

# Programação para Ciência de Dados



Roberto M. Cesar Jr. - [rmcesar@usp.br](mailto:rmcesar@usp.br)



$$f(x) = a_0 + \sum_{n=1}^{\infty} \left[ a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right) \right]$$

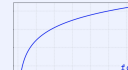
```

def factorial(n):
    if n <= 1:
        return 1
    else:
        return n * factorial(n - 1)
    
```



$$e^{i\pi} + 1 = 0$$

$$a^2 + b^2 = c^2$$



$$S_{\text{intra}} = \sum_{j=1}^k \sum_{x_i \in C_j} (x_i - \mu_j)(x_i - \mu_j)^T$$

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau) d\tau$$

```

for i in range(len(A)):
    for j in range(len(B[0])):
        for k in range(len(B)):
            C[i][j] += A[i][k] * B[k][j]
    
```



$$i\hbar \frac{\partial}{\partial t} \Psi(x, t) = \left[ -\frac{\hbar^2}{2m} \nabla^2 + V(x, t) \right] \Psi(x, t)$$



$$E = mc^2$$

```

for i in range(n-1):
    iMin = i
    for j in range(i+1, n):
        if v[j] < v[iMin]:
            iMin = j
    swapCases += 1
    v[i], v[iMin] = v[iMin], v[i]
    
```

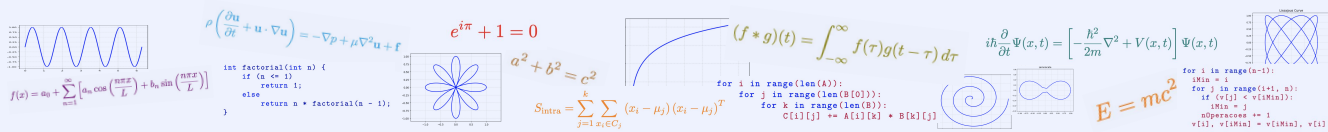


# Etapas típicas de um pipeline de CD

- Datasets
- Carregamento de dados (data\_loaders)
- Representação dos dados
- Tabela sumária de dados
- Manipulação de dados
- Filtragem de dados
- Visualização de dados

[https://github.com/rmcesarjr/iccd/blob/main/notebooks/data\\_science\\_medimento\\_pessoa.ipynb](https://github.com/rmcesarjr/iccd/blob/main/notebooks/data_science_medimento_pessoa.ipynb)

<https://www.youtube.com/watch?v=Rvjpa94DoEU>



# Datasets

rmcesarjr / iccd

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

Files

main

+

Q

Go to file

data

.gitkeep

BRICS\_EmissaoDeCO2PerCapita...

BRICS\_ExpectativaDeVida\_1960...

BRICS\_PIBPerCapita.csv

altura\_peso\_distribuicao.csv

ano-pib-pibpc-ipc.csv

clima\_mensal.csv

**dados\_pressao\_local.csv**

filmes.csv

populacao\_50\_cidades\_brasil.csv

pressao\_ensaio\_2021\_sis\_dia\_r...

pressao\_pacientes.csv

primeiraPlanilhaX.csv

primeiraPlanilhaY.csv

reservas.csv

times.csv

iccd

notebooks

LICENSE

README.md

setup.py

iccd / data / dados\_pressao\_local.csv

rmcesarjr Add files via upload

Preview

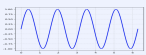
Code

Blame

2181 lines (2181 loc) · 71.2 KB

Search this file

	ID_paciente	Data	Sistolica	Diastolica	Glicemia	Grupo
1						
2	1	19/03/2021	122	440	108	Placebo
3	1	20/03/2021	128	85	108	Placebo
4	1	21/03/2021	122	85	111	Placebo
5	1	22/03/2021	620	89	107	Placebo
6	1	23/03/2021	123	81	107	Placebo
7	1	24/03/2021	116	78	109	Placebo
8	1	25/03/2021	130	77	106	Placebo
9	2	21/12/2021	133	81	114	Placebo
10	2	22/12/2021	620	87	114	Placebo
11	2	23/12/2021	122	80	-40	Placebo
12	2	24/12/2021	125	89	106	Placebo
13	2	25/12/2021	121	81	111	Placebo
14	2	26/12/2021	127	89	110	Placebo
15	2	27/12/2021	123	85	108	Placebo
16	3	20/04/2021	128	84	111	Placebo
17	3	21/04/2021	133	85	116	Placebo
18	3	22/04/2021	130	78	-40	Placebo
19	3	23/04/2021	129	87	114	Placebo


$$f(x) = a_0 + \sum_{n=1}^{\infty} \left[ a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right) \right]$$

$$\rho \left( \frac{\partial u}{\partial t} + u \cdot \nabla u \right) = -\nabla p + \mu \nabla^2 u + f$$

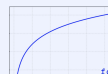
```
int factorial(int n) {
    if (n <= 1)
        return 1;
    else
        return n * factorial(n - 1);
}
```



$$e^{i\pi} + 1 = 0$$

$$a^2 + b^2 = c^2$$

$$S_{\text{intra}} = \sum_{j=1}^k \sum_{x_i \in C_j} (x_i - \mu_j)(x_i - \mu_j)^T$$

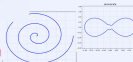


$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau) d\tau$$

```
for i in range(len(A)):
    for j in range(len(B[0])):
        for k in range(len(B)):
            C[i][j] += A[i][k] * B[k][j]
```



$$i\hbar \frac{\partial}{\partial t} \Psi(x, t) = \left[ -\frac{\hbar^2}{2m} \nabla^2 + V(x, t) \right] \Psi(x, t)$$



$$E = mc^2$$

```
for i in range(n-1):
    min = i
    for j in range(i+1, n):
        if v[i] < v[min]:
            min = j
    if min != i:
        v[i], v[min] = v[min], v[i]
```

