

```

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import matplotlib.cm as cm # Importa o módulo matplotlib.cm
5 import matplotlib.pyplot as plt
6 import seaborn as sns
7 import plotly.express as px
8

```

```

1 # ... (código para ler o dataset - substitua pelo path de seu arquivo viia upload) ...
2 # A linha countryData.drop("Unnamed: 0", axis = 1) remove uma coluna desnecessária chamada "Unnamed: 0", que provavelmente foi criada durante a exportação do CSV.
3
4 countryData = pd.read_csv("/content/countryData.csv").drop("Unnamed: 0", axis = 1)
5 countryData

```

	country	year	Access to electricity	Electricity production(coal)	Energy use per capita	co2 emissions	Inflation	Exports	GDP Growth(Annual)	GDP Cap (l
0	BGD	2011	59.599998	1.873032	212.058286	72.875055	NaN	NaN	NaN	
1	BGD	2012	66.155571	1.927433	219.972662	71.993094	6.217504	12.532259	6.521459	883.11
2	BGD	2013	61.500000	2.306099	222.061405	71.590639	7.530406	2.451884	6.013606	981.86
3	BGD	2014	62.400002	1.969738	229.250540	72.052527	6.991639	3.201149	6.061059	1118.87
4	BGD	2015	74.903740	1.689516	NaN	68.045893	6.194280	-2.829990	6.552640	1248.45
...
103	USA	2017	100.000000	NaN	NaN	NaN	2.130110	4.077911	2.332679	60109.65
104	USA	2018	100.000000	NaN	NaN	NaN	2.442583	2.811103	2.996464	63064.41
105	USA	2019	100.000000	NaN	NaN	NaN	1.812210	-0.065189	2.161177	65279.52
106	USA	2020	100.000000	NaN	NaN	NaN	1.233584	-13.562815	-3.404592	63206.52
107	USA	2021	NaN	NaN	NaN	NaN	4.697859	NaN	NaN	

```

1 # Informações sobre os dados, como o número de linhas e colunas, os tipos de dados de cada coluna e algumas estatísticas descritivas.
2
3 countryData.shape # Mostra o número de linhas e colunas
4 countryData.info() # Mostra os tipos de dados de cada coluna
5 countryData.describe() # Mostra estatísticas descritivas (média, desvio padrão, etc.)

```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 108 entries, 0 to 107
Data columns (total 10 columns):
#   Column                                Non-Null Count  Dtype
---  ---                                -
0   country                              108 non-null    object
1   year                                108 non-null    int64
2   Access to electricity                100 non-null    float64
3   Electricity production(coal)         50 non-null     float64
4   Energy use per capita                43 non-null     float64
5   co2 emissions                       60 non-null     float64
6   Inflation                           97 non-null     float64
7   Exports                             81 non-null     float64
8   GDP Growth(Annual)                  90 non-null     float64
9   GDP Per Capita (USD)                 90 non-null     float64
dtypes: float64(8), int64(1), object(1)
memory usage: 8.6+ KB
```

	year	Access to electricity	Electricity production(coal)	Energy use per capita	co2 emissions	Inflation	Exports	GDP Growth(Annual)	GDP Cap (L
count	108.000000	100.000000	50.000000	43.000000	60.000000	97.000000	81.000000	90.000000	90.000000
mean	2015.907407	81.704380	22.932185	2587.131698	25.074314	5.406258	1.809701	3.698121	16486.18
std	3.131203	19.421887	28.831284	2850.983024	21.965650	4.304760	8.849389	3.242400	21753.25
min	2011.000000	36.000000	0.000000	212.058286	0.000000	-0.233353	-26.963364	-7.251755	883.11
25%	2013.000000	69.174997	0.000000	462.073016	4.376406	1.906636	-1.603304	1.851459	1577.30
50%	2016.000000	86.936775	6.052748	778.843658	22.616266	4.948216	2.196750	3.806677	2190.76
75%	2019.000000	100.000000	39.358013	3589.220761	35.714269	7.808765	6.616385	6.477358	39706.31

```
1 # 1-Definição de regiões: Uma lista chamada regions é criada com os nomes das regiões.
2 # 2-Seleção de dados: O código seleciona as colunas "country" e a coluna especificada pelo argumento coluna do DataFrame countryData, removendo linhas com valores ausentes (dropna())
3 # 3-Agrupamento por país: Os dados são agrupados por país usando o método groupby("country"), e a soma dos valores da coluna especificada é calculada para cada país
4 # 4-Cálculo por região
5 # 5-Retorno de dados: A função retorna duas listas: regions com os nomes das regiões e countryPerData com os valores calculados para cada região.
6
7 def separacaoRegiao(countryData, coluna):
8     regions = ["Norte America", "Asia", "Africa", "Sul da Asia"]
9     countryPerData = countryData[["country", coluna]].dropna()
10    countryPerData = countryPerData.groupby("country").sum()
11    countryPerData = [countryPerData[coluna].iloc[1] + countryPerData[coluna].iloc[9],
12                      countryPerData[coluna].iloc[2] + countryPerData[coluna].iloc[5],
13                      countryPerData[coluna].iloc[3] + countryPerData[coluna].iloc[6] + countryPerData[coluna].iloc[7],
14                      countryPerData[coluna].iloc[0] + countryPerData[coluna].iloc[4] + countryPerData[coluna].iloc[8]]
15    return regions, countryPerData
```

