# Guia de estatística aplicada com Python

## 1. Ambiente de Desenvolvimento Google Colaboratory (Colab)

**Elaborado por Matheus Sanclé**

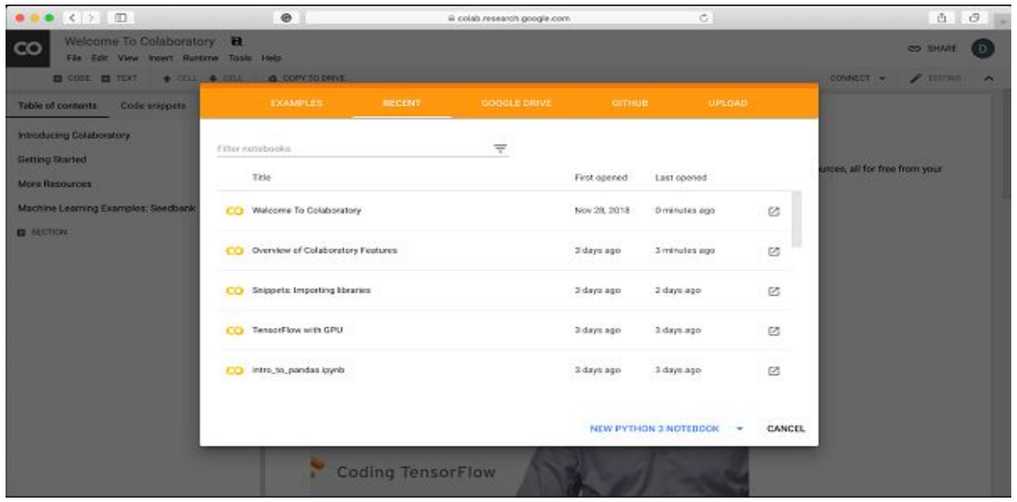
### 1.1. Sobre:

O Colab permite que qualquer pessoa escreva e execute código python arbitrário através do navegador e é especialmente adequado para aprendizado de máquina, análise de dados e educação, o Colab permite que você use e compartilhe os notebooks Jupyter com outras pessoas sem precisar baixar, instalar ou executar nada.

. Mais tecnicamente, O Projeto Jupyter por sua vez é uma organização sem fins lucrativos criada para "desenvolver software de código aberto, padrões abertos e serviços para computação interativa em dezenas de linguagens de programação". Originado do IPython (interpretador interativo focado no Python) em 2014, o Projeto Jupyter suporta ambientes de execução em dezenas de linguagens de programação. Basicamente o Jupyter é o projeto de código aberto no qual o Colab se baseia

### 1.2. Primeiro Documento no Colab:

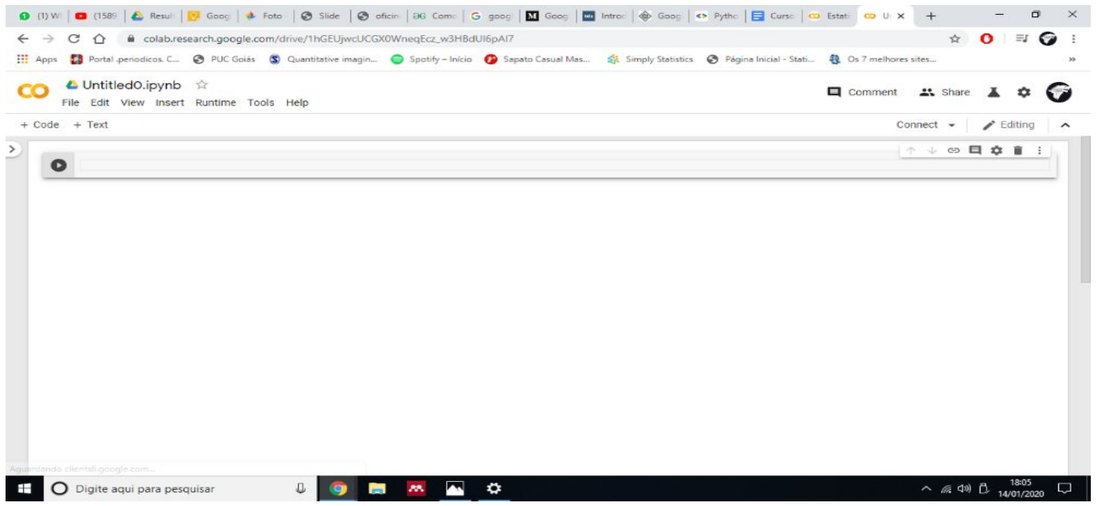
Para começar a trabalhar com a Colab, primeiro você precisa fazer login na sua conta do Google e, em seguida, acessar este link https://colab.research.google.com . Seu navegador exibirá a seguinte tela (supondo que você esteja conectado ao seu Google Drive).




