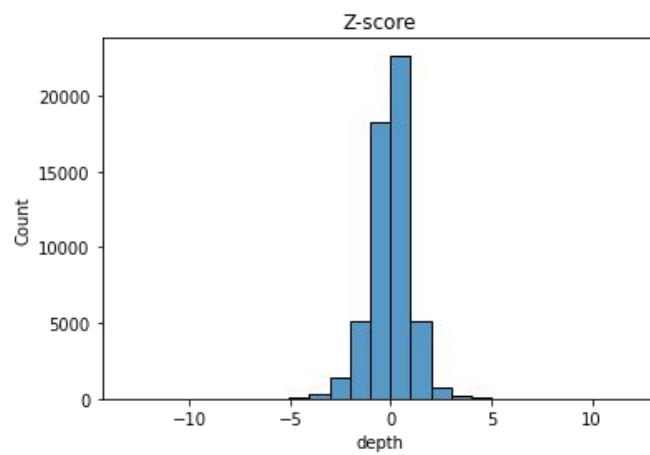
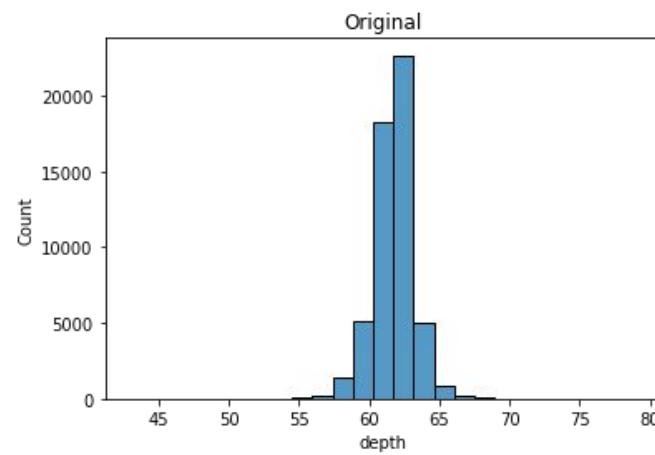
```
from scipy import stats
import numpy as np

diam_z=np.abs(stats.zscore(diam_df['depth']))

diam_df3 = diam_df[diam_z<3]
```

Tratamento de outliers

```
fig, axes = plt.subplots(ncols=2, figsize=(5,5))
sns.boxplot(data=diam_df,y='depth',palette="muted",ax=axes[0])
sns.boxplot(data=diam_df3,y='depth',palette="muted",ax=axes[1])
plt.tight_layout()
plt.show()
```



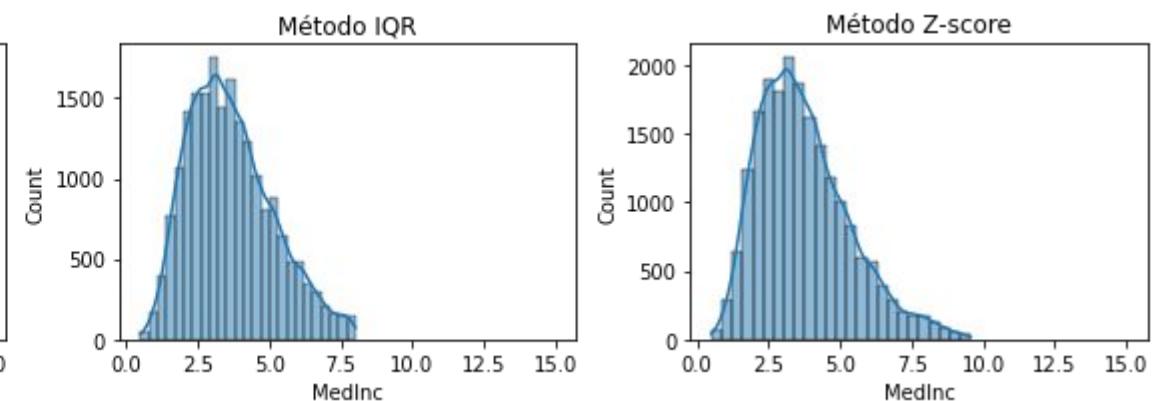
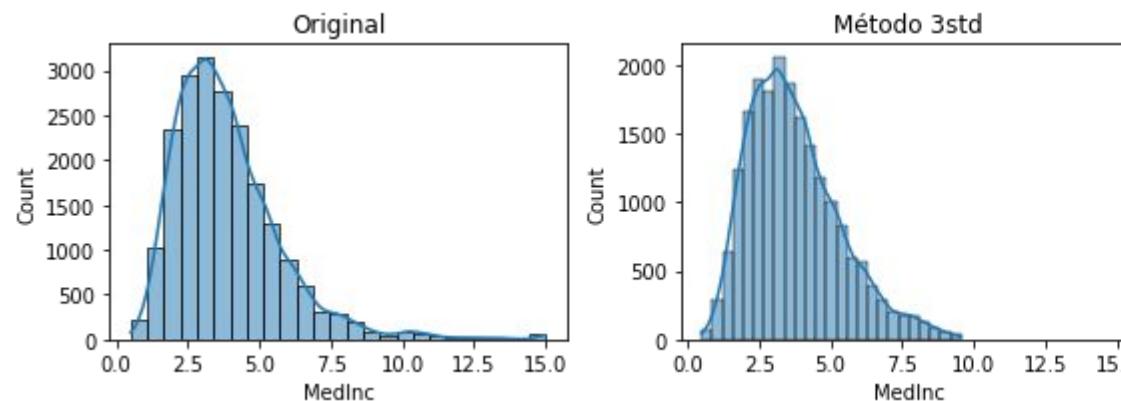
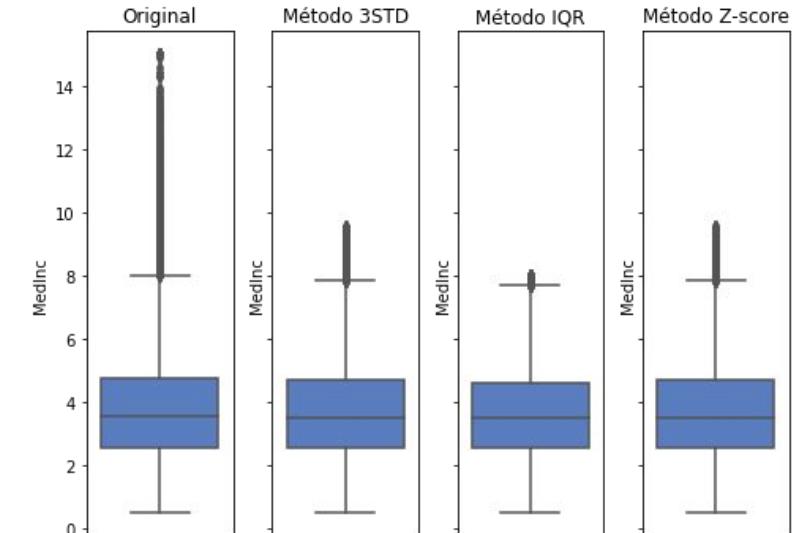
Tratamento de outliers

Comparação

O método empírico é o mais intuitivo

O método IQR é similar ao z-score com corte no valor 2

O método z-score é mais simples de implementar e mais flexível. Equivalente à método empírico



Valores errados



Valores errados

Valores que quebram uma regra de negócio, são ilegais, ou impossíveis

Idade negativa, **pressão arterial zero**, aluno com média de notas maior que 10

Podem indicar erros no processo de aquisição

Após a identificação do erro, use o método **replace()** com um valor adequado

Constante, média, moda, etc

Se não for possível corrigir, remova a linha ou a coluna com erros

Transformação

Transformação de dados

Há vários passos na etapa de transformação, incluindo:

Re-escala

Conversão de tipos

Dados categóricos

Datas

Outras transformações

Reescala dos valores

Remove o efeito de escala em atributos diferentes

Subtrair uma constante e dividir por outra constante

Alguns algoritmos precisam de dados em uma escala comum (cálculo de distância).

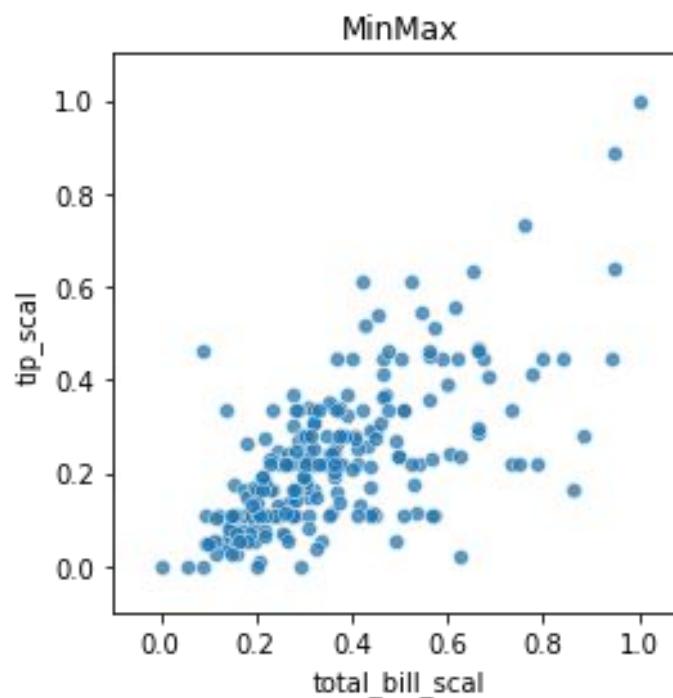
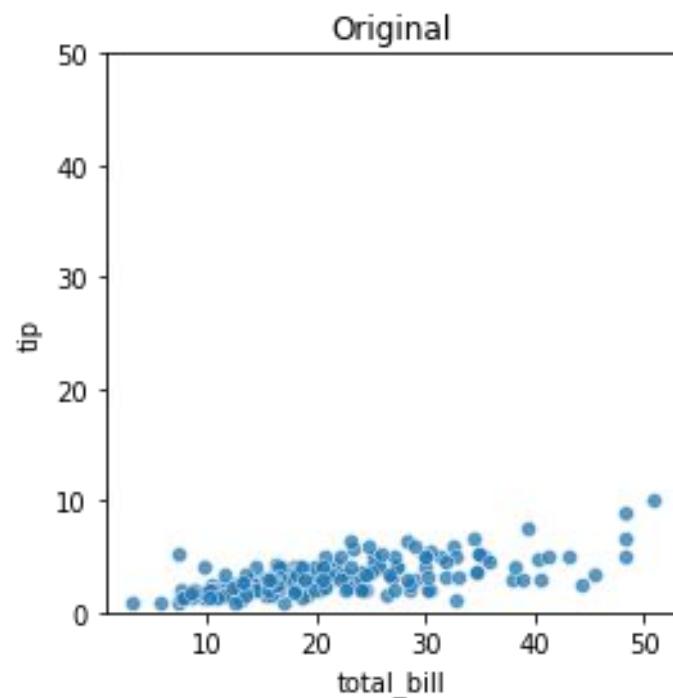
Ex: SVM, KNN, Clustering

Técnicas

MinMax, Robust, Standard (z-score)

Reescala de valores

Exemplo: tips dataset



Reescala com MinMax

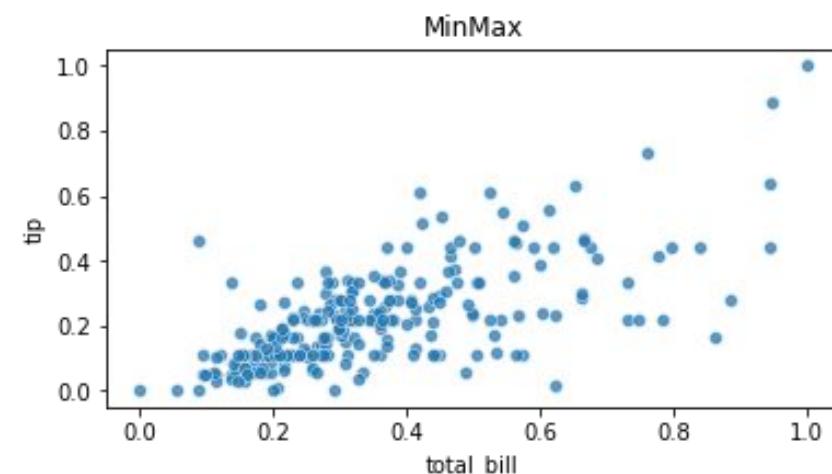
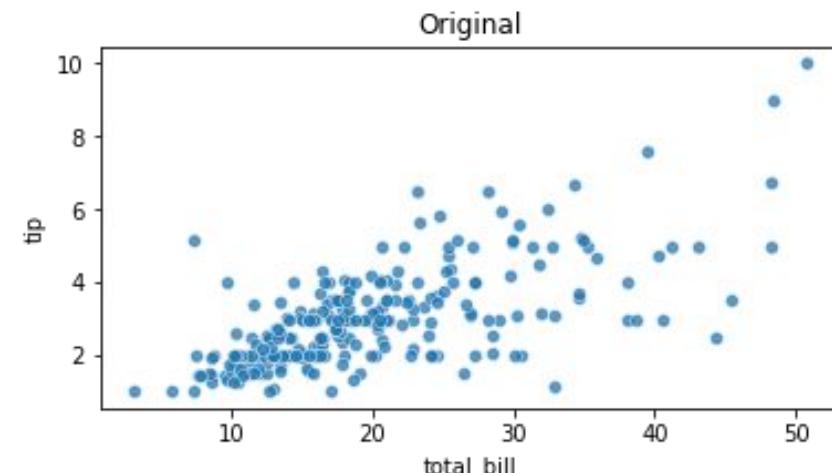
Dados na escala de 0 a 1

$$X_{scal} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

```
def min_max_scaling(x):
    return (x - x.min()) / (x.max() - x.min())

df = tips_df.copy()
cols = ['total_bill', 'tip']

for col in cols:
    df[col] = min_max_scaling(df[col])
```



Reescala com Standard (z-score)

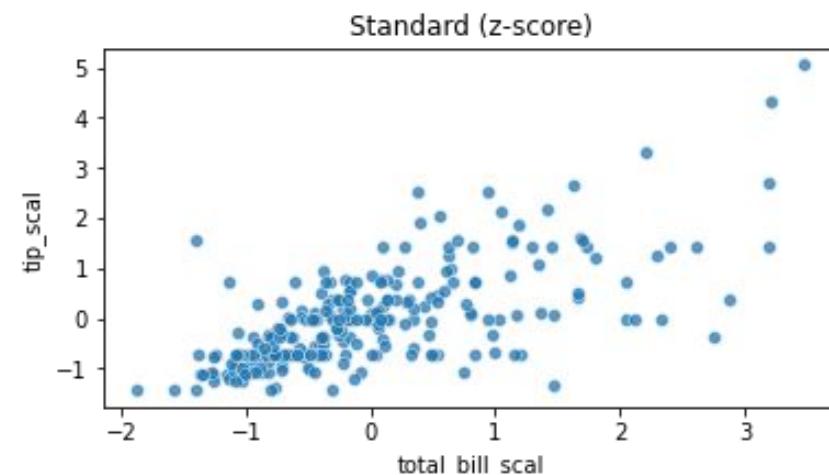
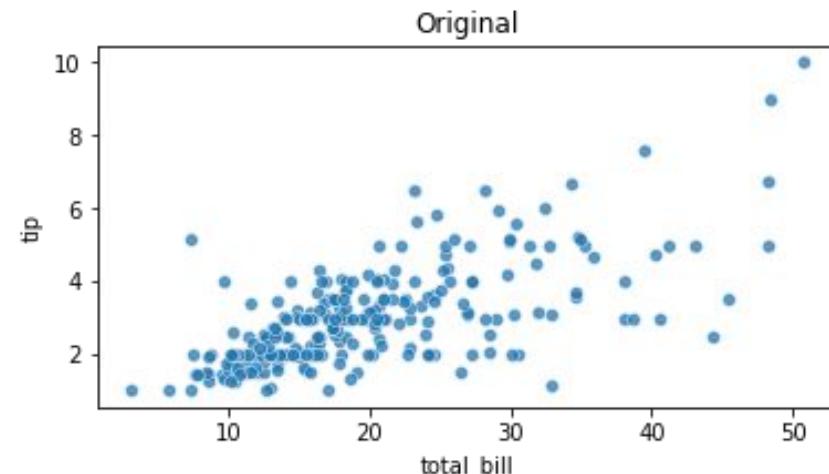
Média 0 e desvio padrão 1

$$X_z = \frac{X - \mu}{\sigma}$$

```
def standard_scaling(x):
    return (x - x.mean()) / x.std()

df = tips_df.copy()
cols = ['total_bill','tip']

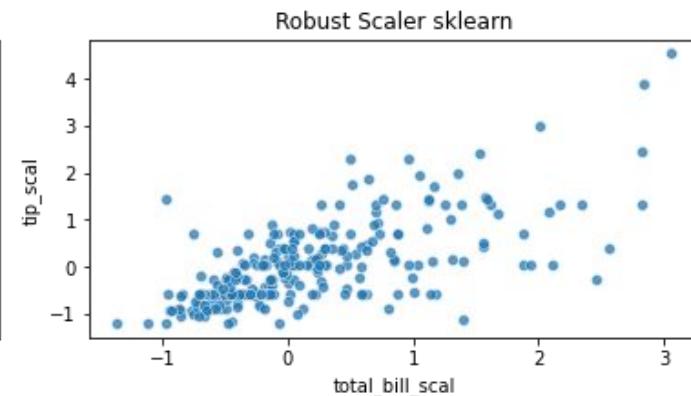
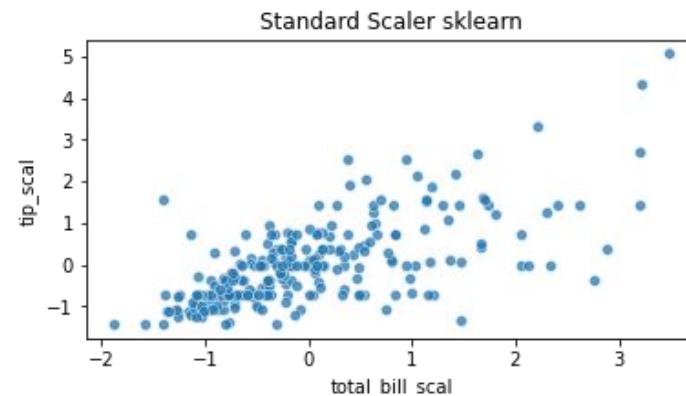
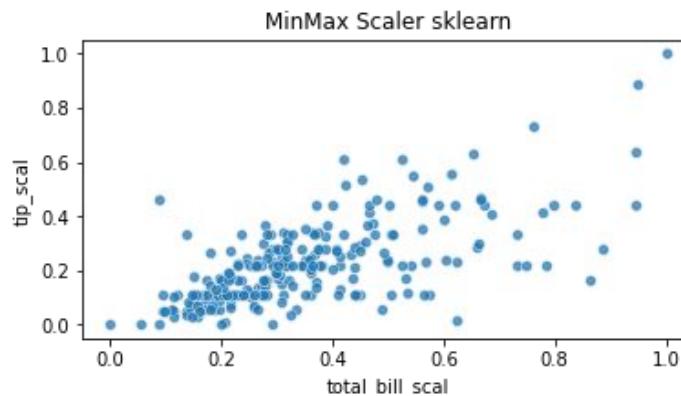
for col in cols:
    df[col] = standard_scaling(df[col])
```



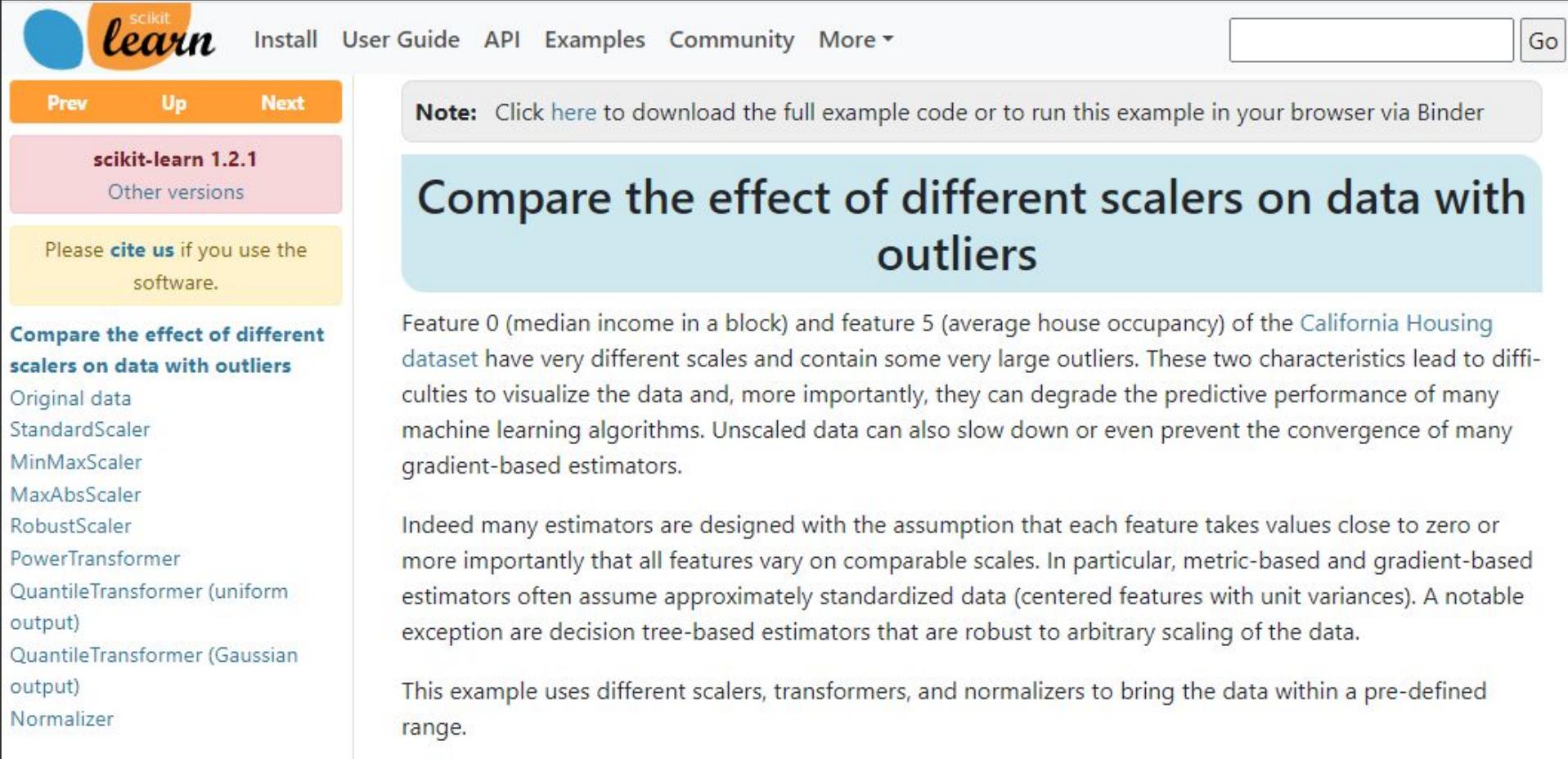
sklearn.preprocessing

```
import MinMaxScaler  
import StandardScaler  
import RobustScaler
```

```
df = tips_df.copy()  
cols = ['total_bill', 'tip']  
  
scaler = MinMaxScaler()  
scaled_data = scaler.fit_transform(df[cols])  
  
for i, col in enumerate(cols):  
    df[col] = scaled_data[:, i]
```



sklearn.preprocessing



The screenshot shows a section of the scikit-learn documentation. At the top, there's a navigation bar with links for 'Install', 'User Guide', 'API', 'Examples', 'Community', and 'More'. Below the navigation bar, there are buttons for 'Prev', 'Up', and 'Next'. A sidebar on the left lists various scalers: 'scikit-learn 1.2.1', 'Other versions', 'Please cite us if you use the software.', 'Compare the effect of different scalers on data with outliers' (which is the current page), 'Original data', 'StandardScaler', 'MinMaxScaler', 'MaxAbsScaler', 'RobustScaler', 'PowerTransformer', 'QuantileTransformer (uniform output)', 'QuantileTransformer (Gaussian output)', and 'Normalizer'. The main content area has a note about downloading code or running it via Binder. The title 'Compare the effect of different scalers on data with outliers' is prominently displayed. The text explains that feature 0 (median income) and feature 5 (average house occupancy) have different scales and contain outliers, which can degrade machine learning performance. It also notes that many estimators assume standardized data. The URL at the bottom is https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html.

https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html

Dados Categóricos

Os dados categóricos são

Nominais. Ex: lista de estados, cores, bandeira de cartão

Ordinais. Ex: dias da semana, tamanho de roupa, clareza de um diamante

Muitos algoritmos exigem dados numéricos

Converter com ordem: ordinal encoding

Converter sem ordem: one hot encoding

Codificação ordinal

Usando o pandas

Use um dicionário com o método replace

Converta a coluna para o tipo category e crie uma ordem

Usando o sklearn.preprocessing

OrdinalEncoder

Codificação ordinal

```
mapa_fumante = {'No':0,'Yes':1}
```

```
tips_df['smoker_code'] = tips_df.smoker.replace(mapa_fumante)
```

```
tips_df.sample(n=10)
```

	total_bill	tip	sex	smoker	day	time	size	smoker_code
109	14.31	4.00	Female	Yes	Sat	Dinner	2	1
77	27.20	4.00	Male	No	Thur	Lunch	4	0
190	15.69	1.50	Male	Yes	Sun	Dinner	2	1
200	18.71	4.00	Male	Yes	Thur	Lunch	3	1
29	19.65	3.00	Female	No	Sat	Dinner	2	0
70	12.02	1.97	Male	No	Sat	Dinner	2	0
161	12.66	2.50	Male	No	Sun	Dinner	2	0
24	19.82	3.18	Male	No	Sat	Dinner	2	0
171	15.81	3.16	Male	Yes	Sat	Dinner	2	1
205	16.47	3.23	Female	Yes	Thur	Lunch	3	1

```
tips_df['day_cat'] = tips_df['day'].cat.codes
```

```
tips_df.sample(n=10)
```

	total_bill	tip	sex	smoker	day	time	size	smoker_code	day_cat
130	19.08	1.50	Male	No	Thur	Lunch	2	0	0
21	20.29	2.75	Female	No	Sat	Dinner	2	0	2
151	13.13	2.00	Male	No	Sun	Dinner	2	0	3
198	13.00	2.00	Female	Yes	Thur	Lunch	2	1	0
222	8.58	1.92	Male	Yes	Fri	Lunch	1	1	1
183	23.17	6.50	Male	Yes	Sun	Dinner	4	1	3
182	45.35	3.50	Male	Yes	Sun	Dinner	3	1	3
144	16.43	2.30	Female	No	Thur	Lunch	2	0	0
15	21.58	3.92	Male	No	Sun	Dinner	2	0	3
90	28.97	3.00	Male	Yes	Fri	Dinner	2	1	1

Codificação ordinal

Se a coluna não for category, use
`astype('category')`

```
codes = taxi_df['dropoff_borough'].astype('category').cat.codes
```

```
taxi_df['dropoff_borough_code'] = codes
```

```
taxi_df[['dropoff_borough','dropoff_borough_code']].sample(n=5)
```

	dropoff_borough	dropoff_borough_code	
2091	Manhattan	2	
6315	Brooklyn	1	
6058	Manhattan	2	
2649	Manhattan	2	
3899	Manhattan	2	

```
taxi_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6433 entries, 0 to 6432
Data columns (total 14 columns):
 #   Column           Non-Null Count Dtype  
 --- 
 0   pickup          6433 non-null   object 
 1   dropoff         6433 non-null   object 
 2   passengers      6433 non-null   int64  
 3   distance        6433 non-null   float64 
 4   fare             6433 non-null   float64 
 5   tip              6433 non-null   float64 
 6   tolls            6433 non-null   float64 
 7   total            6433 non-null   float64 
 8   color            6433 non-null   object 
 9   payment          6389 non-null   object 
 10  pickup_zone     6407 non-null   object 
 11  dropoff_zone    6388 non-null   object 
 12  pickup_borough  6407 non-null   object 
 13  dropoff_borough 6388 non-null   object 
dtypes: float64(5), int64(1), object(8)
memory usage: 703.7+ KB
```

Codificação ordinal

Definindo uma ordem

```
diam_df['clarity'].cat.ordered  
False  
  
c = diam_df['clarity'].cat.categories.values.tolist()  
print(c)  
  
['IF', 'VVS1', 'VVS2', 'VS1', 'VS2', 'SI1', 'SI2', 'I1']  
  
c.reverse()  
print(c)  
  
['I1', 'SI2', 'SI1', 'VS2', 'VS1', 'VVS2', 'VVS1', 'IF']  
  
diam_df['clarity']=diam_df['clarity'].cat.set_categories(c,ordered=True)
```

```
filtro = diam_df['clarity']>='VVS2'  
diam_df[filtro].sample(n=5)
```

	carat	cut	color	clarity
36697	0.34	Ideal	G	IF
31640	0.25	Very Good	H	VVS1
44956	0.58	Very Good	I	VVS1
6775	0.81	Ideal	H	IF
3788	0.74	Ideal	G	VVS2

Codificação Nominal

Método `get_dummies()`

Retorna um dataframe com **novas colunas** binárias nomeadas de acordo com os valores da categoria

Indicado para categorias com **baixa cardinalidade**

```
pd.get_dummies(auto_df['num-of-cylinders']).
```

	eight	five	four	six	three	twelve	two
6	0	1	0	0	0	0	0
202	0	0	0	1	0	0	0
49	0	0	0	0	0	1	0
189	0	0	1	0	0	0	0
53	0	0	1	0	0	0	0

Categorias com sklearn.preprocessing

Nominais

```
from sklearn.preprocessing import OneHotEncoder, OrdinalEncoder

ohe = OneHotEncoder(sparse=False)

transf_ohe = ohe.fit_transform(taxi_df[['dropoff_borough']])

transf_ohe[:5]

array([[0., 0., 1., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 1., 0., 0.]])
```

Categorias com sklearn.preprocessing

Ordinais

```
from sklearn.preprocessing import OneHotEncoder, OrdinalEncoder

ord = OrdinalEncoder()

transf_ord = ord.fit_transform(auto_df[['drive-wheels',
                                         'num-of-cylinders',
                                         'make']])

transf_ord[:5]

array([[2., 2., 0.],
       [2., 2., 0.],
       [2., 3., 0.],
       [1., 2., 1.],
       [0., 1., 1.]])
```

Outras transformações

Durante o processo de preparação pode ser necessário

Converter valores (temperatura, moeda, etc)

Converter tipos (de string para data, ou float)

Extração de informações de strings

Aplicar transformações específicas (log transform)

Em pandas pode-se utilizar

`replace()`

`apply()`

`map()`

Dados Organizados (tidy data)

O que são dados organizados (tidy data)?

É muito comum dados serem apresentados em formatos que são fáceis de visualizar, mas difíceis de tratar computacionalmente.

Ex: wide data, tabelas de contingencias, etc

Tidy data é um padrão de organização de dados introduzido por Hadley Wickham em 2014

Tornou-se padrão de formato para análise de dados na comunidade de R e visualização de dados

Características

As principais características de dados organizados para análise são:

1. Cada variável corresponde a uma coluna
2. Cada observação corresponde a uma tupla (linha)
3. Cada unidade de interesse forma uma tabela distinta

Cada variável tem sua coluna, cada amostra tem uma linha e cada tipo de dados tem uma tabela

Organizando dados em pandas

O principal método para organizar os dados em pandas é o método melt()

Produz um formato que extrai valores que residem em colunas

Melhor para analisar computacionalmente

Ao contrário de dados wide, produz o formato long

Exemplo: pew.csv (desorganizado)

Alguns nomes de colunas são valores de variável

```
pew_df = pd.read_csv('pew-untidy.csv')
```

```
pew_df.head()
```

	religion	<\$10k	\$10-20k	\$20-30k	\$30-40k	\$40-50k	\$50-75k	\$75-100k	\$100-150k	>150k	Don't know/refused
0	Agnostic	27	34	60	81	76	137	122	109	84	96
1	Atheist	12	27	37	52	35	70	73	59	74	76
2	Buddhist	27	21	30	34	33	58	62	39	53	54
3	Catholic	418	617	732	670	638	1116	949	792	633	1489
4	Don't know/refused	15	14	15	11	10	35	21	17	18	116

Pew organizado

Cada variável tem uma coluna

religion <\$10k	
0	Agnostic 27
1	Atheist 12
2	Buddhist 27
3	Catholic 418
4	Don't know/refused 15

```
tidy_pew_df=pew_df.melt(id_vars=coluna_id,  
                        value_vars=coluna_valores,  
                        var_name='income',  
                        value_name='freq')  
  
tidy_pew_df.head()
```

	religion	income	freq
0	Agnostic	<\$10k	27
1	Atheist	<\$10k	12
2	Buddhist	<\$10k	27
3	Catholic	<\$10k	418
4	Don't know/refused	<\$10k	15

Exemplo: billboard.csv

Alguns nomes de colunas são valores de variável

```
bil_df = pd.read_csv("billboard-untidy.csv", encoding="mac_latin2")
bil_df.head(10)
```

	year	artist.inverted	track	time	genre	date.entered	date.peaked	x1st.week	x2nd.week	x3rd.week	...	x67th.w
0	2000	Destiny's Child	Independent Women Part I	3:38	Rock	2000-09-23	2000-11-18	78	63.0	49.0	...	NaN
1	2000	Santana	Maria, Maria	4:18	Rock	2000-02-12	2000-04-08	15	8.0	6.0	...	NaN
2	2000	Savage Garden	I Knew I Loved You	4:07	Rock	1999-10-23	2000-01-29	71	48.0	43.0	...	NaN
3	2000	Madonna	Music	3:45	Rock	2000-08-12	2000-09-16	41	23.0	18.0	...	NaN
4	2000	Aguilera, Christina	Come On Over Baby (All I Want Is You)	3:38	Rock	2000-08-05	2000-10-14	57	47.0	45.0	...	NaN

Bilboard organizado

Week agora é uma coluna

```
bil_df_melt = df = pd.melt(frame=bil_df,
                             id_vars=id_vars,
                             var_name="week",
                             value_name="rank")
```

```
bil_df_melt.head()
```

	year	artist.inverted		track	time	genre	date.entered	date.peaked	week	rank
0	2000	Destiny's Child	Independent Women Part I		3:38	Rock	2000-09-23	2000-11-18	x1st.week	78.0
1	2000	Santana	Maria, Maria		4:18	Rock	2000-02-12	2000-04-08	x1st.week	15.0
2	2000	Savage Garden	I Knew I Loved You		4:07	Rock	1999-10-23	2000-01-29	x1st.week	71.0
3	2000	Madonna	Music		3:45	Rock	2000-08-12	2000-09-16	x1st.week	41.0
4	2000	Aguilera, Christina	Come On Over Baby (All I Want Is You)	3:38	Rock	2000-08-05	2000-10-14	x1st.week	57.0	

Outros problemas

Informações de entidades diferentes em uma mesma tabela

Separar as informações em tabelas diferentes

Similar a normalização

Múltiplas informações em nomes de colunas

Separar e extrair a informação a partir dos nomes das colunas

Conclusão

Resumo

Preparação de dados é uma etapa essencial no processo de análise de dados

Nessa aula vimos os como tratar os principais problemas com dados, incluindo dados ausentes e valores extremos. Também discutimos várias técnicas de transformações de dados

A próxima etapa, é realizar análise exploratória com os dados para descobrir padrões interessantes para tomada de decisões

Referências

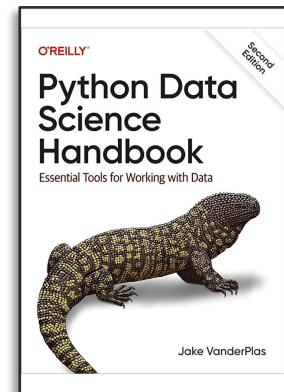
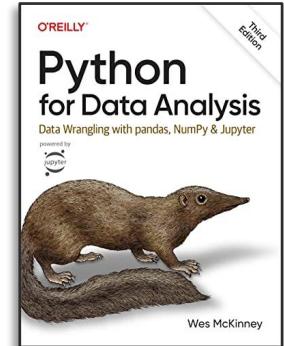
Livros

McKINNEY, Wes. "Python for Data Analysis". O'Reilly, 2017.

VANDERPLASS, Jake. "Python Data Science Handbook". O'Reilly, 2016.

HARRISON, Matt. "Machine Learning: guia de referencia rápida". Novatec, 2020.

CHEN, Daniel Y. "Analise de Dados com Python e Pandas". Novatec, 2018.



Links

<https://towardsdatascience.com/data-cleaning-with-python-and-pandas-detecting-missing-values-3e9c6ebcf78b>

<https://medium.com/analytics-vidhya/outlier-treatment-9bbe87384d02>

<https://www.pluralsight.com/guides/preparing-data-modeling-scikit-learn/>

<https://www.pluralsight.com/guides/cleaning-up-data-from-outliers>