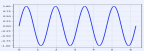


# Carregamento de dados (data\_loaders)

```
RAW_URL = "https://raw.githubusercontent.com/rmcesarjr/iccd/main/data/dados_pressao_local.csv"  
#RAW_URL = None # util para ler do proprio hd local
```

```
VERBOSE = False
```

```
def carrega_dados():  
    if RAW_URL:  
        df = pd.read_csv(RAW_URL)  
    else:  
        uploaded = files.upload()  
        nome = next(iter(uploaded))  
        df = pd.read_csv(io.BytesIO(uploaded[nome]))  
  
    return df
```


$$f(x) = a_0 + \sum_{n=1}^{\infty} \left[ a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right) \right]$$

$$\rho \left( \frac{\partial u}{\partial t} + u \cdot \nabla u \right) = -\nabla p + \mu \nabla^2 u + f$$

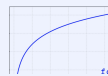
```
def factorial(n):  
    if n <= 1:  
        return 1  
    else:  
        return n * factorial(n - 1)
```



$$e^{i\pi} + 1 = 0$$

$$a^2 + b^2 = c^2$$

$$S_{\text{intra}} = \sum_{j=1}^k \sum_{x_i \in C_j} (x_i - \mu_j)(x_i - \mu_j)^T$$



$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau) d\tau$$

```
for i in range(len(A)):  
    for j in range(len(B[0])):  
        for k in range(len(B)):  
            C[i][j] += A[i][k] * B[k][j]
```



$$i\hbar \frac{\partial}{\partial t} \Psi(x, t) = \left[ -\frac{\hbar^2}{2m} \nabla^2 + V(x, t) \right] \Psi(x, t)$$

$$E = mc^2$$

```
for i in range(n-1):  
    min = 1  
    for j in range(i+1, n):  
        if v[i] < v[min]:  
            min = j  
    v[i], v[min] = v[min], v[i]
```



# Representação dos dados

```
def carrega_dados():
    if RAW_URL:
        df = pd.read_csv(RAW_URL)
    else:
        uploaded = files.upload()
        nome = next(iter(uploaded))
        df = pd.read_csv(io.BytesIO(uploaded[nome]))

    ids = df["ID_paciente"].tolist()
    datas = df["Data"].tolist()
    sist = df["Sistolica"].tolist()
    dias = df["Diastolica"].tolist()
    glic = df["Glicemia"].tolist()
    grupos = df["Grupo"].tolist()

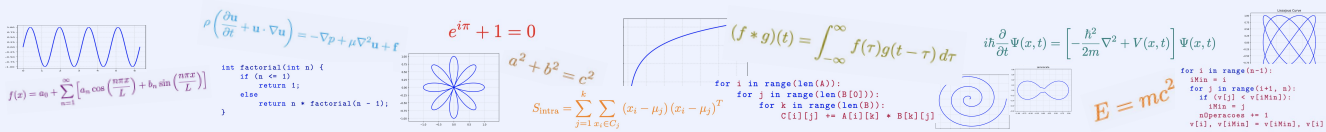
    return ids, datas, sist, dias, glic, grupos
```

```
def carrega_dados():
    if RAW_URL:
        df = pd.read_csv(RAW_URL)
    else:
        uploaded = files.upload()
        nome = next(iter(uploaded))
        df = pd.read_csv(io.BytesIO(uploaded[nome]))

    return df
```

Data frames

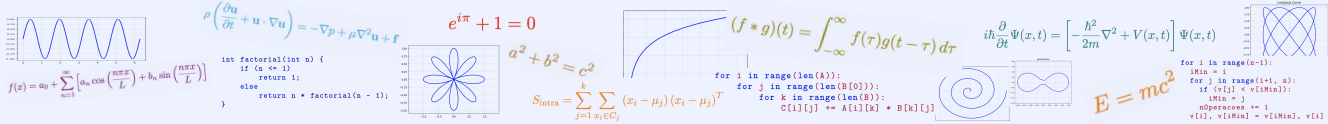
Listas



# Tabela sumária de dados

Resumo estatístico por grupo (inclui #Pacientes distintos):

Grupo	#Pacientes	(Sistolica, count)	(Sistolica, mean)	(Sistolica, min)	(Sistolica, max)	(Diastolica, count)	(Diastolica, mean)	(Diastolica, min)	(Diastolica, max)	(Glicemia, count)	(Glicemia, mean)	(Glicemia, min)	(Glicemia, max)
Medicamento	100	700	121.4	-40	620	700	78.2	-20	440	700	99.3	-40	450
Nada	100	700	135.7	-40	620	700	91.6	-20	440	700	112.4	-40	450
Placebo	100	700	132.3	-40	620	700	90.5	-20	440	700	114.8	-40	450



# Manipulação de dados: controle de qualidade

Ruídos, erros, outliers

27	4	05/10/2021	129	86	-40	Placebo
28	4	06/10/2021	125	85	102	Placebo
29	4	07/10/2021	116	84	111	Placebo
30	5	13/09/2021	130	84	109	Placebo
31	5	14/09/2021	119	87	108	Placebo
32	5	15/09/2021	118	87	113	Placebo
33	5	16/09/2021	124	87	111	Placebo
34	5	17/09/2021	-40	93	111	Placebo
35	5	18/09/2021	127	82	111	Placebo
36	5	19/09/2021	121	88	116	Placebo
37	6	20/01/2021	126	81	107	Placebo
38	6	21/01/2021	128	79	120	Placebo
39	6	22/01/2021	126	85	114	Placebo
40	6	23/01/2021	620	83	118	Placebo
41	6	24/01/2021	120	84	108	Placebo
42	6	25/01/2021	133	93	113	Placebo



$$f(x) = a_0 + \sum_{n=1}^{\infty} \left[ a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right) \right]$$

$$\rho \left( \frac{\partial u}{\partial t} + u \cdot \nabla u \right) = -\nabla p + \mu \nabla^2 u + f$$

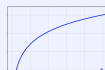
```
int factorial(int n) {  
  if (n <= 1)  
    return 1;  
  else  
    return n * factorial(n - 1);  
}
```



$$e^{i\pi} + 1 = 0$$

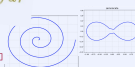
$$a^2 + b^2 = c^2$$

$$S_{\text{intra}} = \sum_{j=1}^k \sum_{x_i \in C_j} (x_i - \mu_j)(x_i - \mu_j)^T$$



$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau) d\tau$$

```
for i in range(len(A)):  
  for j in range(len(B[0])):  
    for k in range(len(B)):  
      C[i][j] += A[i][k] * B[k][j]
```



$$i\hbar \frac{\partial}{\partial t} \Psi(x, t) = \left[ -\frac{\hbar^2}{2m} \nabla^2 + V(x, t) \right] \Psi(x, t)$$

$$E = mc^2$$

```
for i in range(n-1):  
  for j in range(i+1, n):  
    if v[i] < v[j]:  
      v[i], v[j] = v[j], v[i]
```



# Controle de qualidade: filtragem de dados

```
def controle_qualidade(df, max_preview=10):
    ids = df["ID_paciente"].tolist()
    datas = df["Data"].tolist()
    sistolica = df["Sistolica"].tolist()
    diastolica = df["Diastolica"].tolist()
    glicemia = df["Glicemia"].tolist()
    grupos = df["Grupo"].tolist()

    n = len(ids)
    ids_ok, datas_ok, sist_ok, dia_ok, gli_ok, grupos_ok = [], [], [], [], [], []
    n_bad = 0

    print("=== CONTROLE DE QUALIDADE (data) ===")

    for i in range(n):
        idp = ids[i]
        data = datas[i]
        sis = sistolica[i]
        dia = diastolica[i]
        gli = glicemia[i]
        grp = grupos[i]

        ok, motivo = criterios_qualidade(data, sis, dia, gli)
        if ok:
            # mantém a amostra
            ids_ok.append(idp)
            datas_ok.append(data)
            sist_ok.append(sis)
            dia_ok.append(dia)
            gli_ok.append(gli)
            grupos_ok.append(grp)
        else:
            if VERBOSE: print(f"Data invalida excluída da amostra: paciente {idp} motivo {motivo}")
            n_bad+=1

    print(f"Total de amostras: {n}")
    print(f"Removidas: {n_bad} ({(n_bad/n*100):.2f}%)")


    return cria_df(ids_ok, datas_ok, sist_ok, dia_ok, gli_ok, grupos_ok)
```

```
def criterios_qualidade(data_str, sis, dia, gli, ano_valido=2021):
    dia_min = 30
    dia_max = 220
    sis_min = 50
    sis_max = 300
    gli_min = 60
    gli_max = 200

    s = str(data_str).strip()
    partes = s.split("/")
    dd_s, mm_s, aa_s = partes
    dd = int(dd_s)
    mm = int(mm_s)
    aa = int(aa_s)


    if not (1 <= dd <= 31):
        return False, "Dia fora de [1,31]"
    if not (1 <= mm <= 12):
        return False, "Mês fora de [1,12]"
    if aa != ano_valido:
        return False, f"Ano != {ano_valido}"
    if not (sis_min <= sis <= sis_max):
        return False, f"Nao verificado {sis_min} ≤ Sistolica ≤ {sis_max}"
    if not (dia_min <= dia <= dia_max):
        return False, f"Nao verificado {dia_min} ≤ Diastolica ≤ {dia_max}"
    if not (gli_min <= gli <= gli_max):
        return False, f"Nao verificado {gli_min} ≤ Glicemia ≤ {gli_max}"


    return True, None
```



$$f(x) = a_0 + \sum_{n=1}^{\infty} \left[ a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right) \right]$$


$$\rho \left( \frac{\partial u}{\partial t} + u \cdot \nabla u \right) = -\nabla p + \mu \nabla^2 u + f$$

```
def factorial(n):
    if n <= 1:
        return 1
    else:
        return n * factorial(n - 1)
```

$$e^{i\pi} + 1 = 0$$

$$a^2 + b^2 = c^2$$

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau) d\tau$$
$$S_{\text{intra}} = \sum_{j=1}^k \sum_{x_i \in C_j} (x_i - \mu_j)(x_i - \mu_j)^T$$


$$i\hbar \frac{\partial}{\partial t} \Psi(x,t) = \left[ -\frac{\hbar^2}{2m} \nabla^2 + V(x,t) \right] \Psi(x,t)$$

$$E = mc^2$$



```
for j in range(n-1):
    min = 1
    for i in range(1, n):
        if v[i] < v[min]:
            min = i
    v[i], v[min] = v[min], v[i]
```

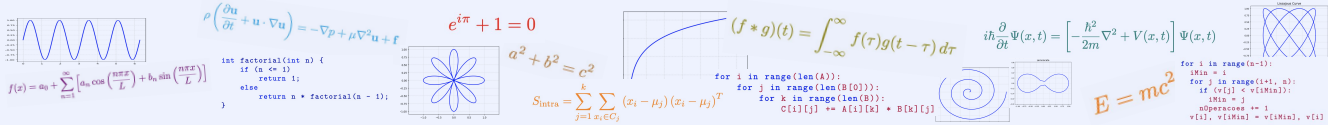


# Controle de qualidade: filtragem de dados

=== CONTROLE DE QUALIDADE (data) ===  
Total de amostras: 2100  
Removidas: 397 (18.90%)

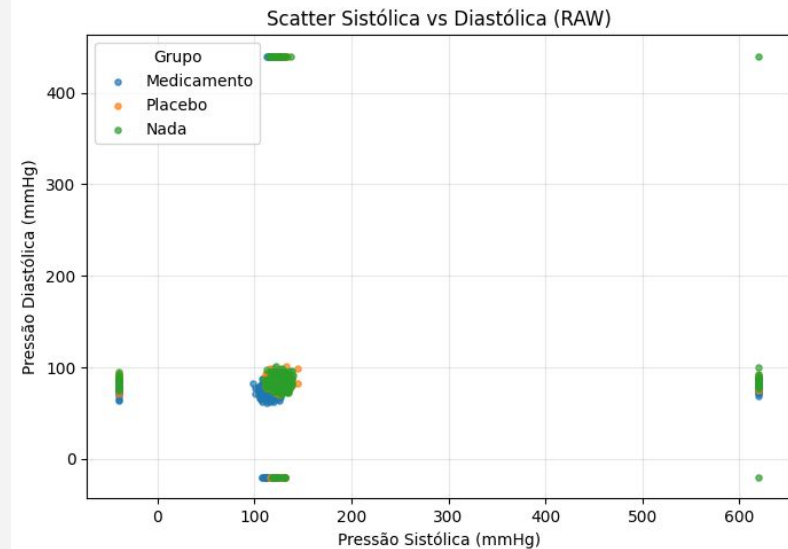
Resumo estatístico por grupo (inclui #Pacientes distintos):

Grupo	#Pacientes	(Sistolica, count)	(Sistolica, mean)	(Sistolica, min)	(Sistolica, max)	(Diastolica, count)	(Diastolica, mean)	(Diastolica, min)	(Diastolica, max)	(Glicemia, count)	(Glicemia, mean)	(Glicemia, min)	(Glicemia, max)
Medicamento	100	561	115.1	98	130	561	75.2	61	92	561	90.2	74	106
Nada	100	566	125.2	109	140	566	84.9	70	101	566	109.5	94	124
Placebo	100	576	125.4	111	145	576	85.0	72	101	576	110.4	95	125



# Visualização de dados

```
def visualiza_dados(df, titulo="Scatter Sistólica vs Diastólica por grupo (cor = Glicemia)":  
    plt.figure(figsize=figure_size())  
    for grupo in ["Medicamento", "Placebo", "Nada"]:  
        sub = df[df["Grupo"] == grupo]  
        plt.scatter(sub["Sistolica"], sub["Diastolica"], s=14, alpha=0.7, label=grupo)  
    plt.xlabel("Pressão Sistólica (mmHg)")  
    plt.ylabel("Pressão Diastólica (mmHg)")  
    plt.title(titulo)  
    plt.grid(True, alpha=0.3)  
    plt.legend(title="Grupo")  
    plt.tight_layout()  
    plt.show()
```



$$f(x) = a_0 + \sum_{n=1}^{\infty} \left[ a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right) \right]$$

$$\rho \left( \frac{\partial u}{\partial t} + u \cdot \nabla u \right) = -\nabla p + \mu \nabla^2 u + f$$

```
def factorial(n):  
    if n <= 1:  
        return 1  
    else:  
        return n * factorial(n - 1)
```

$$a^2 + b^2 = c^2$$

$$e^{i\pi} + 1 = 0$$

$$a^2 + b^2 = c^2$$

$$s_{\text{intra}} = \sum_{j=1}^k \sum_{x_i \in C_j} (x_i - \mu_j)(x_i - \mu_j)^T$$

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau) d\tau$$

```
for i in range(len(A)):  
    for j in range(len(B[0])):  
        for k in range(len(B)):  
            C[i][j] += A[i][k] * B[k][j]
```

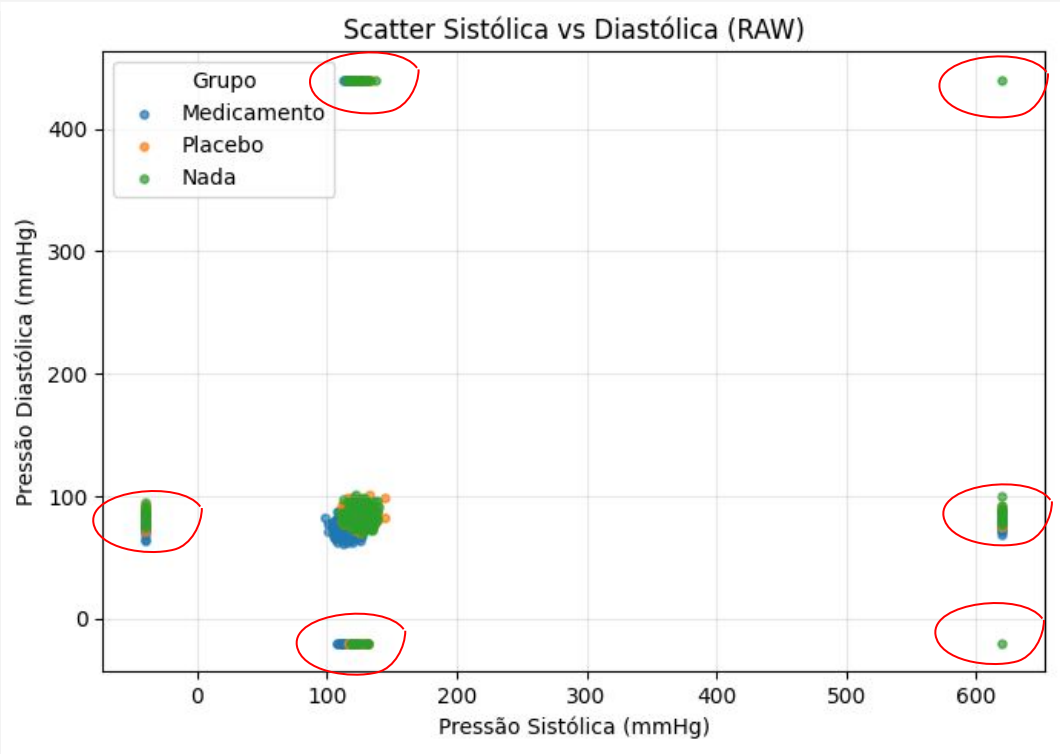
$$i\hbar \frac{\partial}{\partial t} \Psi(x, t) = \left[ -\frac{\hbar^2}{2m} \nabla^2 + V(x, t) \right] \Psi(x, t)$$

$$E = mc^2$$


```
for i in range(n-1):  
    lmin = 1  
    for j in range(i+1, n):  
        if v[i] < v[lmin]:  
            lmin = j  
    noperacoes += 1  
v[i], v[lmin] = v[lmin], v[i]
```

$$E = mc^2$$

# Visualização de dados



Ruídos nos dados


$$f(x) = a_0 + \sum_{n=1}^{\infty} \left[ a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right) \right]$$


$$\rho \left( \frac{\partial u}{\partial t} + u \cdot \nabla u \right) = -\nabla p + \mu \nabla^2 u + f$$

```
int factorial(int n) {
    if (n <= 1)
        return 1;
    else
        return n * factorial(n - 1);
}
```

$$e^{i\pi} + 1 = 0$$
$$a^2 + b^2 = c^2$$
$$S_{\text{intra}} = \sum_{j=1}^k \sum_{x_i \in C_j} (x_i - \mu_j)(x_i - \mu_j)^T$$

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau) d\tau$$

```
for i in range(len(A)):
    for j in range(len(B[0])):
        for k in range(len(B)):
            C[i][j] += A[i][k] * B[k][j]
```

$$i\hbar \frac{\partial}{\partial t} \Psi(x, t) = \left[ -\frac{\hbar^2}{2m} \nabla^2 + V(x, t) \right] \Psi(x, t)$$


$$E = mc^2$$

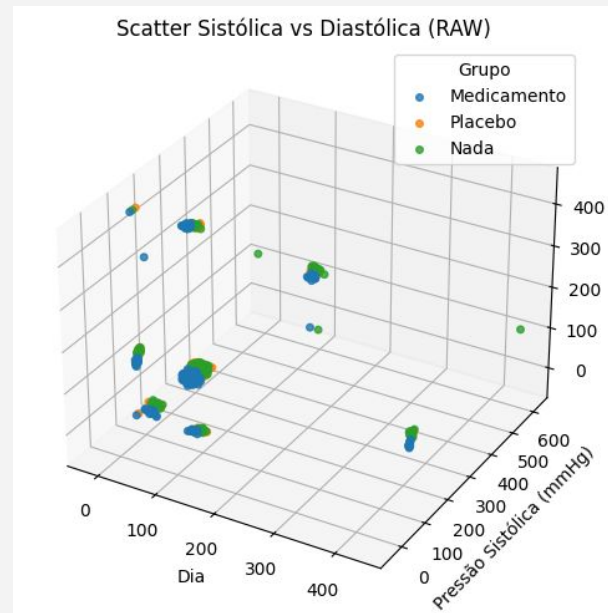
```
for i in range(n-1):
    minn = 1
    for j in range(i+1, n):
        if v[i] < v[minn]:
            minn = j
    swap(v[i], v[minn])
```

# Visualização de dados: 3D

```
fig = plt.figure(figsize=figure_size())
ax = fig.add_subplot(111, projection="3d")

for grupo in ["Medicamento", "Placebo", "Nada"]:
    sub = df[df["Grupo"] == grupo]
    ax.scatter(sub["Diastolica"], sub["Sistolica"], sub["Glicemia"], s=16, alpha=0.8, label=grupo)

ax.set_xlabel("Dia")
ax.set_ylabel("Pressão Sistólica (mmHg)")
ax.set_zlabel("Glicemia")
ax.set_title(titulo)
ax.legend(title="Grupo")
plt.tight_layout()
plt.show()
```



$$f(x) = a_0 + \sum_{n=1}^{\infty} \left[ a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right) \right]$$

$$\rho \left( \frac{\partial u}{\partial t} + u \cdot \nabla u \right) = -\nabla p + \mu \nabla^2 u + f$$

$$e^{i\pi} + 1 = 0$$

$$a^2 + b^2 = c^2$$

$$S_{\text{intra}} = \sum_{j=1}^k \sum_{x_i \in C_j} (x_i - \mu_j)(x_i - \mu_j)^T$$

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau) d\tau$$

$$i\hbar \frac{\partial}{\partial t} \Psi(x,t) = \left[ -\frac{\hbar^2}{2m} \nabla^2 + V(x,t) \right] \Psi(x,t)$$

$$E = mc^2$$

# Visualização de dados: 3D

```
def main():
    #data_loader
    df = carrega_dados()

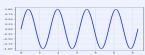
    # Manipulacao de dados: Controle de Qualidade
    df_ok = controle_qualidade(df)

    # analise inicial: tabela sumaria de dados
    tabela_sumaria(df)
    tabela_sumaria(df_ok)

    # Visualização e histogramas RAW
    visualiza_dados(df, titulo="Scatter Sistólica, Diastólica, Glicemia (RAW)")
    histogramas(df, sufixo_titulo="RAW")

    # Visualização e histogramas pós-filtragem
    visualiza_dados(df_ok, titulo="Scatter Sistólica, Diastólica, Glicemia (FILTRADO)")
    histogramas(df_ok, sufixo_titulo="FILTRADO")

    barras_agrupadas(df_ok, sufixo_titulo="FILTRADO")
```


$$f(x) = a_0 + \sum_{n=1}^{\infty} \left[ a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right) \right]$$

$$\rho \left( \frac{\partial u}{\partial t} + u \cdot \nabla u \right) = -\nabla p + \mu \nabla^2 u + f$$

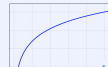
```
int factorial(int n) {
    if (n <= 1)
        return 1;
    else
        return n * factorial(n - 1);
}
```



$$e^{i\pi} + 1 = 0$$

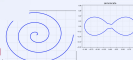
$$a^2 + b^2 = c^2$$

$$S_{\text{intra}} = \sum_{j=1}^k \sum_{x_i \in C_j} (x_i - \mu_j)(x_i - \mu_j)^T$$



$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau) d\tau$$

```
for i in range(len(A)):
    for j in range(len(B[0])):
        for k in range(len(B)):
            C[i][j] += A[i][k] * B[k][j]
```



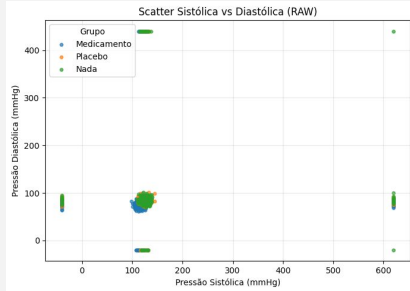
$$i\hbar \frac{\partial}{\partial t} \Psi(x, t) = \left[ -\frac{\hbar^2}{2m} \nabla^2 + V(x, t) \right] \Psi(x, t)$$

$$E = mc^2$$

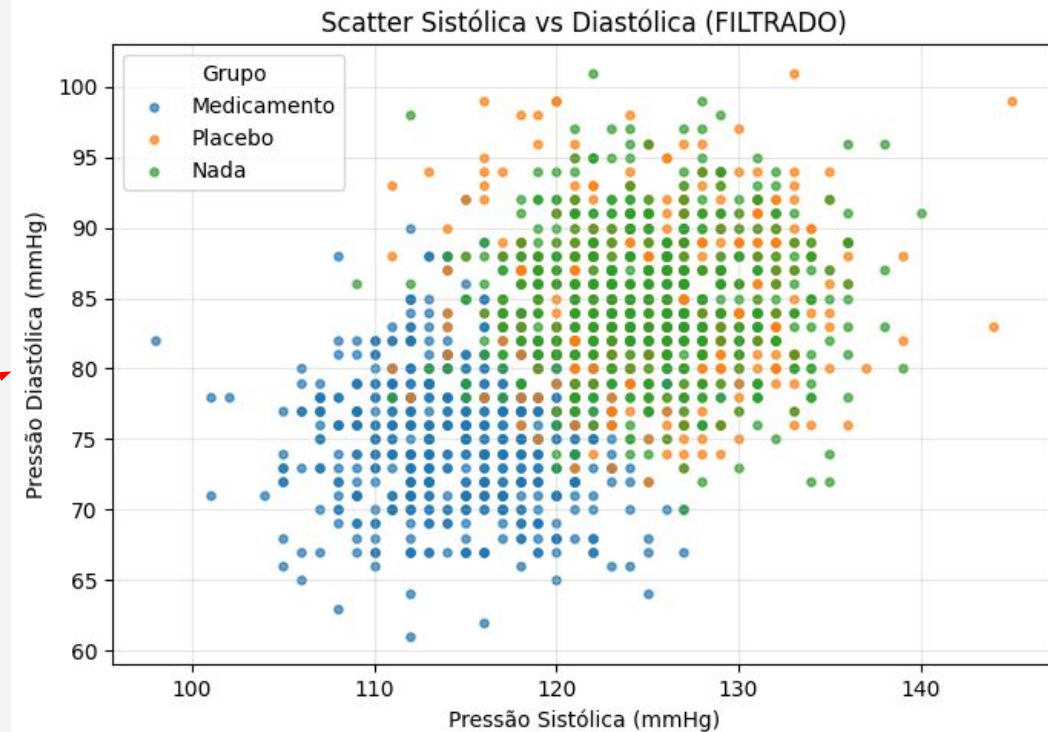
```
for i in range(n-1):
    lmin = 1
    for j in range(i+1, n):
        if v[i] < v[lmin]:
            lmin = j
    noperacoes += 1
    v[i], v[lmin] = v[lmin], v[i]
```



# Visualização de dados: dados filtrados pelo controle de qualidade



```
def main():  
    #data_loader  
    df = carrega_dados()  
  
    # Manipulação de dados: Controle de Qualidade  
    df_ok = controle_qualidade(df)  
  
    # análise inicial: tabela sumária de dados  
    tabela_sumaria(df)  
    tabela_sumaria(df_ok)  
  
    # Visualização e histogramas RAW  
    visualiza_dados(df, titulo="Scatter Sistólica, Diastólica, Glicemia (RAW)")  
    histogramas(df, sufixo_titulo="RAW")  
  
    # Visualização e histogramas pós-filtragem  
    visualiza_dados(df_ok, titulo="Scatter Sistólica, Diastólica, Glicemia (FILTRADO)")  
    histogramas(df_ok, sufixo_titulo="FILTRADO")  
  
    barras_agrupadas(df_ok, sufixo_titulo="FILTRADO")
```



$$f(x) = a_0 + \sum_{n=1}^{\infty} \left[ a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right) \right]$$

$$\rho \left( \frac{\partial u}{\partial t} + u \cdot \nabla u \right) = -\nabla p + \mu \nabla^2 u + f$$

$$e^{i\pi} + 1 = 0$$

$$a^2 + b^2 = c^2$$

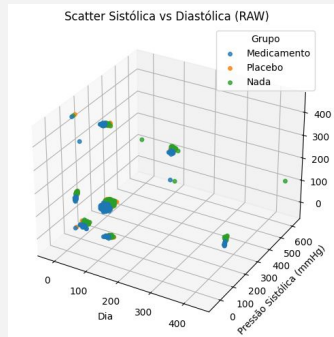
$$S_{\text{intra}} = \sum_{j=1}^k \sum_{x_i \in C_j} (x_i - \mu_j)(x_i - \mu_j)^T$$

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau) d\tau$$

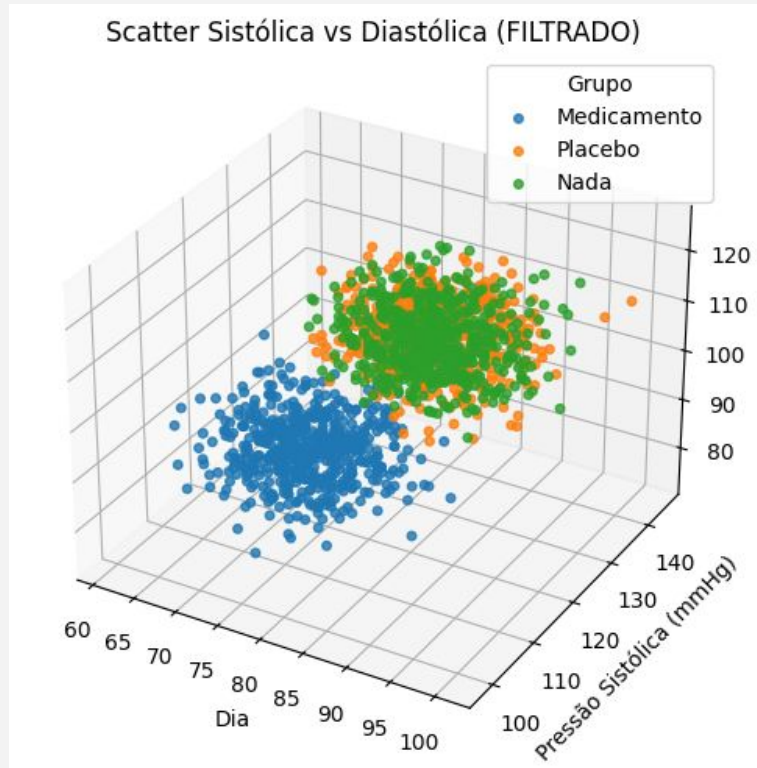
$$i\hbar \frac{\partial}{\partial t} \Psi(x,t) = \left[ -\frac{\hbar^2}{2m} \nabla^2 + V(x,t) \right] \Psi(x,t)$$

$$E = mc^2$$

# Visualização de dados: dados filtrados pelo controle de qualidade



```
def main():  
    #data_loader  
    df = carrega_dados()  
  
    # Manipulação de dados: Controle de Qualidade  
    df_ok = controle_qualidade(df)  
  
    # análise inicial: tabela sumaria de dados  
    tabela_sumaria(df)  
    tabela_sumaria(df_ok)  
  
    # Visualização e histogramas RAW  
    visualiza_dados(df, titulo="Scatter Sistólica, Diastólica, Glicemia (RAW)")  
    histogramas(df, sufixo_titulo="RAW")  
  
    # Visualização e histogramas pós-filtragem  
    visualiza_dados(df_ok, titulo="Scatter Sistólica, Diastólica, Glicemia (FILTRADO)")  
    histogramas(df_ok, sufixo_titulo="FILTRADO")  
  
    barras_agrupadas(df_ok, sufixo_titulo="FILTRADO")
```



$$f(x) = a_0 + \sum_{n=1}^{\infty} \left[ a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right) \right]$$

```
def factorial(n):  
    if n <= 1:  
        return 1  
    else:  
        return n * factorial(n - 1)
```



$$e^{i\pi} + 1 = 0$$

$$a^2 + b^2 = c^2$$

$$S_{\text{intra}} = \sum_{j=1}^k \sum_{x_i \in C_j} (x_i - \mu_j)(x_i - \mu_j)^T$$



$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau) d\tau$$

```
for i in range(len(A)):  
    for j in range(len(B[0])):  
        for k in range(len(B)):  
            C[i][j] += A[i][k] * B[k][j]
```



$$i\hbar \frac{\partial}{\partial t} \Psi(x,t) = \left[ -\frac{\hbar^2}{2m} \nabla^2 + V(x,t) \right] \Psi(x,t)$$



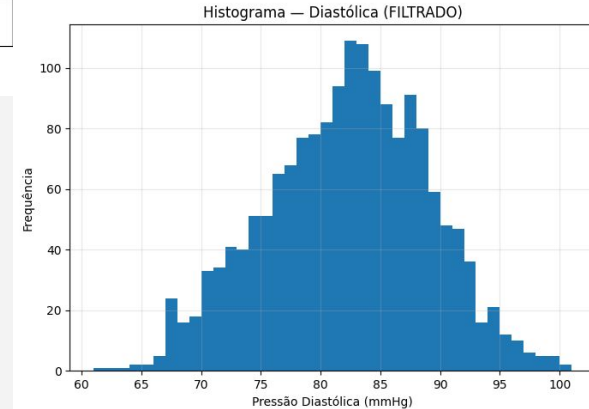
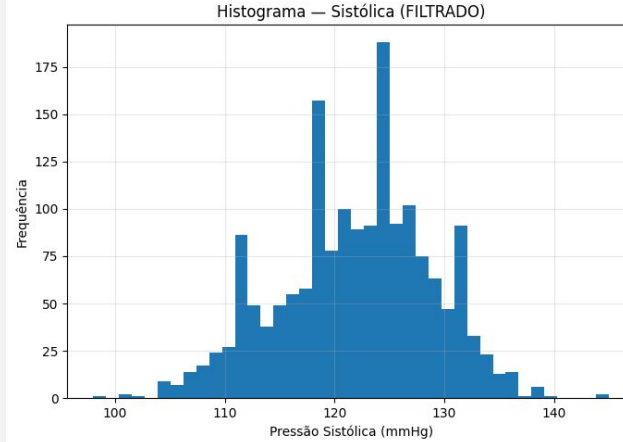
$$E = mc^2$$

```
for i in range(n-1):  
    L[i] = 1  
    for j in range(i+1, n):  
        if v[i][j] < v[L[i]]:  
            L[i] = j  
            noperacao += 1  
v[i], v[L[i]] = v[L[i]], v[i]
```

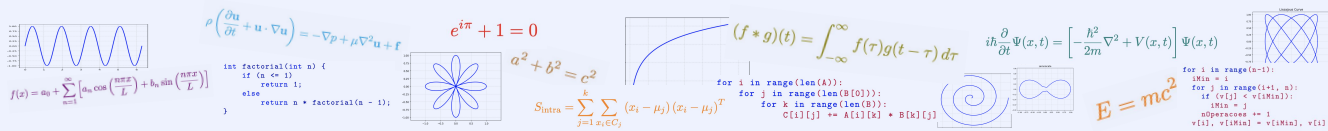


# Visualização de dados: dados filtrados pelo controle de qualidade

```
def histogramas(df, sufixo_titulo="RAW"):  
    # Histograma Sistólica  
    plt.figure(figsize=figure_size())  
    plt.hist(df["Sistolica"].dropna(), bins=40)  
    plt.xlabel("Pressão Sistólica (mmHg)")  
    plt.ylabel("Frequência")  
    plt.title(f"Histograma - Sistólica ({sufixo_titulo})")  
    plt.grid(True, alpha=0.3)  
    plt.tight_layout()  
    plt.show()  
    # Histograma Diastólica  
    plt.figure(figsize=figure_size())  
    plt.hist(df["Diastolica"].dropna(), bins=40)  
    plt.xlabel("Pressão Diastólica (mmHg)")  
    plt.ylabel("Frequência")  
    plt.title(f"Histograma - Diastólica ({sufixo_titulo})")  
    plt.grid(True, alpha=0.3)  
    plt.tight_layout()  
    plt.show()  
    # Histograma Glicemia  
    plt.figure(figsize=figure_size())  
    plt.hist(df["Glicemia"].dropna(), bins=40)  
    plt.xlabel("Glicemia")  
    plt.ylabel("Frequência")  
    plt.title(f"Histograma - Glicemia ({sufixo_titulo})")  
    plt.grid(True, alpha=0.3)  
    plt.tight_layout()  
    plt.show()
```

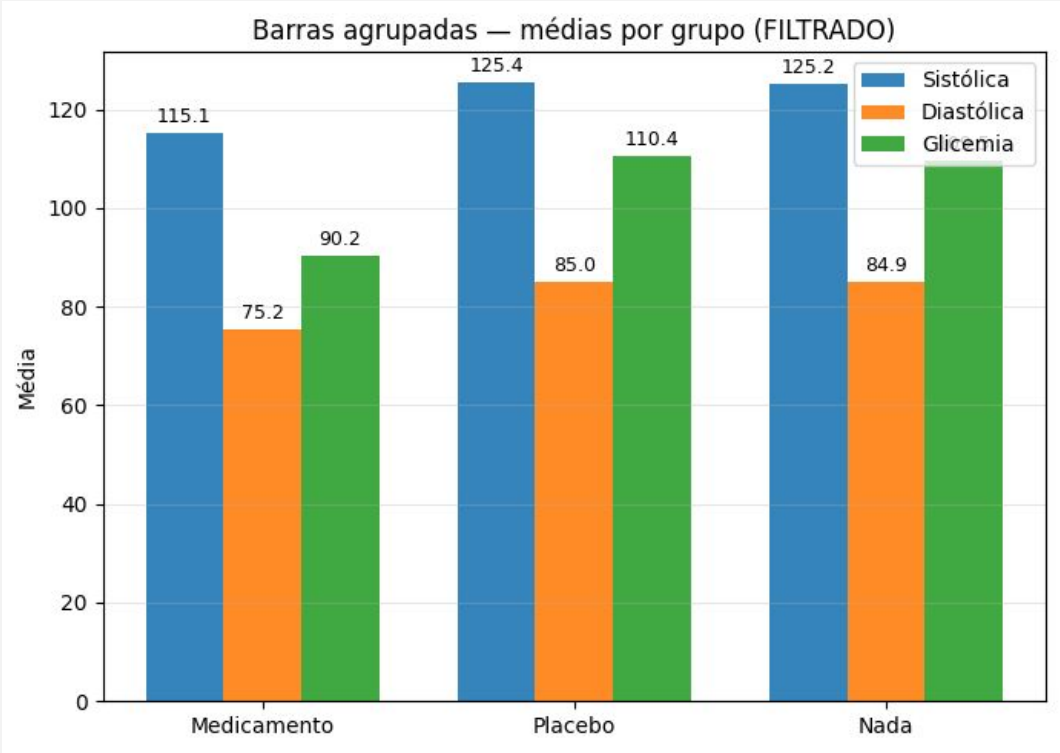



Potencial e facilidade de uso






# Visualização de dados: dados filtrados pelo controle de qualidade




$$f(x) = a_0 + \sum_{n=1}^{\infty} \left[ a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right) \right]$$


$$\rho \left( \frac{\partial u}{\partial t} + u \cdot \nabla u \right) = -\nabla p + \mu \nabla^2 u + f$$

```
int factorial(int n) {  
  if (n <= 1)  
    return 1;  
  else  
    return n * factorial(n - 1);  
}
```

$$e^{i\pi} + 1 = 0$$
$$a^2 + b^2 = c^2$$
$$S_{\text{intra}} = \sum_{j=1}^k \sum_{x_i \in C_j} (x_i - \mu_j)(x_i - \mu_j)^T$$


$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau) d\tau$$

```
for i in range(len(A)):  
  for j in range(len(B[0])):  
    for k in range(len(B)):  
      C[i][j] += A[i][k] * B[k][j]
```



$$i\hbar \frac{\partial}{\partial t} \Psi(x, t) = \left[ -\frac{\hbar^2}{2m} \nabla^2 + V(x, t) \right] \Psi(x, t)$$
$$E = mc^2$$

```
for i in range(n-1):  
  min = 1  
  for j in range(i+1, n):  
    if v[i] < v[min]:  
      min = j  
  swap(v[i], v[min])
```