As Guias apresentadas oferecem as seguintes funcionalidades:

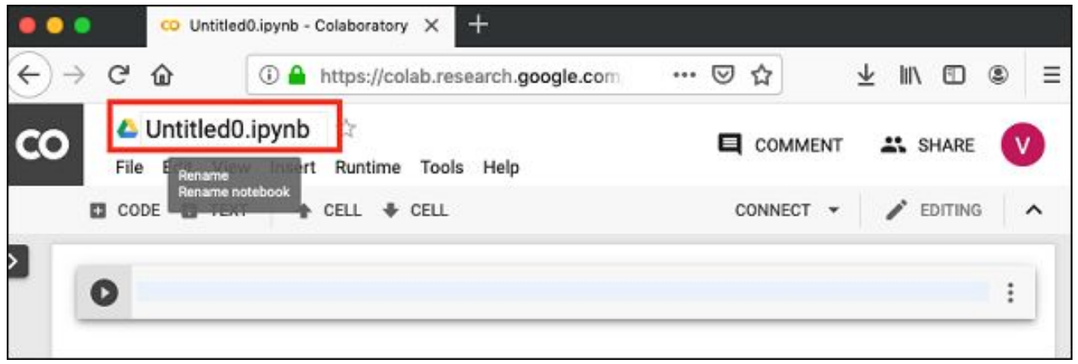
- *EXAMPLES:* Contém vários cadernos Jupyter de vários exemplos.
- *RECENT:* Caderno Jupyter com o qual você trabalhou recentemente.
- *GOOGLE DRIVE:* notebook Jupyter no seu Google Drive.
- *GITHUB:* Você pode adicionar o notebook Jupyter a partir do seu GitHub, mas primeiro você precisa conectar o Colab ao GitHub.
- *UPLOAD:* Faça o upload do seu diretório local.

E têm-se claro a opção de iniciar um novo documento a qualquer momento, clicando em "*NEW PYTHON 3 NOTEBOOK*". Há uma janela de código na qual você irá digitar seu código Python:



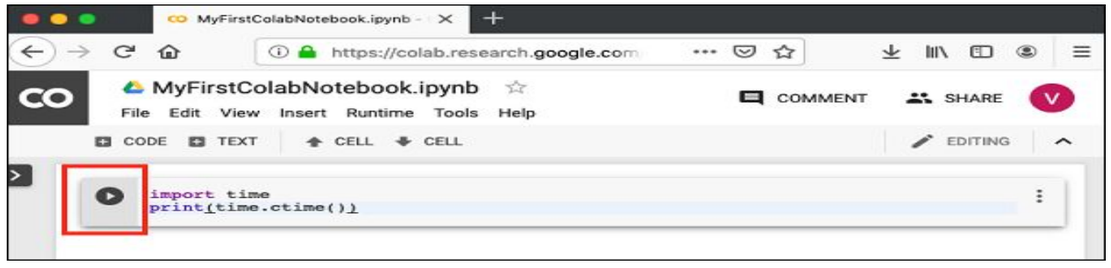
## 1.3. Renomeando o arquivo:

Por padrão, o notebook usa a convenção de nomenclatura *UntitledXX.ipynb* (a extensão .ipynb corresponde a extensão de um notebook iPython). Para renomear o bloco de anotações, clique no nome e digite a alteração desejada na caixa de edição, como mostrado:

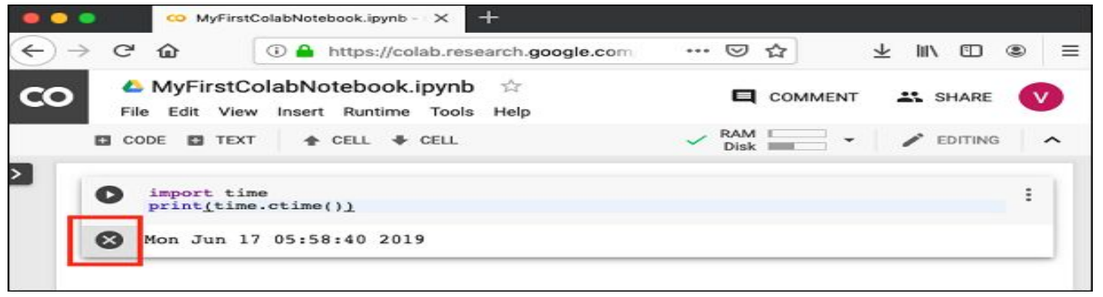



## 1.4. Introdução e Execução de código:

Na janela de código exibida, poderá ser inserido seu código em Python, e você pode executá-lo clicando na seta do lado esquerdo da janela de código, como no exemplo:

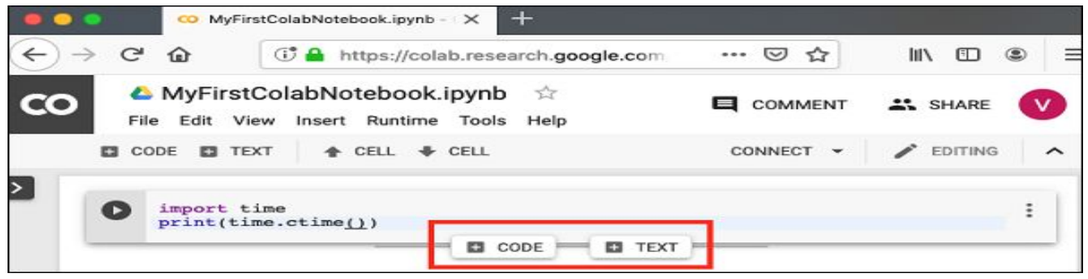



Após um tempo a execução será finalizada, você verá a saída do código executado logo abaixo, podendo limpar a mesma a qualquer momento, apenas clicando no ícone de exclusão também ao lado esquerdo da janela de código:




## 1.5. Adicionando janela de código:

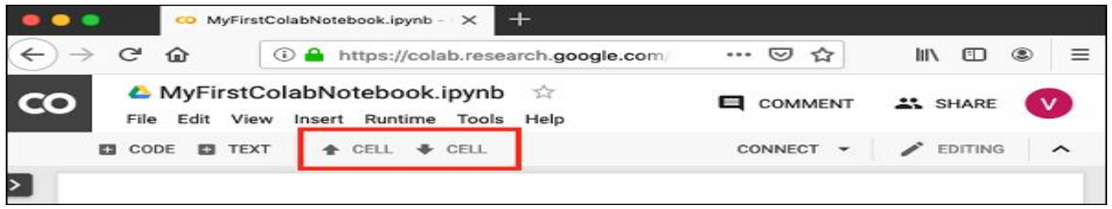
Sempre que necessário, novas janelas de código com seus blocos de execução específicos podem ser criadas, basta passar o mouse no centro da janela do código e quando os botões *CODE* e *TEXT* aparecerem, clique em *CODE* para adicionar uma nova célula. Outra alternativa, é selecionar diretamente do menu abaixo do nome do arquivo, os mesmos botões.



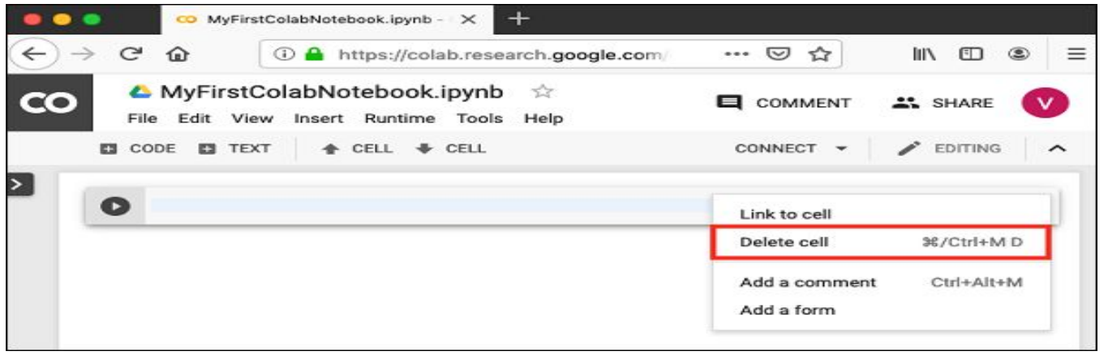

Com mais de uma janela de código em seu documento, você pode vir a querer mudar a ordem ou a forma de execução das mesmas, essa opção pode ser alterada selecionando a forma de execução, na aba "*Runtime*".

## 1.6. Alterando a ordem e excluindo janelas de código:

Quando o seu notebook contém um grande número de células de código, você pode encontrar situações nas quais deseja alterar a ordem de execução dessas células. Você pode fazer isso selecionando a célula que deseja mover e clicando nos botões *UP CELL* ou *DOWN CELL*, podendo clicar várias vezes para mover a janela para mais de uma única posição:

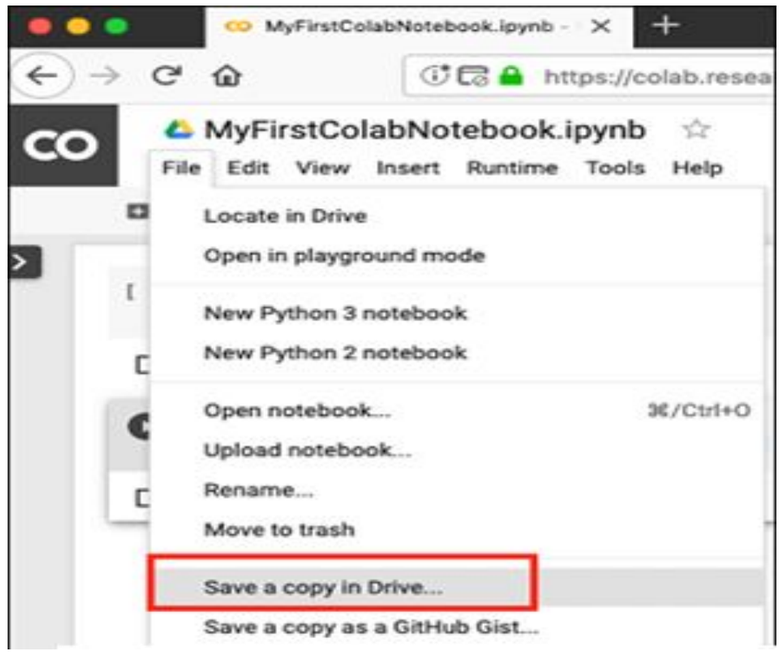



Durante o desenvolvimento do seu projeto, você pode ter introduzido algumas janelas agora indesejadas em seu notebook. Você pode remover essas células do seu projeto facilmente com um único clicando no ícone pontilhado vertical no canto superior direito da sua célula de código, e em seguida selecionando “Excluir célula”:



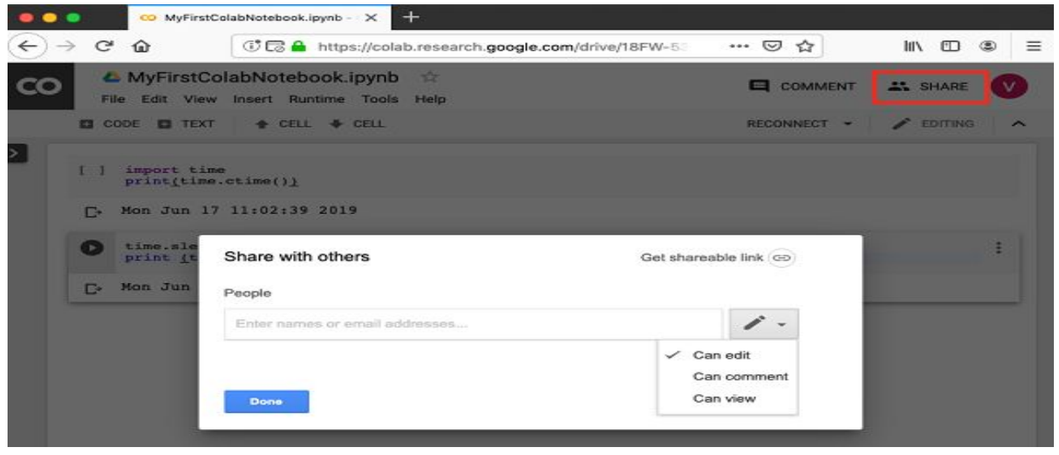

## 1.7. Salvando seu Documento no Google Drive:

O Colab permite salvar seu trabalho no Google Drive de maneira bem simples, a ação criará uma cópia do seu notebook e salvará posteriormente na sua unidade, podendo renomear a cópia para sua escolha de nome:




## 1.8. Compartilhando um Documento:

Para compartilhar o bloco de anotações que você criou com outros co-desenvolvedores, compartilhe a cópia que você fez no seu Google Drive. Há mais uma maneira de compartilhar seu trabalho, clicando no link *SHARE* no canto superior direito do seu notebook Colab. Isso abrirá a caixa de compartilhamento:



