

Linguagem de Programação

Linguagem de programação: Introdução a Biblioteca Pandas

Prof.ª Elisa Antolli

- Unidade de Ensino: 04
- Competência da Unidade: Linguagem de programação:
- Introdução a Biblioteca Pandas
- Resumo: Saber utilizar manipulação de dados panda na
- linguagem Python.
- Palavras-chave: Algoritmos; Python; Panda; Manipulação de
- Dados; Orientação a Objeto.
- Título da Teleaula: Linguagem de programação: Introdução a
- Biblioteca Pandas
- Teleaula nº: 04

Contextualização

- Linguagem de programação: Introdução a Biblioteca Pandas
- Introdução a Manipulação de Dados em Pandas
- Visualização de Dados em Python

Linguagem de programação: Introdução a Biblioteca Pandas

Introdução a Biblioteca Pandas

Pandas é um pacote Python que fornece estruturas de dados projetadas para facilitar o trabalho com dados estruturados (tabelas) e de séries temporais.

Para utilizar a biblioteca pandas é preciso fazer a instalação.

Pandas possui duas estruturas de dados que são as principais para a análise/manipulação de dados: a **Series** e o **DataFrame**.

Introdução a Biblioteca Pandas

Uma **Series** é um como um vetor de dados unidimensional), capaz de armazenar diferentes tipos de dados.

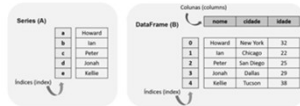
Um **DataFrame** é conjunto de Series, ou como a documentação apresenta, um contêiner para Series.

Ambas estruturas, possuem como grande característica, a indexação das linhas, ou seja, cada linha possui um rótulo (nome) que o identifica, o qual pode ser uma string, um inteiro, um decimal ou uma data.

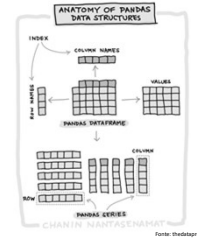
Introdução a Biblioteca Pandas

A Figura abaixo ilustra uma Series (A) e um DataFrame (B).

Veja que uma Series possui somente "uma coluna" de informação e seus rótulos (índices). Um DataFrame pode ter uma ou mais colunas e além dos índices, também há um rótulo de identificação com o nome da coluna. Podemos comparar um DataFrame como uma planilha eletrônica, como o Excel (da Microsoft) ou o Calc (do Open Office).



Introdução a Biblioteca Pandas



Introdução a Biblioteca Pandas

Vamos importar a biblioteca antes de começar nossa primeira linha de código. Por convenção, a biblioteca é importada com o apelido (as) pd. Logo, para utilizar as funcionalidades, vamos utilizar a sintaxe pd.funionalidade.

```
import pandas as pd
```

Introdução a Biblioteca Pandas

- Series: Precisamos utilizar o método Series() do pacote pandas. O método possui o seguinte construtor: `pandas.Series(data=None, index=None, dtype=None, name=None, copy=False, fastpath=False)`.
- Dentre todos os parâmetros esperados, somente um é obrigatório para se criar uma Series com dados (se for uma Series sem dados, nenhum parâmetro é obrigatório), o parâmetro `data=XXXX`.

Introdução a Biblioteca Pandas

```
import pandas as pd

# Criando uma Series com um único valor
s1 = pd.Series([10])

# Criando uma Series com uma lista de valores
s2 = pd.Series([10, 2, -1, None, 15, 23.4])

# Criando uma Series com uma lista de nomes
s3 = pd.Series(['Howard', 'Ian', 'Peter', 'Joseph', 'Kellen'])

# Criando uma Series com uma lista de índices
s4 = pd.Series([0, 1, 2, 3, 4], index=['Howard', 'Ian', 'Peter', 'Joseph', 'Kellen'])
```

Criamos uma Series com um único valor, veja que aparece 0 como índice e 5 como valor.
Quando não deixamos explícito os rótulos (índices) que queremos usar é construído um range de 0 até N-1, onde N é a quantidade de valores.
Criamos uma Series a partir de uma lista de nomes, veja que agora os índices variam de 0 até 4 e o dtype é "object".
Criamos uma Series a partir de um dicionário, a grande diferença desse tipo de dado na construção é que a chave do dicionário é usada como índice.

Introdução a Biblioteca Pandas

```
import pandas as pd

series_dados = pd.Series([10, 2, -1, None, 15, 23.4])

print("Quantidade de linhas = ", series_dados.shape) # Retorna uma tupla com o número de linhas
print("Tipo de dados", series_dados.dtypes) # Retorna o tipo de dados, se for misto será object
print("Os valores são únicos?", series_dados.is_unique) # Verifica se os valores são únicos (sem duplicações)
print("Existem valores nulos?", series_dados.hasna) # Verifica se existem valores nulos
print("Quantos valores existem?", series_dados.count()) # Conta quantos valores existem (exclui os nulos)
```

Criamos uma série contando números e um valor nulo (None). As informações extraídas são mais com relação a "forma" dos dados, portanto poderiam ser usadas independentemente do tipo de dado armazenado na Series, inclusive em um cenário de dados com diferentes tipos.

```
Quantidade de linhas = (6,)
Tipo de dados float64
Os valores são únicos? True
Existem valores nulos? True
Quantos valores existem? 4
```

Introdução a Biblioteca Pandas

```
import pandas as pd

series_dados = pd.Series([10, 2, -1, None, 15, 23, 4])
print('Qual o menor valor?', series_dados.min()) # Extrai o menor valor da Series (nesse caso os dados precisam ser do mesmo tipo)
print('Qual o maior valor?', series_dados.max()) # Extrai o valor máximo, com a mesma condição do mínimo
print('Qual a média aritmética?', series_dados.mean()) # Extrai a média aritmética de uma Series numérica
print('Qual o desvio padrão?', series_dados.std()) # Extrai o desvio padrão de uma Series numérica
print('Qual a mediana?', series_dados.median()) # Extrai a mediana de uma Series numérica
print('Vejamos!', series_dados.describe()) # Exibe um resumo sobre os dados na Series
```

Já as informações neste caso, como se tratam de funções matemáticas e estatísticas, podem ficar muito úteis quando utilizadas para tipos numéricos.

```
Qual o menor valor? -1.0
Qual o maior valor? 23.0
Qual a média aritmética? 10.000000000000001
Qual o desvio padrão? 10.105441871175778
Qual a mediana? 4.0

Series
dtype: float64

count    7.000000
mean    10.000000
std     10.105442
min      2.000000
max     23.000000
0.000000
1.000000
2.000000
3.000000
4.000000
5.000000
6.000000
7.000000
8.000000
9.000000
10.000000
11.000000
12.000000
13.000000
14.000000
15.000000
16.000000
17.000000
18.000000
19.000000
20.000000
21.000000
22.000000
23.000000
24.000000
25.000000
26.000000
27.000000
28.000000
29.000000
30.000000
31.000000
32.000000
33.000000
34.000000
35.000000
36.000000
37.000000
38.000000
39.000000
40.000000
41.000000
42.000000
43.000000
44.000000
45.000000
46.000000
47.000000
48.000000
49.000000
50.000000
51.000000
52.000000
53.000000
54.000000
55.000000
56.000000
57.000000
58.000000
59.000000
60.000000
61.000000
62.000000
63.000000
64.000000
65.000000
66.000000
67.000000
68.000000
69.000000
70.000000
71.000000
72.000000
73.000000
74.000000
75.000000
76.000000
77.000000
78.000000
79.000000
80.000000
81.000000
82.000000
83.000000
84.000000
85.000000
86.000000
87.000000
88.000000
89.000000
90.000000
91.000000
92.000000
93.000000
94.000000
95.000000
96.000000
97.000000
98.000000
99.000000
100.000000
101.000000
102.000000
103.000000
104.000000
105.000000
106.000000
107.000000
108.000000
109.000000
110.000000
111.000000
112.000000
113.000000
114.000000
115.000000
116.000000
117.000000
118.000000
119.000000
120.000000
121.000000
122.000000
123.000000
124.000000
125.000000
126.000000
127.000000
128.000000
129.000000
130.000000
131.000000
132.000000
133.000000
134.000000
135.000000
136.000000
137.000000
138.000000
139.000000
140.000000
141.000000
142.000000
143.000000
144.000000
145.000000
146.000000
147.000000
148.000000
149.000000
150.000000
151.000000
152.000000
153.000000
154.000000
155.000000
156.000000
157.000000
158.000000
159.000000
160.000000
161.000000
162.000000
163.000000
164.000000
165.000000
166.000000
167.000000
168.000000
169.000000
170.000000
171.000000
172.000000
173.000000
174.000000
175.000000
176.000000
177.000000
178.000000
179.000000
180.000000
181.000000
182.000000
183.000000
184.000000
185.000000
186.000000
187.000000
188.000000
189.000000
190.000000
191.000000
192.000000
193.000000
194.000000
195.000000
196.000000
197.000000
198.000000
199.000000
200.000000
201.000000
202.000000
203.000000
204.000000
205.000000
206.000000
207.000000
208.000000
209.000000
210.000000
211.000000
212.000000
213.000000
214.000000
215.000000
216.000000
217.000000
218.000000
219.000000
220.000000
221.000000
222.000000
223.000000
224.000000
225.000000
226.000000
227.000000
228.000000
229.000000
230.000000
231.000000
232.000000
233.000000
234.000000
235.000000
236.000000
237.000000
238.000000
239.000000
240.000000
241.000000
242.000000
243.000000
244.000000
245.000000
246.000000
247.000000
248.000000
249.000000
250.000000
251.000000
252.000000
253.000000
254.000000
255.000000
256.000000
257.000000
258.000000
259.000000
260.000000
261.000000
262.000000
263.000000
264.000000
265.000000
266.000000
267.000000
268.000000
269.000000
270.000000
271.000000
272.000000
273.000000
274.000000
275.000000
276.000000
277.000000
278.000000
279.000000
280.000000
281.000000
282.000000
283.000000
284.000000
285.000000
286.000000
287.000000
288.000000
289.000000
290.000000
291.000000
292.000000
293.000000
294.000000
295.000000
296.000000
297.000000
298.000000
299.000000
300.000000
301.000000
302.000000
303.000000
304.000000
305.000000
306.000000
307.000000
308.000000
309.000000
310.000000
311.000000
312.000000
313.000000
314.000000
315.000000
316.000000
317.000000
318.000000
319.000000
320.000000
321.000000
322.000000
323.000000
324.000000
325.000000
326.000000
327.000000
328.000000
329.000000
330.000000
331.000000
332.000000
333.000000
334.000000
335.000000
336.000000
337.000000
338.000000
339.000000
340.000000
341.000000
342.000000
343.000000
344.000000
345.000000
346.000000
347.000000
348.000000
349.000000
350.000000
351.000000
352.000000
353.000000
354.000000
355.000000
356.000000
357.000000
358.000000
359.000000
360.000000
361.000000
362.000000
363.000000
364.000000
365.000000
366.000000
367.000000
368.000000
369.000000
370.000000
371.000000
372.000000
373.000000
374.000000
375.000000
376.000000
377.000000
378.000000
379.000000
380.000000
381.000000
382.000000
383.000000
384.000000
385.000000
386.000000
387.000000
388.000000
389.000000
390.000000
391.000000
392.000000
393.000000
394.000000
395.000000
396.000000
397.000000
398.000000
399.000000
400.000000
401.000000
402.000000
403.000000
404.000000
405.000000
406.000000
407.000000
408.000000
409.000000
410.000000
411.000000
412.000000
413.000000
414.000000
415.000000
416.000000
417.000000
418.000000
419.000000
420.000000
421.000000
422.000000
423.000000
424.000000
425.000000
426.000000
427.000000
428.000000
429.000000
430.000000
431.000000
432.000000
433.000000
434.000000
435.000000
436.000000
437.000000
438.000000
439.000000
440.000000
441.000000
442.000000
443.000000
444.000000
445.000000
446.000000
447.000000
448.000000
449.000000
450.000000
451.000000
452.000000
453.000000
454.000000
455.000000
456.000000
457.000000
458.000000
459.000000
460.000000
461.000000
462.000000
463.000000
464.000000
465.000000
466.000000
467.000000
468.000000
469.000000
470.000000
471.000000
472.000000
473.000000
474.000000
475.000000
476.000000
477.000000
478.000000
479.000000
480.000000
481.000000
482.000000
483.000000
484.000000
485.000000
486.000000
487.000000
488.000000
489.000000
490.000000
491.000000
492.000000
493.000000
494.000000
495.000000
496.000000
497.000000
498.000000
499.000000
500.000000
501.000000
502.000000
503.000000
504.000000
505.000000
506.000000
507.000000
508.000000
509.000000
510.000000
511.000000
512.000000
513.000000
514.000000
515.000000
516.000000
517.000000
518.000000
519.000000
520.000000
521.000000
522.000000
523.000000
524.000000
525.000000
526.000000
527.000000
528.000000
529.000000
530.000000
531.000000
532.000000
533.000000
534.000000
535.000000
536.000000
537.000000
538.000000
539.000000
540.000000
541.000000
542.000000
543.000000
544.000000
545.000000
546.000000
547.000000
548.000000
549.000000
550.000000
551.000000
552.000000
553.000000
554.000000
555.000000
556.000000
557.000000
558.000000
559.000000
560.000000
561.000000
562.000000
563.000000
564.000000
565.000000
566.000000
567.000000
568.000000
569.000000
570.000000
571.000000
572.000000
573.000000
574.000000
575.000000
576.000000
577.000000
578.000000
579.000000
580.000000
581.000000
582.000000
583.000000
584.000000
585.000000
586.000000
587.000000
588.000000
589.000000
590.000000
591.000000
592.000000
593.000000
594.000000
595.000000
596.000000
597.000000
598.000000
599.000000
600.000000
601.000000
602.000000
603.000000
604.000000
605.000000
606.000000
607.000000
608.000000
609.000000
610.000000
611.000000
612.000000
613.000000
614.000000
615.000000
616.000000
617.000000
618.000000
619.000000
620.000000
621.000000
622.000000
623.000000
624.000000
625.000000
626.000000
627.000000
628.000000
629.000000
630.000000
631.000000
632.000000
633.000000
634.000000
635.000000
636.000000
637.000000
638.000000
639.000000
640.000000
641.000000
642.000000
643.000000
644.000000
645.000000
646.000000
647.000000
648.000000
649.000000
650.000000
651.000000
652.000000
653.000000
654.000000
655.000000
656.000000
657.000000
658.000000
659.000000
660.000000
661.000000
662.000000
663.000000
664.000000
665.000000
666.000000
667.000000
668.000000
669.000000
670.000000
671.000000
672.000000
673.000000
674.000000
675.000000
676.000000
677.000000
678.000000
679.000000
680.000000
681.000000
682.000000
683.000000
684.000000
685.000000
686.000000
687.000000
688.000000
689.000000
690.000000
691.000000
692.000000
693.000000
694.000000
695.000000
696.000000
697.000000
698.000000
699.000000
700.000000
701.000000
702.000000
703.000000
704.000000
705.000000
706.000000
707.000000
708.000000
709.000000
710.000000
711.000000
712.000000
713.000000
714.000000
715.000000
716.000000
717.000000
718.000000
719.000000
720.000000
721.000000
722.000000
723.000000
724.000000
725.000000
726.000000
727.000000
728.000000
729.000000
730.000000
731.000000
732.000000
733.000000
734.000000
735.000000
736.000000
737.000000
738.000000
739.000000
740.000000
741.000000
742.000000
743.000000
744.000000
745.000000
746.000000
747.000000
748.000000
749.000000
750.000000
751.000000
752.000000
753.000000
754.000000
755.000000
756.000000
757.000000
758.000000
759.000000
760.000000
761.000000
762.000000
763.000000
764.000000
765.000000
766.000000
767.000000
768.000000
769.000000
770.000000
771.000000
772.000000
773.000000
774.000000
775.000000
776.000000
777.000000
778.000000
779.000000
780.000000
781.000000
782.000000
783.000000
784.000000
785.000000
786.000000
787.000000
788.000000
789.000000
790.000000
791.000000
792.000000
793.000000
794.000000
795.000000
796.000000
797.000000
798.000000
799.000000
800.000000
801.000000
802.000000
803.000000
804.000000
805.000000
806.000000
807.000000
808.000000
809.000000
810.000000
811.000000
812.000000
813.000000
814.000000
815.000000
816.000000
817.000000
818.000000
819.000000
820.000000
821.000000
822.000000
823.000000
824.000000
825.000000
826.000000
827.000000
828.000000
829.000000
830.000000
831.000000
832.000000
833.000000
834.000000
835.000000
836.000000
837.000000
838.000000
839.000000
840.000000
841.000000
842.000000
843.000000
844.000000
845.000000
846.000000
847.000000
848.000000
849.000000
850.000000
851.000000
852.000000
853.000000
854.000000
855.000000
856.000000
857.000000
858.000000
859.000000
860.000000
861.000000
862.000000
863.000000
864.000000
865.000000
866.000000
867.000000
868.000000
869.000000
870.000000
871.000000
872.000000
873.000000
874.000000
875.000000
876.000000
877.000000
878.000000
879.000000
880.000000
881.000000
882.000000
883.000000
884.000000
885.000000
886.000000
887.000000
888.000000
889.000000
890.000000
891.000000
892.000000
893.000000
894.000000
895.000000
896.000000
897.000000
898.000000
899.000000
900.000000
901.000000
902.000000
903.000000
904.000000
905.000000
906.000000
907.000000
908.000000
909.000000
910.000000
911.000000
912.000000
913.000000
914.000000
915.000000
916.000000
917.000000
918.000000
919.000000
920.000000
921.000000
922.000000
923.000000
924.000000
925.000000
926.000000
927.000000
928.000000
929.000000
930.000000
931.000000
932.000000
933.000000
934.000000
935.000000
936.000000
937.000000
938.000000
939.000000
940.000000
941.000000
942.000000
943.000000
944.000000
945.000000
946.000000
947.000000
948.000000
949.000000
950.000000
951.000000
952.000000
953.000000
954.000000
955.000000
956.000000
957.000000
958.000000
959.000000
960.000000
961.000000
962.000000
963.000000
964.000000
965.000000
966.000000
967.000000
968.000000
969.000000
970.000000
971.000000
972.000000
973.000000
974.000000
975.000000
976.000000
977.000000
978.000000
979.000000
980.000000
981.000000
982.000000
983.000000
984.000000
985.000000
986.000000
987.000000
988.000000
989.000000
990.000000
991.000000
992.000000
993.000000
994.000000
995.000000
996.000000
997.000000
998.000000
999.000000
1000.000000
1001.000000
1002.000000
1003.000000
1004.000000
1005.000000
1006.000000
1007.000000
1008.000000
1009.000000
1010.000000
1011.000000
1012.000000
1013.000000
1014.000000
1015.000000
1016.000000
1017.000000
1018.000000
1019.000000
1020.000000
1021.000000
1022.000000
1023.000000
1024.000000
1025.000000
1026.000000
1027.000000
1028.000000
1029.000000
1030.000000
1031.000000
1032.000000
1033.000000
1034.000000
1035.000000
1036.000000
1037.000000
1038.000000
1039.000000
1040.000000
1041.000000
1042.000000
1043.000000
1044.000000
1045.000000
1046.000000
1047.000000
1048.000000
1049.000000
1050.000000
1051.000000
1052.000000
1053.000000
1054.000000
1055.000000
1056.000000
1057.000000
1058.000000
1059.000000
1060.000000
1061.000000
1062.000000
1063.000000
1064.000000
1065.000000
1066.000000
1067.000000
1068.000000
1069.000000
1070.000000
1071.000000
1072.000000
1073.000000
1074.000000
1075.000000
1076.000000
1077.
```

Introdução a Biblioteca Pandas

Um dos grandes recursos da biblioteca pandas é sua capacidade de fazer leitura de dados estruturados, através de seus métodos, guardando em um DataFrame.

A biblioteca possui uma série de métodos "read", cuja sintaxe é: `pandas.read_XXXX()` onde a sequência de X representa as diversas opções disponíveis.

Vamos utilizar o método `read_html()` para capturar os dados e carregar em um DataFrame.

```
import pandas as pd
url = 'https://en.wikipedia.org/wiki/Minnesota'
dfs = pd.read_html(url)
print(type(dfs))
print(len(dfs))
```

Sabendo que o tamanho da lista resultado do método é 1, então para obter a tabela que queremos, basta acessar a posição 0 da lista.

Introdução a manipulação de dados em pandas

Manipulação de dados em pandas

Métodos para leitura e escrita da biblioteca pandas

- A biblioteca pandas foi desenvolvida para trabalhar com dados estruturados, ou seja, dados dispostos em linhas e colunas. Os dados podem estar gravados em arquivos, em páginas web, em APIs, em outros softwares, em object stores (sistemas de armazenamento em cloud) ou em bancos de dados.
- Para todas essas origens (e até mais), a biblioteca possui métodos capazes de fazer a leitura dos dados e carregar em um DataFrame.
- Dentre todos os possíveis métodos para leitura, nessa aula vamos estudar o `read_json`, o `read_csv` e a função `read_sql`, que contempla a função `read_sql_query`.

Manipulação de dados em pandas

Métodos para leitura e escrita da biblioteca pandas

- JSON (JavaScript Object Notation - Notação de Objetos JavaScript) é uma formatação leve de troca de dados e independente de linguagem de programação.
- CSV (comma-separated values - valores separados por vírgula) é um formato de arquivo, nos quais os dados são separados por um delimitador.

Manipulação de dados em pandas

Leitura de JSON e CSV com pandas

É necessário fazer a importação da biblioteca, só precisamos importar uma única vez no notebook ou no script .py.

```
import pandas
```

A leitura de um arquivo JSON deve ser feita com o método:

```
pandas.read_json(path_or_buf=None, orient=None, typ='frame', dtype=None,
convert_axes=None, convert_dates=True, keep_default_dates=True, numpy=False,
precise_float=False, date_unit=None, encoding=None, lines=False, chunksize=None,
compression='infer')
```

O único parâmetro que é obrigatório para se carregar os dados é o "path_or_buf", no qual deve ser passado um caminho para o arquivo ou um "arquivo como objeto" que é um arquivo lido com a função `open()`, por exemplo.

Manipulação de dados em pandas

Estamos usando o método `read_json` para carregar dados de uma API. Veja que estamos passando o caminho para o método. Nessa fonte de dados, são encontradas a taxa selic de cada dia.

```
pd.read_json("https://api.bcb.gov.br/dados/serie/bcdata.sgs.11/dados?formato=json").head()
```

Para realizar o carregamento os dados, é necessário incluir o caminho(diretório), portanto o parâmetro "`filepath_or_buffer`" é obrigatório.

Outro parâmetro que é importante para a leitura desse arquivo é o `sep` ou `delimiter` (ambos fazem a mesma coisa), veja que `sep`, por padrão possui o valor `','`, ou seja, caso não seja especificado nenhum valor, então o método fará a leitura dos dados considerando que estão separados por **vírgula**.

O parâmetro `header`, tem como valor padrão 'infer', que significa que o método realiza a inferência para os nomes das colunas a partir da primeira linha de dados do arquivo.

Manipulação de dados em pandas

Além de vários métodos para carregar e salvar os dados, a biblioteca pandas possui uma diversidade de métodos para a transformação dos dados e a extração de informação para áreas de negócio.

O trabalho com dados: capturar os dados em suas origens, fazer transformações nos dados a fim de padronizá-los, aplicar técnicas estatísticas clássicas ou algoritmos de **machine/deep learning** feito por engenheiros e cientistas de dados.

Cada profissional atuando em uma parte específica, dependendo da organização da empresa. Em todo esse trabalho é comum fazer a divisão em duas etapas: **(I) captura e transformação/padronização dos dados, (II) extração de informações.**

Manipulação de dados em pandas

Além de vários métodos para carregar e salvar os dados, a biblioteca pandas possui uma diversidade de métodos para a transformação dos dados e a extração de informação para áreas de negócio.

O trabalho com dados: capturar os dados em suas origens, fazer transformações nos dados a fim de padronizá-los, aplicar técnicas estatísticas clássicas ou algoritmos de **machine/deep learning** feito por engenheiros e cientistas de dados.

Cada profissional atuando em uma parte específica, dependendo da organização da empresa. Em todo esse trabalho é comum fazer a divisão em duas etapas: **(I) captura e transformação/padronização dos dados, (II) extração de informações.**

Trabalhando com dados

Trabalhando com dados

Etapas de captura e transformação/padronização dos dados

A extração dos dados pode ser realizada por meio do método `read_json()` e guardando em um `DataFrame (DF)` pandas. Ao carregar os dados em um DF, podemos visualizar quantas linhas e colunas, bem como, os tipos de dados em cada coluna, com o método `info()`.

Remover linhas duplicadas

Para o carregamento de uma base de dados, um dos primeiros tratamentos que devemos fazer é remover os dados duplicados. Certamente, saber qual registro remover, depende da área de negócio e do problema a ser resolvido.

Um `DataFrame` da biblioteca pandas possui o método `meu_df.drop_duplicates()` que permite fazer essa remoção de dados duplicados.

Trabalhando com dados

Criar novas colunas

A segunda transformação que veremos é como criar uma nova coluna. A sintaxe é similar a criar uma nova chave em um dicionário:

`meu_df['nova_coluna'] = dado.`

Ex.: Uma coluna que adiciona a data de extração das informações:

```
import pandas as pd
from datetime import datetime as dt

data_names = "https://arquivo.cnpj.br/cnpj/cnpj.json"
df = pd.read_json(data_names, orient="rows")
print(pd.DataFrame(data_names, columns=["nome"]))

data_extracao = data_names
df["data_extracao"] = data_extracao

print(df.info())
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 2 columns):
 #   column      Non-Null Count  Dtype
---  ---
 0   nome        5 non-null        object
 1   data_extracao  5 non-null        object
dtypes: object(2)
memory usage: 208.0+ bytes
None
```

Trabalhando com dados

Método `to_datetime()` e `astype()`

Trabalhar com o tipo "data" pode trazer vantagens, como por exemplo, ordenar da data mais recente para mais antiga. Vamos utilizar os métodos **`pandas.to_datetime()`** e **`minha_series.astype()`** para fazer a conversão e transformar as colunas `data` e `data_extracao`.

Trabalhando com dados

O formato resultante ano-mês-dia é um padrão do datetime64[ns], que segue o padrão internacional, no qual o ano vem primeiro, seguido do mês e por último o dia.

```
data_extracao = date.today()
dfs['data_extracao'] = data_extracao
dfs['data_extracao'] = dfs['data_extracao'].astype('datetime64[ns]')

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 2 columns):
# Column            Non-Null Count  Dtype
---  -
0 name              5 non-null      object
1 data_extracao     5 non-null      datetime64[ns]
dtypes: datetime64[ns](1), object(1)
memory usage: 208.0+ bytes
```

Trabalhando com dados

```
data_extracao = date.today()
dfs['data_extracao'] = data_extracao
dfs['data_extracao'] = dfs['data_extracao'].astype('datetime64[ns]')
dfs.sort_values(by='data_extracao', ascending=False, inplace=True)

print(dfs.head())
```

```
name
0 Howard
1 Ian
2 Peter
3 Jonah
4 Kellie
name data_extracao
0 Howard 2022-11-10
1 Ian 2022-11-10
2 Peter 2022-11-10
3 Jonah 2022-11-10
4 Kellie 2022-11-10
```

Trabalhando com dados

Método `reset_index()` e `set_index()`

Nenhuma transformação afeta o índice, lembra-se como não especificamos rótulos ele usa um intervalo numérico, mas esse intervalo é diferente da posições de um vetor, pois é um nome e vai acompanhar a linha independente da transformação.

As únicas formas de alterar o índice são com os métodos **`reset_index()`** e **`set_index()`**. O primeiro redefine o índice usando o padrão e o segundo define novos índices.

Durante a transformação dos dados, pode ser necessário definir novos valores para os índices, ao invés de usar o range numérico. Essa transformação pode ser feita usando o método **`meu_df.set_index()`**. O método permite especificar os novos valores usando uma coluna já existente ou então passando uma lista, de tamanho igual a quantidade de linhas.

Trabalhando com dados

Filtros com loc

Um dos recursos mais utilizados por equipes das áreas de dados é a aplicação de filtros. A distinção por gênero, por exemplo, pode ser um filtro. Um exemplo de filtro poderia possibilitar comparar a idade de todos, com a idade de cada grupo e entender se as mulheres ou homens estão abaixo ou acima da média geral.

DataFrames da biblioteca pandas possuem uma propriedade chamada **loc**, essa propriedade permite acessar um conjunto de linhas (filtrar linhas), por meio do índice ou por um vetor booleano (vetor de **True** ou **False**).

Trabalhando com dados

Filtros com testes booleanos

Podemos usar operadores relacionais e lógicos para fazer testes condicionais com os valores das colunas de um DF.

Ao criarmos um teste condicional, internamente, a biblioteca testa todas as linhas do DF ou da Series, retornando uma Series booleana, ou seja, composta por valores **True** ou **False**.

O teste condicional pode ser construído também utilizando operadores lógicos. Para a operação lógica **AND** (E), em pandas, usa-se o caracter **&**. Para fazer a operação lógica OR (OU), usa-se o caracter **|**.

Cada teste deve estar entre parênteses, senão ocorre um erro.

Ao criar as condições, basta aplicá-las no DataFrame para criar o filtro.

A construção é feita passando a condição para a propriedade **loc**.

Trabalhando com dados

Filtros com testes booleanos

```
dfs = pd.DataFrame(lista_nomes, columns=['nome'])
data_extracao = dt.now()
dfs['idade'] = 30
dfs=dfs.append({'nome': 'TESTE', 'idade': 25}, ignore_index=True)
print(dfs)
print(dfs.loc[(dfs['idade'] < 30)])
```

nome	idade
TESTE	25

Trabalhando com dados

Banco de dados com pandas

Além dos métodos para acessar arquivos, a biblioteca pandas possui dois métodos que permitem executar instruções SQL em banco de dados.

Os métodos são:

1. `pandas.read_sql(sql, con, index_col=None, coerce_float=True, params=None, parse_dates=None, columns=None, chunksize=None)`
2. `pandas.read_sql_query(sql, con, index_col=None, coerce_float=True, params=None, parse_dates=None, chunksize=None)`

O mínimo de parâmetros que ambos métodos exigem é a instrução SQL e uma conexão com um banco de dados (con). A conexão com o banco de dados, deve ser feita usando uma outra biblioteca, por exemplo, sqlalchemy (suporte para diversos bancos), pyodbc para SQL Server, cx_Oracle para Oracle, psycopg2 para PostgreSQL, dentre outras.

Visualização de dados em python

Visualização de dados em python

Bibliotecas e funções para criação de gráficos

Para a criação de gráficos em Python são utilizadas as bibliotecas `matplotlib` e outras baseadas na matplotlib, além de funções que permitem criar e personalizar os gráficos.

Matplotlib

A instalação da biblioteca pode ser feita via pip install: `pip install matplotlib`, lembrando que em ambientes como o projeto Anaconda e o Colab esse recurso já está disponível.

Visualização de dados em python

Existem duas sintaxes que são amplamente adotadas para importar essa biblioteca para o projeto:

```
import matplotlib.pyplot as plt
from matplotlib import pyplot as plt
```

Em ambas formas de importação utilizamos o apelido "plt" que é uma convenção adotada para facilitar o uso das funções.

Visualização de dados em python

Ao trabalharmos no jupyter notebook com o kernel do IPython (o kernel do IPython é o backend de execução do Python para o Jupyter), podemos habilitar uma opção para fazer a impressão do gráfico "inline", ou seja, no próprio notebook. Para habilitar utiliza-se a sintaxe: `%matplotlib inline`.

Portanto, projetos no jupyter notebook, que utilizem o matplotlib sempre terão no começo, os comandos a seguir.

```
from matplotlib import pyplot as plt
%matplotlib inline
```

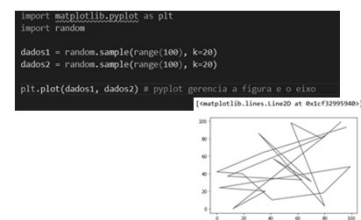
Visualização de dados em python

Vamos criar duas listas aleatórias de valores inteiros com o módulo random e então plotar um gráfico de linhas, com a função **plt.plot()** do módulo pyplot.

Após criar duas listas com valores aleatórios, a função plot() as recebe como parâmetros, utilizando-as para os valores dos eixos **horizontal** (x) e **vertical** (y) e já cria o gráfico.

Parâmetros e muitos outros, podem ser configurados!

Visualização de dados em python



Visualização de dados em python

Existem essencialmente duas maneiras de usar o Matplotlib:

Deixar para o pyplot criar e gerenciar automaticamente figuras e eixos, e usar as funções do pyplot para plotagem.

Criar explicitamente figuras e eixos e chamar métodos sobre eles (o "estilo orientado a objetos (OO)").

No gráfico que criamos, nós utilizamos a opção 1, ou seja, foi o próprio módulo que criou o ambiente da figura e do eixo.

Ao utilizar a segunda opção, podemos criar uma figura com ou sem eixos, com a função **plt.subplots()**, que quando invocada sem parâmetros, cria um layout de figura com 1 linha e 1 coluna.

Visualização de dados em python

Biblioteca pandas

As principais estruturas de dados da biblioteca pandas (Series e DataFrame) possuem o método **plot()**, construído com base no matplotlib e que permite criar gráficos a partir dos dados nas estruturas.

A partir de um DataFrame, podemos invocar o método:

```
df.plot(*args, **kwargs)
```

para criar os gráficos. Os argumentos dessa função, podem variar, mas existem três que são triviais: os **nomes** das colunas com os dados para os eixos **x** e **y**, bem como o tipo de gráfico (kind).

Visualização de dados em python

Biblioteca seaborn

Seaborn é outra biblioteca Python, também baseada na matplotlib, que foi desenvolvida especificamente para criação de gráficos. Seaborn pode ser instalado via pip install: `pip install seaborn`, e para utilizar no projeto existe uma convenção para sintaxe:

```
import seaborn as sns.
```

A biblioteca conta com um repositório de datasets que podem ser usados para explorar as funcionalidades.

O tipo de dados que uma coluna possui é muito importante para a biblioteca seaborn, uma vez que as funções usadas para construir os gráficos são divididas em grupos: relacional, categóricos, distribuição, regressão, matriz e grid.

Visualização de dados em python

Função barplot()

Dentro do grupo de funções para gráficos de variáveis categóricas, temos o **barplot()**, que permite criar gráficos de barras, mas... por que usáramos essa função e não a biblioteca pandas?

A resposta está nas opções de parâmetros que cada biblioteca suporta. Construtor da função barplot possui uma série de parâmetros estatísticos, que dão muita flexibilidade e poder aos cientista de dados, vamos falar sobre o parâmetro **"estimator"**, que por default é a função média. Isso significa que cada barra do gráfico, exibirá a média dos valores de uma determinada coluna, o que pode não fazer sentido, uma vez que queremos exibir a quantidade dos valores (len) ou a soma (sum).

Existem muitos conceitos estatísticos envolvidos e a biblioteca seaborn fornece mecanismos para que essas informações estejam presentes nos resultados visuais.

Visualização de dados em python

Função countplot()

Esse método não aceita que sejam passados valores de x e y ao mesmo tempo, pois a contagem será feita sobre uma variável categórica, portanto devemos especificar **x** ou **y**, a diferença será na orientação do gráfico. Se informamos **x**, teremos um gráfico na vertical, se **y**, na horizontal.

Função scatterplot()

Os gráficos do grupo relacional, permitem avaliar, de forma visual a relação entre duas variáveis: **x**, **y**. O gráfico scatterplot é muito utilizado por cientistas de dados que estão buscando por padrões nos dados.

Visualização de dados em python

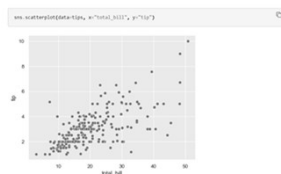
Função countplot()



Fonte: seaborn

Visualização de dados em python

Função scatterplot()



Fonte: seaborn

Recapitulando

Recapitulando

- Linguagem de programação: Introdução a Biblioteca Pandas
- Introdução a Manipulação de Dados em Pandas
- Visualização de Dados em Python

