- **Imports:**

```
%matplotlib inline
import pandas as pd
import matplotlib.pyplot as plt
import random
import numpy as np
import itertools
import seaborn as sns
from scipy import stats
from scipy.stats import expon, norm, uniform
```

- **Análise exploratória**

*OBS: dados é um DataFrame (uma tabela); as colunas são do tipo Series

Leitura

Lendo arquivos excel:
•dados = pd.read_excel('nome_do_arquivo')
Mostrar os primeiros ou últimos:
•dados.head( ) ou dados.tail( )
Tamanho de um DataFrame:
•dados.shape

OBS: sintaxe do pandas

- and = &
- or = |
- not = ~

Mexendo com linhas e colunas

List devolve uma lista das colunas do DataFrame. Ex:
•list(dados)

Colunas podem ser acessadas direto pelo nome, como atributos de objetos. Ex:
•dados.IDADE ou dados[IDADE]

Para definir condições, escrever dentro das chaves. Ex:
•dados[dados.IDADE>40]

Para mudar o índice, usar:
•DataFrame.set_index('*nome do índice*')

Mudar nome das linhas em uma tabela:
•DataFrame.reindex(‘*nova ordem, provavelmente em formato de lista’*)

Como trocar o nome de colunas ou linhas: usar a função rename
•df.rename(columns={'pop':'population'}, inplace=True)

Transformar linhas em colunas e vice versa:
•DataFrame.transpose( )

Ordenar linhas de uma Serie:
•Series.sort_index(ascending=’True ou False’)

O loc permite fazermos seleções de um DataFrame usando os nomes de linhas e colunas.
Ex:
*dados.loc[:, "IDADE”] (*todas as linhas, só a coluna* IDADE *)*
Podemos usar listas para selecionar linhas ou colunas. Ex:
•linhas = [1, 3, 17]
•colunas = ["RENDA", "EDUCACAO”]
•dados.loc[linhas, colunas]

É possível fazer slicing com DataFrames também. Ex:
•dados[["EDUCACAO", "RENDA”]][1:5]
Para o slicing de linhas e colunas:
•dados.iloc[2:5,3:8]

Para ordenar os valores, usar sort_values. Ex:
•dados.sort_values(by=“EDUCACAO”)
Para contar valores:
•dados.EDUCACAO.value_counts( )
E para transformar em percentual:
•dados.EDUCACAO.value_counts(True) * 100

Definindo dado como categórico:
•dados.EDUCACAO = dados.EDUCACAO.astype(‘category’)
Substituindo os valores categóricos - Ex:
•dados.EDUCACAO.cat.categories = ('Analfabeto', '1o. Grau', '2o. Grau', 'Graduacao', 'Pos-Graduacao’)

Tabela cruzada. Primeiro dado será linha, segundo será coluna. Ex:
pd.crosstab(dados.EDUCACAO, dados.DEFAULT)

Dividir uma série de informações em um ponto definido (ex: o ponto em que divide 50%). Sintaxe:
•DataFrame.quantile(q = *porcentagem que vc quer [de 0 a 1]*, interpolation=‘*tipo de interpolação [linear, higher, lower, nearest*')

O pd.cut( ) serve para separar informações em faixas de valor. O resultado do pd.cut é uma série. Sintaxe:
•pd.cut(dados[‘Population’], bins = *’número de faixas que você quer que ele produza OU as faixas de corte —> [0, 50, 100, 200, 300]’*)

Gráficos

Plotando gráficos de barra:
•dados.plot (kind='bar', color=('blue', 'red'), legend=False)

E gráficos de pizza (obs - autopct deixa as porcentagens no gráfico):
•dados.plot(kind='pie', colors=('blue', 'red'), autopct="%0.2f”, legend=False)

Gráfico de dispersão:
•df.plot.scatter

Para plotar histogramas, usar:
•dados.plot.hist(density = True, bins = ‘*numero de divisões’*)
Ou ainda
plt.hist(dados, density=True, bins= ‘*numero de divisões’*)

E boxplots:
•plt.boxplot(dados)
Ou podemos usar a função Boxplots do pandas:
•dados.boxplot(column = *’nome da coluna cujos valores estarão no boxplot’* , by = *‘parâmetro do eixo X (ex: se colocar aqui Sexo, vai plotar um Boxplots pro masculino e outro pro feminino’*)

- Probabilidade

Média, mediana e moda se fala mean, median e mode.

Cálculo de correlações:
•df[‘nome_da_coluna’].corr(df[‘nome_da_coluna’])

Cálculo de covariância:
•df[‘nome_da_coluna’].cov(df[’nome_da_coluna’]
Cálculo de variância:
df[‘nome_da_coluna’].var(ddof=0)
Se o número de amostras for muito pequeno, usar ddof=1

Checar se os dados se encaixam em alguma distribuição:
•stats.probplot(valores_da_funcao, dist=‘(nome do tipo de função; ex: expon,binom,poisson,norm,uniform)’ , plot=plt)

Distribuição binomial

•stats.binom.pmf(y,n,p): $P(Y = y)$

Probabilidade de dar **y** número de sucessos, em **n** tentativas e sendo a prob de sucesso **p**

•stats.binom.cdf(y,n,p): $P(Y \leq y)$

Probabilidade do número de vezes do evento dar menor ou igual a **y** vezes, em **n** tentativas com **p** chance de sucesso

•m, v = stats.binom.stats(n, p, moments='mv')

Cálculo do valor esperado e da variância num experimento binomial

Distribuição de Poisson

•stats.poisson.pmf(y,mu): $P(Y = y)$

Número de ocorrências de **y** evento, tendo **mu** como a média de ocorrência do evento em um intervalo

•stats.poisson.cdf(y,mu): $P(Y \leq y)$

Número de ocorrências do evento menor ou igual que **y**, tendo **mu** como a média de ocorrência do evento em um intervalo

Distribuição Exponencial

•stats.expon.pdf(f(x),scale=beta): $P(Y = y)$

f(x) é o linspace criado, que representa a faixa de eventos a ser analisada, e o **beta** é o tempo esperado até a ocorrência

•stats.expon.cdf(f(x),scale=beta): $P(Y \leq y)$

f(x) é o linspace criado, que representa a faixa de eventos a ser analisada, e o **beta** é o tempo esperado até a ocorrência

Distribuição Normal

•stats.norm.cdf(x, loc=mu, scale=sigma)

x é o valor desejado, **mu** recebe o valor da média e **sigma** o desvio padrão. Isso devolve uma área até **x** (a probabilidade de ser ≤ x)

•stats.norm.ppf(area,loc=mu,scale=sigma)

area é a probabilidade dada, **mu** recebe o valor da média e **sigma** o desvio padrão. Isso devolve um ponto (o que define essa área)

•norm.pdf(f(x),loc=mu,scale=sigma)

f(x) é o linspace criado, que representa a faixa de eventos a ser analisada, **loc** recebe **mu** (média) e **scale** recebe **sigma** (desvio padrão)

•distribuicao.fit(dados)
Dada uma série de dados, essa função devolve a média e o desvio padrão baseado no tipo de distribuição definido

*pdf = probability density function
Para V.A. contínua
*cdf = cumulative density function
Para V.A. contínua ou discreta
*pmf = probability mass function
Para V.A. discreta
*ppf = percent point function
Inverso da cdf (se é dada uma probabilidade, ele te dá o valor)
(no caso da distribuição normal, você dá a área e é devolvido o ponto que divide essa área)

Ajuste de retas em uma linha:
•import seaborn as sns
•sns.regplot(dados['RENDA'], dados['NOTA_ENEM'], line_kws={'color':'red'})