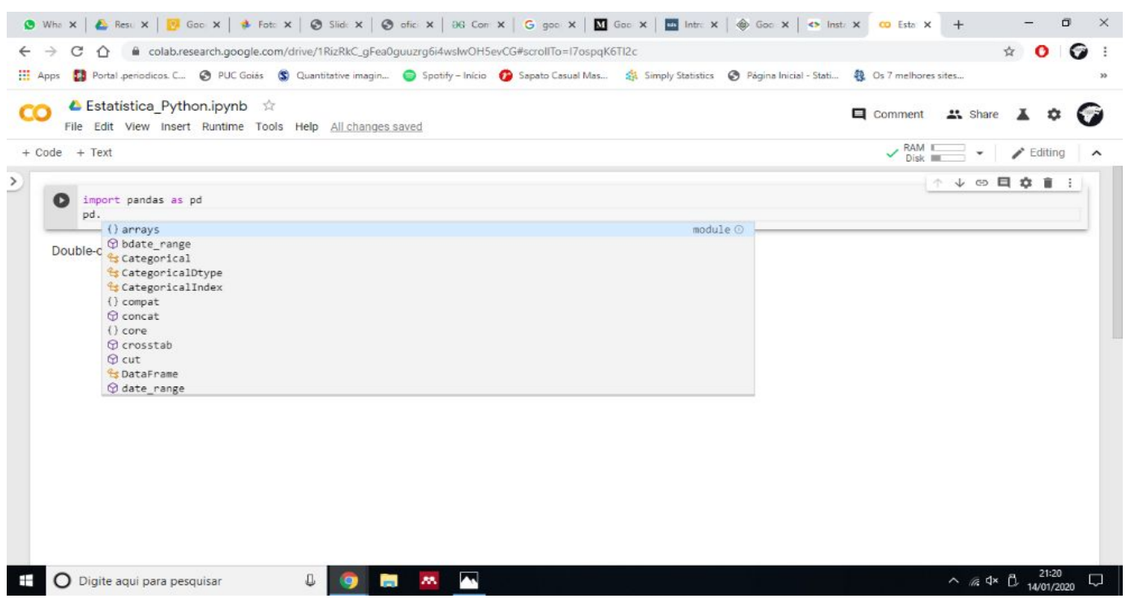
Você pode inserir os *IDs* de e-mail das pessoas com quem você gostaria de compartilhar o documento atual. Você pode definir o tipo de acesso selecionando entre as três opções mostradas na tela acima. Clique na opção Obter link compartilhável para obter o URL do seu notebook. Você encontrará opções para quem compartilhar da seguinte maneira:

- Grupo de pessoas especificado;
- Colegas na sua organização;
- Qualquer pessoa com o link;
- Todo o público na web.

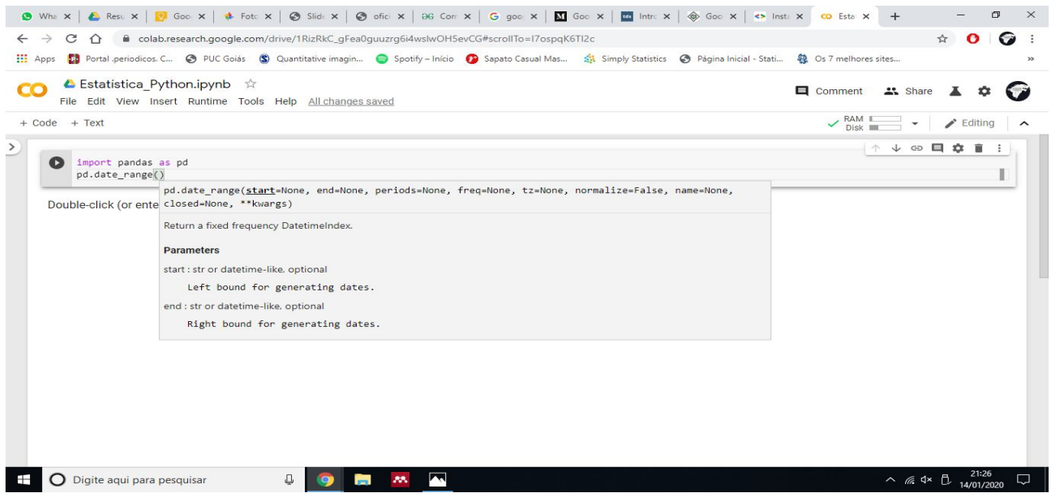
E é exatamente esta opção que foi utilizada para permitir o acesso ao arquivo de execução principal do Curso Estatístico, permitindo a edição, comentários e execução.

## 1.9. Lista de Funções e Documentação:

Atualmente, os desenvolvedores dependem muito da ajuda sensível ao contexto para as sintaxes da linguagem e da biblioteca. É por isso que os *IDEs* são amplamente utilizados. O editor de notebook da Colab fornece esse recurso. Vamos ver como solicitar ajuda sensível ao contexto ao escrever o código Python no Colab. Você pode solicitar ajuda sensível ao contexto sobre nomes de funções pressionando a tecla *TAB* . Observe a presença do *DOT* após a palavra-chave da tocha . Sem esse *DOT*( “.”), você não verá a ajuda do contexto:

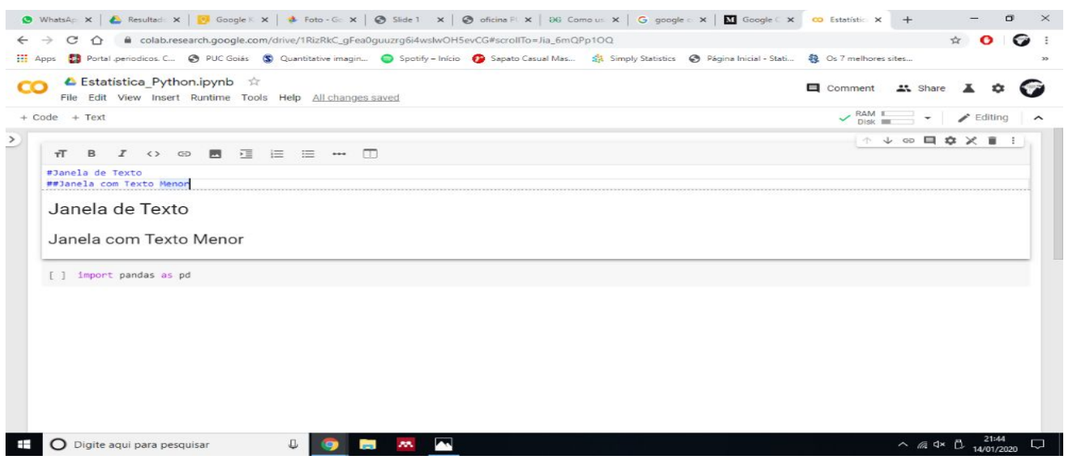



O Colab também fornece a documentação de qualquer função ou classe como uma ajuda sensível ao contexto. Agora, pressione *TAB* e você verá a documentação sobre cos na janela pop-up, como mostrado na imagem aqui. Observe que você precisa digitar parênteses abertos antes de pressionar *TAB*.




## 1.10. Adição de Texto:

Há ainda opção de incluir janelas de texto no seu documento do Google Colab, com o objetivo de organizar e estruturar ainda mais as seções e trechos de códigos. O comando de inclusão é similar ao de inclusão de janelas de código, no entanto deve-se clicar no botão *“TEXT”*. Ainda pode-se aumentar ou diminuir o tamanho da fonte, bem como alterar sua cor, mas essas são opções para um outro momento.



# 2. Bibliotecas python

**Elaborado por Matheus Sanclé/Joás Rodrigues.**

## 2.1 Numpy

Tradicionalmente, começamos nossa lista com as bibliotecas para aplicativos científicos e o NumPy é um dos principais pacotes nessa área. Ele é destinado ao processamento de grandes matrizes e matrizes multidimensionais, e uma extensa coleção de funções matemáticas de alto nível e métodos implementados possibilitam a execução de várias operações com esses objetos.

Durante o último ano, um grande número de melhorias foi feito na biblioteca. Além das correções de bugs e problemas de compatibilidade, as mudanças cruciais dizem respeito às possibilidades de estilo, ou seja, o formato de impressão dos objetos NumPy. Além disso, algumas funções agora podem manipular arquivos de qualquer codificação disponível no Python.

Todos os métodos que iremos utilizar dessa biblioteca operam sobre uma variável aleatória. Criemos então um vetor com valores aleatórios:

X = np.random.rand(1, 5)

Os principais métodos dessa biblioteca que iremos utilizar são:

| ***Comandos*** | **descrição** |
|---|---|
| *X = np.random.rand(linhas, colunas)*<br><br>*X = np.random.rand(1, 5)* | Cria uma matriz de valores aleatórios<br><br>Cria um vetor com dimensão 1x5 com valores aleatórios |
| *np.mediam(X)* | Calcula a mediana dos valores contidos no vetor 'X' . O único parâmetro a se passar é o vetor. |
| *np.mean(X)* | Calcula a média aritmética dos valores contidos no vetor X. |
| *np.std(X)* | Calcula o desvio padrão. |
| | |
| *np.quantile(X, 0.5)* | |
| *np.percentile(X, 0.5)* | |
| *np.var(X)* | Calcula a variância do conjunto de dados |
| *np.random.normal()* | Cria uma distribuição normal de numeros e devolve-os como numpy array. |
| *np.random.exponential()* | Cria uma distribuição exponencial de numeros e devolve-os como numpy array. |
| *np.random.uniform()* | Cria uma distribuição uniforme de numeros e devolve-os como numpy array. |
| *np.random.binomial()* | Cria uma distribuição binomial de numeros e devolve-os como numpy array. |

A seguir damos um exemplo do uso de todas os métodos acima, com base em um vetor criado com a função random do módulo nunpy.

```
import numpy as np
import pandas as pd
import statistics as st

#criação de array com numeros aleatorios
x = np.random.rand(1, 5)

#calculo da mediana
print("a mediana é: ", np.median(x))
#cálculo da média aritimética
print("media aritmética: ", np.mean(x))
#cálculo do desvio padrão
print("desvio padrão: ", np.std(x))

#cálculo do 1º quartil 0 - 0.25
print("1º quartil: ", np.quantile(x, 0.25))
#cálculo do 2º quartil 0.25 - 0.5
```

```
print("2º quartil: ", np.quantile(x, 0.5))
#cálculo do 3º quartil 0.5 - 0.75
print("3º quartil: ", np.quantile(x, 0.75))


#cálculo do 1º percentil 0 - 0.25
print("1º percentil: ", np.percentile(x, 0.25))
#cálculo do 2º percentil 0.25 - 0.5
print("2º percentil: ", np.percentile(x, 0.5))
#cálculo do 3º percentil 0.5 - 0.75
print("3º percentil: ", np.percentile(x, 0.75))


#distribuições
media = 100
desvio_padrao = 10
tamanho = 1000

dist_norm = np.random.normal(media, desvio_padrao, tamanho)
dist_exp = np.random.exponential(0.1, tamanho)
dist_uni = np.random.uniform(0, st.math.sqrt(tamanho),  tamanho)
dist_bin = np.random.binomial(50, 0.7, tamanho)

dist = {'Normal': dist_norm,
        'Exponencial': dist_exp,
        'Uniforme': dist_uni,
        'Binomial': dist_bin}

df_dist = pd.DataFrame(dist)
print("Conjunto de Dados do DataFrame: \n")
df_dist
```

Para ter um aprofundamento na biblioteca ***numpy***, sugere-se consultar a sua documentação através dos links abaixo:

https://docs.scipy.org/doc/numpy/index.html

## 2.2 Statistics

Este módulo fornece funções para o cálculo de estatísticas matemáticas de dados numéricos (com valor real).

| ***Comandos*** | ***Descrição*** |
|---|---|
| st.mean(X) | Calcula a média aritmética dos valores do vetor X |
| st.median(X) | Calcula a mediana dos valores do vetor X |
| st.mode(X) | Calcula a moda estatística dos valores do vetor x |
| st.pstdev(X) | Calcula o desvio padrão populacional dos valores do vetor X |
| st.pvariance(X) | Calcula a variância populacional sobre os valores da variável X |
| st.variance(X) | Calcula a variância dos valores da variável aleatória X |
| st.stdev(X) | Calcula o desvio padrão dos valores da variável aleatória X |

A seguir demostramos a utilização de todos estes métodos na plataforma *google colab*, veja:

```python
import statistics as st

# aqui criamos um conjunto de dados, um vetor que
# será utilizado para o cálculo das medidas estatísticas

x = [1.2, 5.7, 2.3, 3.9, 6.7, 1.2]

#média aritmética
print("média aritmética: ", st.mean(x))
# mediana
print("médiana: ", st.median(x))
# moda
print("moda: ", st.mode(x))
# desvio padrão populacional
print("desvio padrão populacional: ",st.pstdev(x))
# variância populacional
print("variância populacional: ", st.pvariance(x))
# desvio padrão
print("desvio padrão: ", st.variance(x))
```

Para ter um aprofundamento na biblioteca ***statistics***, sugere-se consultar sua documentação através dos links abaixo:

https://docs.python.org/3/library/statistics.html

QR Code

## 2.3 Seaborn e matplotlib

O Seaborn é essencialmente uma API de alto nível baseada na biblioteca matplotlib. Ele contém configurações padrão mais adequadas para o processamento de gráficos. Além disso, há uma rica galeria de visualizações, incluindo alguns tipos complexos, como séries temporais, diagramas conjuntos e diagramas de violino.

As atualizações recentes cobrem principalmente correções de bugs. No entanto, houve melhorias na compatibilidade entre FacetGrid ou PairGrid e backends de matplotlib interativos aprimorados, adicionando parâmetros e opções às visualizações.

Os principais métodos dessa biblioteca que iremos utilizar são:

| **Comandos** | **Biblioteca** | **Descrição** |
|---|---|---|
| *set_style()* | seaborn | Configura o estilo de fundo do gráfico |
| *distplot()* | seaborn | Plota um histograma do conjunto de dados passado como parâmetro. |
| *boxplot()* | seaborn | Plota um gráfico do tipo boxplot do conjunto de dados passado como parâmetro. |
| *suplots()* | matplotlib | Atua como um invólucro de utilitário e ajuda a criar layouts comuns de subtramas, incluindo o objeto de figura anexo, em uma única chamada. |
| *qqplot()* | statsmodel | Plota um gráfico quantil-quantil. |
| | | |

Abaixo damos um exemplo do uso de cada um dos gráficos.

```
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.graphics.gofplots as qq

# primeiro cria-se uma distribuição uniforme de numeros

media = 100
desvio_padrao = 10
tamanho = 1000
x = np.random.uniform(media, desvio_padrao, tamanho)

# Cria a Estrutura para plotagem dos gráficos (1x4)
fig, axes = plt.subplots(1, 4, figsize = (20, 7))

# Histograma
sns.set_style("whitegrid")
sns.distplot(x, color = 'blue', ax = axes[0], axlabel = 'Histograma')

# BoxPlot
bx = sns.boxplot(x,  orient = 'v', ax = axes[1])

# Gráfico Quantil-Quantil
fig = qq.qqplot(x, line = 'r', ax = axes[2])
```

```
# barplot
sns.set(style="whitegrid")
tips = sns.load_dataset("tips")
ax = sns.barplot(x="day", y="total_bill", ax = axes[3], data=tips)
```

Para ter um aprofundamento na biblioteca ***seaborn***, sugere-se consultar sua documentação através dos links abaixo:

https://seaborn.pydata.org/
https://matplotlib.org/contents.html

**QR Code**

## 2.4 Pandas

Pandas é uma biblioteca Python que fornece estruturas de dados de alto nível e uma grande variedade de ferramentas para análise. A grande característica deste pacote é a capacidade de traduzir operações bastante complexas com dados em um ou dois comandos. O Pandas contém muitos métodos internos para agrupar, filtrar e combinar dados, bem como a funcionalidade de séries temporais. Tudo isso é seguido por indicadores de velocidade impressionantes.

Houve alguns novos lançamentos recentes da biblioteca Pandas, incluindo centenas de novos recursos, aprimoramentos, correções de bugs e alterações de API. As melhorias consideram as habilidades do Pandas de agrupar e classificar dados, saída mais adequada para o método apply e o suporte na execução de operações de tipos personalizados.

Para um conhecimento aprofundado dessa biblioteca utilize os endereços abaixo:

Lembrando que importamos a biblioteca pandas usando o al

Os principais métodos dessa biblioteca que iremos utilizar são:

| Comandos | biblioteca | Descrição |
|---|---|---|
| *series()* | pandas | Cria uma matriz unidimensional com um índice. Como um vetor. |
| *DataFrame()* | pandas | Cria uma estrutura, como uma matriz, com índices e colunas. |
| *DataFrame(data).describe()* | pandas | Retorna a descrição das medidas estatísticas do conjunto de dados. |
| *DataFrame(data).sum()* | pandas | Retorna a soma de todos os valores contidos no DataFrame. |
| *DataFrame(data).index[n]* | pandas | Retorna o valor indexado pelo índice n. |

| | | |
|---|---|---|
| *DataFrame(data).count()* | pandas | Retorna a quantidade de elementos contidos no DataFrame. |

Como teste, digite o código abaixo na plataforma google colab e verifique.

```
import pandas as pd
import scipy.stats as sp

# Criação Manual de Dados
data = [10, 7, 67, 63, 45, 42, 34, 39, 25, 21, 18, 16, 60, 2, 52, 58, 29, 9, 3,
56]

sd = pd.Series(data)
print("- Conjunto de Dados:\n", data, "\n")

summary_pd = pd.DataFrame(data).describe()
print("- Sumário gerado a partir da Biblioteca Pandas: \n", summary_pd, "\n")
print(" apenas os indices do DataFrame: ", pd.DataFrame(summary_pd).index[1]);
print(" quantidade de valores no DataFrame: ", pd.DataFrame(summary_pd).count());

print(" soma dos valores do DataFrame: ", pd.DataFrame(summary_pd).sum())
```

Para ter um aprofundamento na biblioteca ***pandas***, sugere-se consultar sua documentação através dos links abaixo:

https://pandas.pydata.org/pandas-docs/stable/

QR Code: