Missing values

January 25, 2022

[1]:
```
#tem dois tipos de missing data, o primeiro é quando você faz uma pesquisa e a␣
 ↪pessoa não responde uma pergunta
#específica = missing at random
#geralmente o dado que está faltando tem relação com um outro campo do dataset,␣
 ↪caso não possua relacionamento
#com algum outro dado, então é chamado de missing completly at random MCAR
```

[ ]:
```

```

[3]:
```
import pandas as pd
#o pandas consegue facilmente identificar um dado faltando
#dados faltando geralmente vem no dataset como NaN, Null, NONE ou N/A
#a função read_csv tem um parâmetro chamado na_values que nos deixa especificar␣
 ↪o formato dos missing values
#como scalar, strin, lista ou dicionários para serem usados
```

[12]:
```
df = pd.read_csv('resources/week-2/datasets/class_grades.csv')
df.head()
```

[12]:

	Prefix	Assignment	Tutorial	Midterm	TakeHome	Final
0	5	57.14	34.09	64.38	51.48	52.50
1	8	95.05	105.49	67.50	99.07	68.33
2	8	83.70	83.17	NaN	63.15	48.89
3	7	NaN	NaN	49.38	105.93	80.56
4	8	91.32	93.64	95.00	107.41	73.89

[18]:
```
#nós podemos usar a função isnull() para criar uma máscara booleana do DF, isso␣
 ↪transmite o isnull
#para todas células
mask = df.isnull()
mask.head()
```

[18]:

	Prefix	Assignment	Tutorial	Midterm	TakeHome	Final
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	True	False	False
3	False	True	True	False	False	False
4	False	False	False	False	False	False

[19]:
```
#isso pode ser util para processar linhas baseadas em certas colunas de dados.␣
 ↪outra operação bem útil
#é ser capaz de dropar todas as colunas que estão com dados faltando usando o␣
 ↪dropna()
df.dropna().head(10)
```

[19]:

	Prefix	Assignment	Tutorial	Midterm	TakeHome	Final
0	5	57.14	34.09	64.38	51.48	52.50
1	8	95.05	105.49	67.50	99.07	68.33
4	8	91.32	93.64	95.00	107.41	73.89
5	7	95.00	92.58	93.12	97.78	68.06
6	8	95.05	102.99	56.25	99.07	50.00
8	8	84.26	93.10	47.50	18.52	50.83
9	7	90.10	97.55	51.25	88.89	63.61
10	7	80.44	90.20	75.00	91.48	39.72
12	8	97.16	103.71	72.50	93.52	63.33
13	7	91.28	83.53	81.25	99.81	92.22

[22]:
```
#uma outra função util é a fillna() que preenche todos os missing values com um␣
 ↪valor dado por nós

df.fillna(0, inplace=True)
df.head(10)
```

[22]:

	Prefix	Assignment	Tutorial	Midterm	TakeHome	Final
0	5	57.14	34.09	64.38	51.48	52.50
1	8	95.05	105.49	67.50	99.07	68.33
2	8	83.70	83.17	0.00	63.15	48.89
3	7	0.00	0.00	49.38	105.93	80.56
4	8	91.32	93.64	95.00	107.41	73.89
5	7	95.00	92.58	93.12	97.78	68.06
6	8	95.05	102.99	56.25	99.07	50.00
7	7	72.85	86.85	60.00	0.00	56.11
8	8	84.26	93.10	47.50	18.52	50.83
9	7	90.10	97.55	51.25	88.89	63.61

[ ]:
```
#podemos usar o na_filter para acabar com os espaços em branco, caso ele seja␣
 ↪um valor de interesse real.
#em dados sem N/A, passar um na_filter=False pode melhorar a performance de␣
 ↪leitura de um arquivo muito grande
```

[34]:
```
df = pd.read_csv('resources/week-2/datasets/log.csv')
df.head(20)
```

[34]:

	time	user	video	playback position	paused	volume
0	1469974424	cheryl	intro.html	5	False	10.0
1	1469974454	cheryl	intro.html	6	NaN	NaN
2	1469974544	cheryl	intro.html	9	NaN	NaN
3	1469974574	cheryl	intro.html	10	NaN	NaN
4	1469977514	bob	intro.html	1	NaN	NaN

```
5   1469977544     bob     intro.html                1    NaN     NaN
6   1469977574     bob     intro.html                1    NaN     NaN
7   1469977604     bob     intro.html                1    NaN     NaN
8   1469974604  cheryl     intro.html               11    NaN     NaN
9   1469974694  cheryl     intro.html               14    NaN     NaN
10  1469974724  cheryl     intro.html               15    NaN     NaN
11  1469974454     sue  advanced.html               24    NaN     NaN
12  1469974524     sue  advanced.html               25    NaN     NaN
13  1469974424     sue  advanced.html               23  False    10.0
14  1469974554     sue  advanced.html               26    NaN     NaN
15  1469974624     sue  advanced.html               27    NaN     NaN
16  1469974654     sue  advanced.html               28    NaN     5.0
17  1469974724     sue  advanced.html               29    NaN     NaN
18  1469974484  cheryl     intro.html                7    NaN     NaN
19  1469974514  cheryl     intro.html                8    NaN     NaN
```

[35]:
```
#o método parameter()
#os dois valores fill mais comuns são o ffill (foward) e o bfill(backwards). o␣
 ↪fFill preenche a célula com o
#valor da linha anterior. o bFill é ao contrário, preenche uma célula com o␣
 ↪valor da célula seguinte
#note que para que o efeito desejado seja alcançado, o dataset precisa ser␣
 ↪colocado em ordem

#no pandas nós podemos organizar pelo index ou valores. aqui vamos promover o␣
 ↪timestamp para indexador e então organziar por ele

df = df.set_index('time')
df = df.sort_index()
df.head(20)
```

[35]:
```
              user          video  playback position paused  volume
time
1469974424  cheryl     intro.html                  5  False    10.0
1469974424     sue  advanced.html                 23  False    10.0
1469974454  cheryl     intro.html                  6    NaN     NaN
1469974454     sue  advanced.html                 24    NaN     NaN
1469974484  cheryl     intro.html                  7    NaN     NaN
1469974514  cheryl     intro.html                  8    NaN     NaN
1469974524     sue  advanced.html                 25    NaN     NaN
1469974544  cheryl     intro.html                  9    NaN     NaN
1469974554     sue  advanced.html                 26    NaN     NaN
1469974574  cheryl     intro.html                 10    NaN     NaN
1469974604  cheryl     intro.html                 11    NaN     NaN
1469974624     sue  advanced.html                 27    NaN     NaN
1469974634  cheryl     intro.html                 12    NaN     NaN
1469974654     sue  advanced.html                 28    NaN     5.0
1469974664  cheryl     intro.html                 13    NaN     NaN
```

```
1469974694  cheryl     intro.html                14    NaN     NaN
1469974724  cheryl     intro.html                15    NaN     NaN
1469974724     sue  advanced.html                29    NaN     NaN
1469974754     sue  advanced.html                30    NaN     NaN
1469974824     sue  advanced.html                31    NaN     NaN
```

[36]:
```
#notamos que um só index é compartilhado por mais de um usiário, isso não é o␣
 ↪ideal
#agora vamos resetar o index e usar indexação multi-level com o time e user␣
 ↪name
#e daí promover o user para um indexador de segundo nível

df = df.reset_index()
df = df.set_index(['time', 'user'])
df.head(20)
```

[36]:
```
                          video  playback position paused  volume
time       user
1469974424 cheryl     intro.html                  5  False    10.0
           sue     advanced.html                 23  False    10.0
1469974454 cheryl     intro.html                  6    NaN     NaN
           sue     advanced.html                 24    NaN     NaN
1469974484 cheryl     intro.html                  7    NaN     NaN
1469974514 cheryl     intro.html                  8    NaN     NaN
1469974524 sue     advanced.html                 25    NaN     NaN
1469974544 cheryl     intro.html                  9    NaN     NaN
1469974554 sue     advanced.html                 26    NaN     NaN
1469974574 cheryl     intro.html                 10    NaN     NaN
1469974604 cheryl     intro.html                 11    NaN     NaN
1469974624 sue     advanced.html                 27    NaN     NaN
1469974634 cheryl     intro.html                 12    NaN     NaN
1469974654 sue     advanced.html                 28    NaN     5.0
1469974664 cheryl     intro.html                 13    NaN     NaN
1469974694 cheryl     intro.html                 14    NaN     NaN
1469974724 cheryl     intro.html                 15    NaN     NaN
           sue     advanced.html                 29    NaN     NaN
1469974754 sue     advanced.html                 30    NaN     NaN
1469974824 sue     advanced.html                 31    NaN     NaN
```

[37]:
```
#agora que ordedamos o dataset, vamos usar o fill para preencher os dados␣
 ↪faltando usando o ffill
df = df.fillna(method='ffill')
df.head(10)
```

[37]:
```
                          video  playback position  paused  volume
time       user
1469974424 cheryl     intro.html                  5   False    10.0
           sue     advanced.html                 23   False    10.0
1469974454 cheryl     intro.html                  6   False    10.0
```

```
           sue     advanced.html                 24   False    10.0
1469974484 cheryl     intro.html                  7   False    10.0
1469974514 cheryl     intro.html                  8   False    10.0
1469974524 sue      advanced.html                25   False    10.0
1469974544 cheryl     intro.html                  9   False    10.0
1469974554 sue      advanced.html                26   False    10.0
1469974574 cheryl     intro.html                 10   False    10.0
```

[64]:
```
#ainda podemos usar o fill-in para substituir valores com a função replace().
 ↪ela permite substituir de várias maneiras:
#value-to-value, lista, dicionário, regex
df = pd.DataFrame({'A': [3, 2, 1, 5, 4],
                   'B': [1, 7, 1, 3, 1],
                   'C': ['a', 'b', 'c', 'd', 'e']})
df.head()
```

[64]:
```
   A  B  C
0  3  1  a
1  2  7  b
2  1  1  c
3  5  3  d
4  4  1  e
```

[62]:
```
#podemos substituir os 1's com 100's usando o value-to-value
df.replace(1, 7) #olha, substituis os 1's com 100's
```

[62]:
```
   A  B  C
0  6  7  a
1  2  7  b
2  7  7  c
3  5  0  d
4  4  7  e
```

[65]:
```
#agora vamos tentar pelas listas
#queremos trocar os 1 por 100 e 3 por 300

df.replace([1, 3], [100, 300])
```

[65]:
```
     A    B  C
0  300  100  a
1    2    7  b
2  100  100  c
3    5  300  d
4    4  100  e
```

[69]:
```
#o replacement também suporta regex
df = pd.read_csv('resources/week-2/datasets/log.csv')
df
```

[69]:
```
          time    user          video  playback position paused  volume
0   1469974424  cheryl     intro.html                  5  False    10.0
```

1	1469974454	cheryl	intro.html	6	NaN	NaN
2	1469974544	cheryl	intro.html	9	NaN	NaN
3	1469974574	cheryl	intro.html	10	NaN	NaN
4	1469977514	bob	intro.html	1	NaN	NaN
5	1469977544	bob	intro.html	1	NaN	NaN
6	1469977574	bob	intro.html	1	NaN	NaN
7	1469977604	bob	intro.html	1	NaN	NaN
8	1469974604	cheryl	intro.html	11	NaN	NaN
9	1469974694	cheryl	intro.html	14	NaN	NaN
10	1469974724	cheryl	intro.html	15	NaN	NaN
11	1469974454	sue	advanced.html	24	NaN	NaN
12	1469974524	sue	advanced.html	25	NaN	NaN
13	1469974424	sue	advanced.html	23	False	10.0
14	1469974554	sue	advanced.html	26	NaN	NaN
15	1469974624	sue	advanced.html	27	NaN	NaN
16	1469974654	sue	advanced.html	28	NaN	5.0
17	1469974724	sue	advanced.html	29	NaN	NaN
18	1469974484	cheryl	intro.html	7	NaN	NaN
19	1469974514	cheryl	intro.html	8	NaN	NaN
20	1469974754	sue	advanced.html	30	NaN	NaN
21	1469974824	sue	advanced.html	31	NaN	NaN
22	1469974854	sue	advanced.html	32	NaN	NaN
23	1469974924	sue	advanced.html	33	NaN	NaN
24	1469977424	bob	intro.html	1	True	10.0
25	1469977454	bob	intro.html	1	NaN	NaN
26	1469977484	bob	intro.html	1	NaN	NaN
27	1469977634	bob	intro.html	1	NaN	NaN
28	1469977664	bob	intro.html	1	NaN	NaN
29	1469974634	cheryl	intro.html	12	NaN	NaN
30	1469974664	cheryl	intro.html	13	NaN	NaN
31	1469977694	bob	intro.html	1	NaN	NaN
32	1469977724	bob	intro.html	1	NaN	NaN

[4]:
```
#para substituir usando o regex, fazmos o primeiro parametro com o padrão do␣
 ↪regex
#que queremos combinar, o segundo parâmetro que desejamos emitir com a␣
 ↪combinação
#dai passamos o terceiro parâmetro como regex=True

#queremos aqui capturar tudo que é HTML e substituir por 'webpage'

df.replace(to_replace='.*.html$', value='webpage', regex=True)

#qualquer número de caracteres terminados com html substitui por 'webpage'
```

```
␣
↪-------------------------------------------------------------------------
```

```
NameError                                 Traceback (most recent call␣
↪last)

<ipython-input-4-2a72d0bcd6c0> in <module>
  5 #queremos aqui capturar tudo que é HTML e substituir por 'webpage'
  6
----> 7 df.replace(to_replace='.*.html$', value='webpage', regex=True)
  8
  9 #qualquer número de caracteres terminados com html substitui por␣
↪'webpage'


NameError: name 'df' is not defined
```

1 manipulating a data frame

[26]:
```
df = pd.read_csv('resources/week-2/datasets/presidents.csv')
df.head()
```

[26]:
```
   #          President              Born      Age atstart of presidency  \
0  1  George Washington  Feb 22, 1732[a]  57ãyears, 67ãdaysApr 30, 1789
1  2         John Adams  Oct 30, 1735[a]  61ãyears, 125ãdaysMar 4, 1797
2  3   Thomas Jefferson  Apr 13, 1743[a]  57ãyears, 325ãdaysMar 4, 1801
3  4      James Madison  Mar 16, 1751[a]  57ãyears, 353ãdaysMar 4, 1809
4  5       James Monroe     Apr 28, 1758  58ãyears, 310ãdaysMar 4, 1817

         Age atend of presidency Post-presidencytimespan          Died  \
0   65ãyears, 10ãdaysMar 4, 1797       2ãyears, 285ãdays  Dec 14, 1799
1  65ãyears, 125ãdaysMar 4, 1801      25ãyears, 122ãdays   Jul 4, 1826
2  65ãyears, 325ãdaysMar 4, 1809      17ãyears, 122ãdays   Jul 4, 1826
3  65ãyears, 353ãdaysMar 4, 1817      19ãyears, 116ãdays  Jun 28, 1836
4  66ãyears, 310ãdaysMar 4, 1825       6ãyears, 122ãdays   Jul 4, 1831

                 Age
0  67ãyears, 295ãdays
1  90ãyears, 247ãdays
2   83ãyears, 82ãdays
3  85ãyears, 104ãdays
4   73ãyears, 67ãdays
```

[6]:
```
#primeiro vamos dividir o nome do presidente em nome/sobrenome
#criando uma cópia da coluna nome
df['Firstname'] = df['President']
df.head()
```

[6]:
```
   #          President             Born      Age atstart of presidency  \
0  1  George Washington  Feb 22, 1732[a]  57ãyears, 67ãdaysApr 30, 1789
1  2         John Adams  Oct 30, 1735[a]  61ãyears, 125ãdaysMar 4, 1797
2  3   Thomas Jefferson  Apr 13, 1743[a]  57ãyears, 325ãdaysMar 4, 1801
3  4      James Madison  Mar 16, 1751[a]  57ãyears, 353ãdaysMar 4, 1809
4  5       James Monroe     Apr 28, 1758  58ãyears, 310ãdaysMar 4, 1817

         Age atend of presidency Post-presidencytimespan          Died  \
0   65ãyears, 10ãdaysMar 4, 1797       2ãyears, 285ãdays  Dec 14, 1799
1  65ãyears, 125ãdaysMar 4, 1801      25ãyears, 122ãdays   Jul 4, 1826
2  65ãyears, 325ãdaysMar 4, 1809      17ãyears, 122ãdays   Jul 4, 1826
3  65ãyears, 353ãdaysMar 4, 1817      19ãyears, 116ãdays  Jun 28, 1836
4  66ãyears, 310ãdaysMar 4, 1825       6ãyears, 122ãdays   Jul 4, 1831

                  Age          Firstname
0  67ãyears, 295ãdays  George Washington
1  90ãyears, 247ãdays         John Adams
2   83ãyears, 82ãdays   Thomas Jefferson
3  85ãyears, 104ãdays      James Madison
4   73ãyears, 67ãdays       James Monroe
```

[11]:
```
#agora quebrando a string do nome da coluna Firstname e mantendo só o primeiro␣
 ↪nome usando o Regex
df['Firstname'] = df['Firstname'].replace('[ ].*', '', regex = True)
#um espaço em branco, seguido por qualquer caractere, substitui por nada
df.head()
```

[11]:
```
   #          President             Born      Age atstart of presidency  \
0  1  George Washington  Feb 22, 1732[a]  57ãyears, 67ãdaysApr 30, 1789
1  2         John Adams  Oct 30, 1735[a]  61ãyears, 125ãdaysMar 4, 1797
2  3   Thomas Jefferson  Apr 13, 1743[a]  57ãyears, 325ãdaysMar 4, 1801
3  4      James Madison  Mar 16, 1751[a]  57ãyears, 353ãdaysMar 4, 1809
4  5       James Monroe     Apr 28, 1758  58ãyears, 310ãdaysMar 4, 1817

         Age atend of presidency Post-presidencytimespan          Died  \
0   65ãyears, 10ãdaysMar 4, 1797       2ãyears, 285ãdays  Dec 14, 1799
1  65ãyears, 125ãdaysMar 4, 1801      25ãyears, 122ãdays   Jul 4, 1826
2  65ãyears, 325ãdaysMar 4, 1809      17ãyears, 122ãdays   Jul 4, 1826
3  65ãyears, 353ãdaysMar 4, 1817      19ãyears, 116ãdays  Jun 28, 1836
4  66ãyears, 310ãdaysMar 4, 1825       6ãyears, 122ãdays   Jul 4, 1831

                  Age Firstname
0  67ãyears, 295ãdays    George
1  90ãyears, 247ãdays      John
2   83ãyears, 82ãdays    Thomas
3  85ãyears, 104ãdays     James
4   73ãyears, 67ãdays     James
```

[12]:
```
#isso funciona mas tem uma maneira mais fácil e bonitinha de fazer
#vamos dropar a coluna Firstname e fazer de novo usando a função apply()
del(df['Firstname'])
```

[13]:
```
#a função apply() vai pegar uma função arbitrária que escrevemos e aplciá-la na␣
 ↪Série ou Dataframe
#em todas as linhas ou colunas
#o que segue é uma função que faz um split numa string usando uma única linha␣
 ↪de código

def splitname(row):
    #extraindo o primeiro nome e criando uma nova entrada na Série
    row['Firstname'] = row['President'].split(" ")[0]
    #agora com o último nome
    row['Lastname'] = row['President'].split(" ")[-1]
    #por fim retornamos a row e o pandas vai dar um .apply()
    return row
```

[21]:
```
#e se aplicarmos essa função no DataFrame indicando que queremos faê-lo por␣
 ↪colunas
df = df.apply(splitname, axis = 'columns')
df.head()
```

[21]:
```
   #          President              Born      Age atstart of presidency  \
0  1  George Washington  Feb 22, 1732[a]  57ãyears, 67ãdaysApr 30, 1789
1  2         John Adams  Oct 30, 1735[a]  61ãyears, 125ãdaysMar 4, 1797
2  3   Thomas Jefferson  Apr 13, 1743[a]  57ãyears, 325ãdaysMar 4, 1801
3  4      James Madison  Mar 16, 1751[a]  57ãyears, 353ãdaysMar 4, 1809
4  5       James Monroe     Apr 28, 1758  58ãyears, 310ãdaysMar 4, 1817

         Age atend of presidency Post-presidencytimespan          Died  \
0   65ãyears, 10ãdaysMar 4, 1797       2ãyears, 285ãdays  Dec 14, 1799
1  65ãyears, 125ãdaysMar 4, 1801      25ãyears, 122ãdays   Jul 4, 1826
2  65ãyears, 325ãdaysMar 4, 1809      17ãyears, 122ãdays   Jul 4, 1826
3  65ãyears, 353ãdaysMar 4, 1817      19ãyears, 116ãdays  Jun 28, 1836
4  66ãyears, 310ãdaysMar 4, 1825       6ãyears, 122ãdays   Jul 4, 1831

                 Age Firstname    Lastname
0  67ãyears, 295ãdays    George  Washington
1  90ãyears, 247ãdays      John       Adams
2   83ãyears, 82ãdays    Thomas   Jefferson
3  85ãyears, 104ãdays     James     Madison
4   73ãyears, 67ãdays     James      Monroe
```

[23]:
```
#tem um outro jeito de fazer o mesmo trabalho usando a função extract()
#vamos dropar essas colunas Firstname/Lastname que criamos e usar esse terceiro␣
 ↪método
del(df['Firstname'])
del(df['Lastname'])
```

```
      ␣
↪---------------------------------------------------------------------------

      KeyError                                  Traceback (most recent call␣
↪last)

      /opt/conda/lib/python3.7/site-packages/pandas/core/indexes/base.py in␣
↪get_loc(self, key, method, tolerance)
     2889             try:
   -> 2890                 return self._engine.get_loc(key)
     2891             except KeyError:


      pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()


      pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()


      pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.
↪PyObjectHashTable.get_item()


      pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.
↪PyObjectHashTable.get_item()


      KeyError: 'Firstname'


  During handling of the above exception, another exception occurred:


      KeyError                                  Traceback (most recent call␣
↪last)

      <ipython-input-23-005055761695> in <module>
        1 #tem um outro jeito de fazer o mesmo trabalho usando a função␣
↪extract()
        2 #vamos dropar essas colunas Firstname/Lastname que criamos e usar␣
↪esse terceiro método
  ----> 3 del(df['Firstname'])
        4 del(df['Lastname'])
```

```
      /opt/conda/lib/python3.7/site-packages/pandas/core/generic.py in␣
↪__delitem__(self, key)
     3515             # there was no match, this call should raise the␣
↪appropriate
     3516             # exception:
   -> 3517             self._data.delete(key)
     3518
     3519         # delete from the caches


      /opt/conda/lib/python3.7/site-packages/pandas/core/internals/managers.py␣
↪in delete(self, item)
      993         Delete selected item (items if non-unique) in-place.
      994         """
   --> 995         indexer = self.items.get_loc(item)
      996
      997         is_deleted = np.zeros(self.shape[0], dtype=np.bool_)


      /opt/conda/lib/python3.7/site-packages/pandas/core/indexes/base.py in␣
↪get_loc(self, key, method, tolerance)
     2890                 return self._engine.get_loc(key)
     2891             except KeyError:
   -> 2892                 return self._engine.get_loc(self.
↪_maybe_cast_indexer(key))
     2893         indexer = self.get_indexer([key], method=method,␣
↪tolerance=tolerance)
     2894         if indexer.ndim > 1 or indexer.size > 1:


      pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()


      pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()


      pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.
↪PyObjectHashTable.get_item()


      pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.
↪PyObjectHashTable.get_item()


      KeyError: 'Firstname'
```

[25]:
```
#o extract pega uma expressão regular para funcionar e requer que você capture␣
 ↪apenas itens que correspondem aos
#grupos que você deseja inserie, como nome e sobrenome

#esse é o padrão para capturar nome e sobrenome, se tiver um nome do meio ele␣
 ↪ignora, pega só primeiro e último
pattern="""(^[\w]*)(?:.* )([\w]*$)"""

#a função extract é construida dentro do atributo str da Série objeto, então␣
 ↪chamamos ela usando Series.str.extract(pattern)
df['President'].str.extract(pattern).head()
```

[25]:
```
         0           1
0   George  Washington
1     John       Adams
2   Thomas   Jefferson
3    James     Madison
4    James      Monroe
```

[ ]:
```
#daí ainda podemos criar nomes para os grupos
```

[30]:
```
pattern="""(?P<Firstname>^[\w]*)(?:.* )(?P<Lastname>[\w]*$)"""
names = df['President'].str.extract(pattern)
names.head()
```

[30]:
```
  Firstname    Lastname
0    George  Washington
1      John       Adams
2    Thomas   Jefferson
3     James     Madison
4     James      Monroe
```

[31]:
```
#daí podemos copiar esses dados para dentro do nosso dataframe original
df['Firstname'] = names['Firstname']
df['Lastname'] = names['Lastname']
df.head()
```

[31]:
```
   #          President             Born      Age atstart of presidency  \
0  1  George Washington  Feb 22, 1732[a]  57ãyears, 67ãdaysApr 30, 1789
1  2         John Adams  Oct 30, 1735[a]  61ãyears, 125ãdaysMar 4, 1797
2  3   Thomas Jefferson  Apr 13, 1743[a]  57ãyears, 325ãdaysMar 4, 1801
3  4      James Madison  Mar 16, 1751[a]  57ãyears, 353ãdaysMar 4, 1809
4  5       James Monroe     Apr 28, 1758  58ãyears, 310ãdaysMar 4, 1817

         Age atend of presidency Post-presidencytimespan          Died  \
0   65ãyears, 10ãdaysMar 4, 1797       2ãyears, 285ãdays  Dec 14, 1799
1  65ãyears, 125ãdaysMar 4, 1801      25ãyears, 122ãdays   Jul 4, 1826
2  65ãyears, 325ãdaysMar 4, 1809      17ãyears, 122ãdays   Jul 4, 1826
3  65ãyears, 353ãdaysMar 4, 1817      19ãyears, 116ãdays  Jun 28, 1836
4  66ãyears, 310ãdaysMar 4, 1825       6ãyears, 122ãdays   Jul 4, 1831
```

```
                   Age Firstname    Lastname
0  67ãyears, 295ãdays    George  Washington
1  90ãyears, 247ãdays      John       Adams
2   83ãyears, 82ãdays    Thomas   Jefferson
3  85ãyears, 104ãdays     James     Madison
4   73ãyears, 67ãdays     James      Monroe
```

[ ]:
```
#agora limpando a  coluna Born, removeremos tudo que não é dia/mês/ano
```

[37]:
```
df['Born'] = df['Born'].str.extract('([\w]{3} [\w]{1,2}, [\w]{4})')
df['Born'].head()
```

[37]:
```
0    Feb 22, 1732
1    Oct 30, 1735
2    Apr 13, 1743
3    Mar 16, 1751
4    Apr 28, 1758
Name: Born, dtype: object
```

[38]:
```
df.head()
```

[38]:
```
   #          President          Born      Age atstart of presidency  \
0  1  George Washington  Feb 22, 1732  57ãyears, 67ãdaysApr 30, 1789
1  2         John Adams  Oct 30, 1735  61ãyears, 125ãdaysMar 4, 1797
2  3   Thomas Jefferson  Apr 13, 1743  57ãyears, 325ãdaysMar 4, 1801
3  4      James Madison  Mar 16, 1751  57ãyears, 353ãdaysMar 4, 1809
4  5       James Monroe  Apr 28, 1758  58ãyears, 310ãdaysMar 4, 1817

         Age atend of presidency Post-presidencytimespan          Died  \
0   65ãyears, 10ãdaysMar 4, 1797       2ãyears, 285ãdays  Dec 14, 1799
1  65ãyears, 125ãdaysMar 4, 1801      25ãyears, 122ãdays   Jul 4, 1826
2  65ãyears, 325ãdaysMar 4, 1809      17ãyears, 122ãdays   Jul 4, 1826
3  65ãyears, 353ãdaysMar 4, 1817      19ãyears, 116ãdays  Jun 28, 1836
4  66ãyears, 310ãdaysMar 4, 1825       6ãyears, 122ãdays   Jul 4, 1831

                   Age Firstname    Lastname
0  67ãyears, 295ãdays    George  Washington
1  90ãyears, 247ãdays      John       Adams
2   83ãyears, 82ãdays    Thomas   Jefferson
3  85ãyears, 104ãdays     James     Madison
4   73ãyears, 67ãdays     James      Monroe
```

[ ]:

[ ]:

[ ]:

[ ]: