```
import pandas as pd
import numpy as np

QE = pd.read_csv("https://raw.githubusercontent.com/DATAUNIRIO/Base_de_dados/master/(
```

```
# listar os objetos
%whos
```

```
Variable   Type         Data/Info
---------------------------------
QE         DataFrame        Aluno  T Mora_pais   <...>n\n[95 rows x 10 columns]
np         module       <module 'numpy' from '/us<...>kages/numpy/__init__.py'>
pd         module       <module 'pandas' from '/u<...>ages/pandas/__init__.py'>
```

```
# as 3 primeiras linhas do banco de dados
print(QE.head(3))
```

```
   Aluno  T Mora_pais   RJ Namorado_a Trabalha  Desempenho  Estresse  \
0      1  1       Nao  Nao        Nao      Nao        8.89        23
1      2  1       Sim  Sim        Nao      Nao        8.80        24
2      3  1       Nao  Nao        Nao      Nao        8.00        25

   Creditos  Horas_estudo
0      27.0            27
1      28.0            28
2      25.0            25
```

```
QE.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 95 entries, 0 to 94
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Aluno         95 non-null     int64
 1   T             95 non-null     int64
 2   Mora_pais     95 non-null     object
 3   RJ            95 non-null     object
 4   Namorado_a    95 non-null     object
 5   Trabalha      95 non-null     object
 6   Desempenho    95 non-null     float64
 7   Estresse      95 non-null     int64
 8   Creditos      94 non-null     float64
 9   Horas_estudo  95 non-null     int64
dtypes: float64(2), int64(4), object(4)
memory usage: 7.5+ KB
```

VARIÁVEL QUALITATIVA

1. tabela em números absolutos
2. proporções
3. gráfico de pizza
4. gráfico de barras

```
#fazendo tabelas
tabela_simples = QE.Trabalha.value_counts()
tabela_simples
```

```
Nao    59
Sim    36
Name: Trabalha, dtype: int64
```

```
#fazendo proporções
#tabela_simples/tabela_simples.sum()*100

round(tabela_simples/tabela_simples.sum()*100,2)
```

```
Nao    62.11
Sim    37.89
Name: Trabalha, dtype: float64
```

```
# grafico de pizza
tabela_simples.plot.pie()
```

```
# grafico de barras
#tabela_simples.plot.bar()
#tabela_simples.plot.bar(color="red")
tabela_simples.plot.bar(color=["red","blue"])
```

VARIÁVEL QUANTITATIVA

Vamos fazer:

- Resumos
- Histograma

```
# O ponto (".") pode ser o $ ou ::
# aqui vou usar como $ para selecionar variáveis
QE.Horas_estudo.describe()
```

```
QE.Desempenho.describe()
```

```
# histograma
#QE.Horas_estudo.plot.hist()
QE.Horas_estudo.plot.hist(color="red")
```

DUAS VARIÁVEIS QUALITATIVAS

Vamos fazer:

- Tabela para duas variáveis (crosstab)
- barplot para duas variáveis

```
#tabela = pd.crosstab(QE.Trabalha, QE.Mora_pais)
tabela = pd.crosstab(QE.Trabalha, QE.Mora_pais, rownames=['Trabalha'], colnames=['Mor
tabela
```

```
# soma da coluna (100 na coluna)
round(tabela/tabela.sum()*100,2)
```

```
# abordagem melhor: use o normalise
# normalise por total (all), linhas (index), ou colunas (columns).
tabela_linha = pd.crosstab(QE.Trabalha, QE.Mora_pais, rownames=['Trabalha'], colnames
tabela_linha
```

```
# normalise por colunas (columns).
tabela_coluna = pd.crosstab(QE.Trabalha, QE.Mora_pais, rownames=['Trabalha'], colname
round(tabela_coluna*100,2)
```

```
tabela.plot.bar()
```

Uma variável qualitativa e uma variável quantitativa

Vamos fazer:

1. Resumo por grupos
2. Boxplot

```
QE.groupby("Trabalha").agg(horas_media =("Horas_estudo", "mean"),horas_mediano = ("Ho
```

```
QE.groupby("Trabalha").agg(minimo=("Horas_estudo","min"), horas_media =("Horas_estudo
```

```
QE.groupby("Trabalha").agg(horas_media =("Horas_estudo", "mean"),desvio_padrao = ("Ho
```

```
QE.boxplot("Horas_estudo",by='Trabalha',color="red")
```

DUAS VARIÁVEIS QUANTITATIVAS

1. Diagrama de dispersão
2. Coeficiente de correlação

```
# Draw a scatter plot
QE.plot.scatter(x = 'Horas_estudo', y = 'Desempenho', s = 100);
```

```
# Draw a scatter plot and here size of dots determined by price
QE.plot.scatter(x = 'Horas_estudo', y = 'Desempenho', s = 'Horas_estudo', c = 'red');
```

```
# The Pandas Plot Function
#df.plot(
#    x=None,          # Values to use for x axis
#    y=None,          # Values to use for y axis
#    kind='line',     # The type of chart to make
#    title=None,      # The title to use
#    legend=False,    # Whether to show a legend
#    xlabel=None,     # What the x-axis label should be
#    ylabel=None      # What the y-axis label should be
#    c=None,          # The color to use for the dots
#    s=None           # How to size dots (single number or column)
#)
```

```
QE.plot.scatter(x = 'Horas_estudo', y = 'Desempenho', s = 'Horas_estudo', c = 'red',t
```

```
# filter de colunas no python e linhas no R
QE.filter(["Horas_estudo", "Desempenho"]).corr()
```

```
 QE.filter(["Horas_estudo", "Desempenho","Estresse","Creditos"]).corr()
```

```
 QE.filter(["Horas_estudo", "Desempenho","Estresse","Creditos"]).corr(method='spearma
```

```
# plot the heatmap
import seaborn as sns
```

```
import seaborn as sns
%matplotlib inline

correlacao =  QE.filter(["Horas_estudo", "Desempenho","Estresse","Creditos"]).corr()
sns.heatmap(correlacao)


sns.heatmap(correlacao, cmap="Blues", annot=True)


# posso fazer:
(QE.assign(
     tem_na = QE.Horas_estudo.isna(),
     tem_sim = QE.Trabalha.str.contains("Sim")
     # esse é um jeito de você acessar métodos mais "básicos" de um objeto
     ))



# (1) selecionar linha com o query
# (2) selecionar coluna com o filter
# (3) criar colunas com o assign

QE.query("Trabalha=='Sim'").Namorado_a



#QE.query("T==1").Namorado_a
#QE.query("T!=1").Namorado_a
#QE.query("T==2|T==3").Namorado_a

#carro.query("Type in ('Small','Midsize')")
# igual no R seria assi, QE %>% filter(T %in% c(2,3))

QE.query("T in (2,3)").Namorado_a



QE.columns

     Index(['Aluno', 'T', 'Mora_pais', 'RJ', 'Namorado_a', 'Trabalha',
     'Desempenho',
            'Estresse', 'Creditos', 'Horas_estudo'],
           dtype='object')


len(QE)

     95
```

Regressão

```
import statsmodels.formula.api as smf
```

```
#fit regression model
fit = smf.ols('Desempenho ~ Horas_estudo', data=QE).fit()

#view model summary
print(fit.summary())
```

Avaliando o pressuposto de normalidade

```
import statsmodels.api as sm

sm.qqplot(fit.resid, line='s');
```

```
# Shapiro-Wilk
from scipy.stats import shapiro

stat, p = shapiro(fit.resid)
print('stat=%.3f, p=%.3f\n' % (stat, p))
```

```
     stat=0.847, p=0.000
```

Teste de Homocedasticidade

```
from statsmodels.compat import lzip
import statsmodels.stats.api as sms

#perform Bresuch-Pagan test
names = ['Lagrange multiplier statistic', 'p-value',
         'f-value', 'f p-value']
teste = sms.het_breuschpagan(fit.resid, fit.model.exog)

lzip(names, teste)
```

```
d = {'Horas_estudo': [10, 20,30]}
Horas_estudos = pd.DataFrame(data=d)
fit.predict(Horas_estudos)

# No R
#d <- data.frame(Horas_estudo=c(10,20,30))
#predict(modelo,d)
```

[meu livro de regressão com o colab]