```
1 # cria uma lista chamada country que contém todos os países únicos presentes na coluna "country" do DataFrame countryData.
2 # O código itera sobre cada valor da coluna "country" e verifica se o país já está presente na lista country. Se não estiver, o país é adicionado à lista.
3 # método unique() do pandas para obter uma lista de valores únicos diretamente
4
5 country = countryData["country"].unique()
```

```

1 import matplotlib.font_manager as fm
2
3 # Listar fontes disponíveis
4 font_list = [f.name for f in fm.fontManager.ttflist]
5 print(font_list)
6
7 # Escolha uma fonte da lista e use-a no código
8 plt.rcParams['font.family'] = 'STIXGeneral' # Substitua pelo nome da fonte escolhida
9
10 ['STIXSizeThreeSym', 'STIXSizeTwoSym', 'STIXGeneral', 'STIXGeneral', 'cmsy10', 'DejaVu Serif', 'STIXNonUnicode', 'DejaVu Sans', 'STIXNonUnicode', 'DejaVu Sans Mono', 'STIXNonUnicode']
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

```

46     dash1[1].plot(countryData.loc[countryData["country"] == paises, "year"],
47                   countryData.loc[countryData["country"] == paises, "Access to electricity"],
48                   label=paises, color=cmap(pos / len(country)))
49
50 dash1[1].legend(loc=(0.97, 0), prop={"size": 15})
51 dash1[1].spines["top"].set_visible(False)
52 dash1[1].spines["bottom"].set_visible(False)
53 dash1[1].spines["right"].set_visible(False)
54 dash1[1].spines["left"].set_visible(False)
55 dash1[1].grid(axis="y", alpha=0.5)
56 dash1[1].set_ylim(32, 101)
57 dash1[1].set_xticks(range(2011, 2022, 1)) # Ajuste o intervalo de anos se necessário
58 dash1[1].set_title("Acesso à energia per capita desde 2012", fontsize=18)
59 dash1[1].set_xlabel("Year", fontsize=18)
60 dash1[1].set_ylabel("Acesso à energia (%)", fontsize=18) # Adicione a unidade de medida
61
62
63
64 ### 3-#GRÁFICOS DE PIZZA
65
66 ## Gráfico de pizza 1
67 regions, countryPerData = separacaoRegiao(countryData, "Energy use per capita")
68
69 # Paleta de cores vibrantes adaptada ao tema escuro
70 colors = ['#F92672', '#66D9EF', '#A6E22E', '#FD971F'] # Exemplos de cores vibrantes
71
72 dimensao, dash = plt.subplots(1, 4, figsize=(25, 2))
73 plt.subplots_adjust(hspace=2)
74
75 # Ajustes para o tema escuro
76 plt.style.use('dark_background') # Fundo escuro
77 plt.rcParams['text.color'] = 'white' # Texto em branco
78
79 dash[0].pie(countryPerData, explode=(0.1, 0.1, 0, 0), labels=regions, colors=colors,
80             autopct='%1.1f%%', shadow=True, startangle=140, textprops={'fontsize': 10})
81 dash[0].set_title("Energia per capita por Continente", fontsize=12) # Ajuste o tamanho da fonte
82
83
84 ## Gráfico de pizza 2
85 regions, countryPerData = separacaoRegiao(countryData, "co2 emissions")
86
87 # Reutilizando a paleta de cores vibrantes
88 colors = ['#F92672', '#66D9EF', '#A6E22E', '#FD971F']
89
90 # Ajustes para o tema escuro já foram aplicados no código anterior
91
92 dash[1].pie(countryPerData, explode=(0.1, 0, 0, 0), labels=regions, colors=colors,
93             autopct='%1.1f%%', shadow=True, startangle=140, textprops={'fontsize': 10})
94 dash[1].set_title("Emissão de CO2 por Continente", fontsize=12) # Ajuste o tamanho da fonte
95
96
97 ## Gráfico de pizza 3
98 regions, countryPerData = separacaoRegiao(countryData, "Inflation")
99
100 # Reutilizando a paleta de cores vibrantes
101 colors = ['#F92672', '#66D9EF', '#A6E22E', '#FD971F']
102

```

```

103 # Ajustes para o tema escuro já foram aplicados no código anterior
104
105 dash[2].pie(countryPerData, explode=(0, 0, 0, 0.1), labels=regions, colors=colors,
106             autopct='%1.1f%%', shadow=True, startangle=140, textprops={'fontsize': 10})
107 dash[2].set_title("Inflação por Continente", fontsize=12) # Ajuste o tamanho da fonte
108
109
110 ## Gráfico de pizza 4
111 regions, countryPerData = separacaoRegiao(countryData, "Access to electricity")
112
113 # Reutilizando a paleta de cores vibrantes
114 colors = ['#F92672', '#66D9EF', '#A6E22E', '#FD971F']
115
116 # Ajustes para o tema escuro já foram aplicados no código anterior
117
118 dash[3].pie(countryPerData, explode=(0.1, 0, 0, 0), labels=regions, colors=colors,
119             autopct='%1.1f%%', shadow=True, startangle=140, textprops={'fontsize': 10})
120 dash[3].set_title("Acesso à eletricidade por Continente", fontsize=12) # Ajuste o tamanho da fonte
121
122
123 # Verificar valores ausentes em 'Inflation'
124 print(countryData['Inflation'].isnull().sum())
125
126 # Remover linhas com valores ausentes (opcional)
127 countryData.dropna(subset=['Inflation'], inplace=True)
128
129
130
131 ### 4-Gráfico de inflação por Continente
132 plt.figure(figsize=(25, 2.5))
133
134 # Cores vibrantes
135 cmap = plt.cm.get_cmap('plasma') # Escolhe o mapa de cores 'plasma' para cores vibrantes
136
137 for i, pais in enumerate(country): # Itera sobre cada país e seu índice na lista 'country'
138     plt.scatter(countryData.loc[countryData['country'] == pais, "year"], # Define os valores do eixo x (ano) para cada país
139               countryData.loc[countryData['country'] == pais, 'Inflation'], # Define os valores do eixo y (inflação) para cada país
140               alpha=0.7, # Define a opacidade das bolhas (0 = transparente, 1 = opaco)
141               color=cmap(i / len(country)), # Define a cor de cada bolha com base no índice e no número de países
142               label=pais, # Define o rótulo de cada bolha com o nome do país
143               s=countryData.loc[countryData['country'] == pais, 'Inflation'] * 10) # Define o tamanho da bolha proporcional ao valor da inflação
144
145
146 # Ajustes para o tema escuro
147 plt.style.use('dark_background') # Define o estilo do gráfico como 'dark_background' para fundo escuro
148 plt.rcParams['text.color'] = 'white' # Define a cor do texto como branco para melhor visibilidade
149
150 # Remove as bordas do gráfico
151 plt.gca().spines["right"].set_visible(False)
152 plt.gca().spines["bottom"].set_visible(False)
153 plt.gca().spines["top"].set_visible(False)
154 plt.gca().spines["left"].set_visible(False)
155
156 # Define os rótulos do eixo x (anos) e sua cor
157 plt.xticks(range(2012, 2022, 1), color='white')
158
159 # Define a cor dos rótulos do eixo y

```

```
159 # Define a cor dos rótulos do eixo y,  
160 plt.yticks(color='white')  
161  
162 # Define o rótulo do eixo x, seu tamanho e cor  
163 plt.xlabel("Year", fontsize=18, color='white')  
164  
165 # Define o rótulo do eixo y, seu tamanho e cor  
166 plt.ylabel("Inflação", fontsize=18, color='white')  
167  
168 # Define o título do gráfico, seu tamanho e cor  
169 plt.title("Inflação por País", fontsize=22, color='white')  
170  
171 # Define a posição da legenda e o tamanho da fonte  
172 plt.legend(loc=(1, 0), prop={'size': 10})  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192
```

```

↳ <ipython-input-137-4d966bc4d6e8>:20: MatplotlibDeprecationWarning: The get_cmap function was deprecated in Matplotlib 3.5; use plt.get_cmap instead.  

   cmap = plt.cm.get_cmap('plasma') # ou 'viridis', 'inferno' etc.  

  

<ipython-input-137-4d966bc4d6e8>:44: MatplotlibDeprecationWarning: The get_cmap function was deprecated in Matplotlib 3.5; use plt.get_cmap instead.  

   cmap = plt.cm.get_cmap('plasma') # ou 'viridis', 'inferno' etc.

```

0

```
<ipython-input-137-4d966b6c4d6e8>:135: MatplotlibDeprecationWarning: The get_cmap function was deprecated in Matplo
cmap = plt.cm.get_cmap('plasma') # Escolhe o mapa de cores 'plasma' para cores vibrantes
/usr/local/lib/python3.10/dist-packages/matplotlib/collections.py:963: RuntimeWarning: invalid value encountered in
scale = np.sqrt(self._sizes) * dpi / 72.0 * self._factor
<matplotlib.legend.Legend at 0x7fc29abc4d00>
/usr/local/lib/python3.10/dist-packages/matplotlib/collections.py:963: RuntimeWarning: invalid value encountered in
scale = np.sqrt(self._sizes) * dpi / 72.0 * self._factor
```



