Data Cleaning Pandas e NumPy

Um livro em particular pode ter apenas uma data de publicação. Portanto, precisamos fazer o seguinte:

- Remova as datas extras em parênteses quadrados, onde quer que presentes: 1879 [1878]
- Converta as faixas de data para a sua "data de início", onde quer que a presente: 1860-63; 1839, 38-54
- Remova completamente as datas sobre as as que não temos certeza e substitua-as por NumPy's : [1897?] `NaN`
- Converta a sequência em valor do NumPy `nan` `NaN`

Sintetizando esses padrões, podemos realmente aproveitar uma única expressão regular para extrair o ano de publicação:

```
>>> df['Identifier'].is_unique
True
```

Os cientistas de dados gastam uma grande parte de seu tempo limpando conjuntos de dados e levando-os a uma forma com a qual eles podem trabalhar.

De fato, muitos cientistas de dados argumentam que as etapas iniciais de obtenção e limpeza de dados constituem 80% do trabalho.

É importante ser capaz de lidar com dados confusos, quer isso signifique valores perdidos, formatação inconsistente, registros malformados ou outliers sem sentido.

Neste tutorial, aproveitaremos as bibliotecas Pandas e NumPy do Python para limpar dados.

Vamos cobrir o seguinte:

- Dropando colunas desnecessárias em um `DataFrame`
- Alterando o índice de um `DataFrame`
- Usando métodos para limpar colunas `.str()`
- Usando a função para limpar todo o conjunto de dados, em termos de elementos `DataFrame.applymap()`
- Renomeando colunas para um conjunto mais reconhecível de rótulos
- Pulando linhas desnecessárias em um arquivo CSV

Conjuntos de dados:

- BL-Flickr-Images-Book.csv - Um arquivo CSV contendo informações sobre livros da Biblioteca Britânica
- university_towns.txt - Um arquivo de texto contendo nomes de cidades universitárias em todos os estados dos EUA
- olimpíadas.csv - Um arquivo CSV resumindo a participação de todos os países nos Jogos Olímpicos de Verão e Inverno

```
>>> import pandas as pd
>>> import numpy as np
```

Soltando colunas em um DataFrame

Muitas vezes, você verá que nem todas as categorias de dados em um conjunto de dados são úteis para você. Por exemplo, você pode ter um conjunto de dados contendo informações dos alunos (nome, nota, padrão, nomes dos pais e endereço), mas quer se concentrar em analisar as notas dos alunos.

Neste caso, as categorias de endereço ou nomes dos pais não são importantes para você. A retenção dessas categorias desnecessárias ocupará espaço desnecessário e, potencialmente, também perderá o tempo de execução.

Pandas fornece uma maneira útil de remover colunas ou linhas indesejadas de uma função com a função `drop()`. Vamos olhar para um exemplo simples onde deixamos cair uma série de colunas de um DataFrame.

```
df = pd.read_csv('Datasets/BL-Flickr-Images-Book.csv')
df.head()

Identifier             Edition Statement      Place of Publication  \
0         206                           NaN                    London
1         216                           NaN  London; Virtue & Yorston
2         218                           NaN                    London
3         472                           NaN                    London
4         480  A new edition, revised, etc.                    London

  Date of Publication             Publisher  \
0         1879 [1878]       S. Tinsley & Co.
1                1868          Virtue & Co.
2                1869  Bradbury, Evans & Co.
3                1851         James Darling
4                1857   Wertheim & Macintosh

                                          Title     Author  \
0                 Walter Forbes. [A novel.] By A. A      A. A.
1  All for Greed. [A novel. The dedication signed...  A., A. A.
2  Love the Avenger. By the author of “All for Gr...  A., A. A.
3  Welsh Sketches, chiefly ecclesiastical, to the...  A., E. S.
```

```
4  [The World in which I live, and my place in it...  A., E. S.

                                Contributors  Corporate Author  \
0                              FORBES, Walter.               NaN
1  BLAZE DE BURY, Marie Pauline Rose - Baroness               NaN
2  BLAZE DE BURY, Marie Pauline Rose - Baroness               NaN
3                   Appleyard, Ernest Silvanus.               NaN
4                           BROOME, John Henry.               NaN

   Corporate Contributors Former owner  Engraver Issuance type  \
0                     NaN          NaN       NaN   monographic
1                     NaN          NaN       NaN   monographic
2                     NaN          NaN       NaN   monographic
3                     NaN          NaN       NaN   monographic
4                     NaN          NaN       NaN   monographic

                                       Flickr URL  \
0  http://www.flickr.com/photos/britishlibrary/ta...
1  http://www.flickr.com/photos/britishlibrary/ta...
2  http://www.flickr.com/photos/britishlibrary/ta...
3  http://www.flickr.com/photos/britishlibrary/ta...
4  http://www.flickr.com/photos/britishlibrary/ta...

                           Shelfmarks
0    British Library HMNTS 12641.b.30.
1    British Library HMNTS 12626.cc.2.
2    British Library HMNTS 12625.dd.1.
3    British Library HMNTS 10369.bbb.15.
4    British Library HMNTS 9007.d.28.
```

Quando olhamos para as cinco primeiras entradas usando o método, podemos ver que um punhado de colunas fornecem informações acessórias que seriam úteis para a biblioteca, mas não são muito descritivas dos próprios livros:

Podemos dropar essas colunas da seguinte maneira:

```
>>> to_drop = ['Edition Statement',
...            'Corporate Author',
...            'Corporate Contributors',
...            'Former owner',
...            'Engraver',
...            'Contributors',
...            'Issuance type',
...            'Shelfmarks']

>>> df.drop(to_drop, inplace=True, axis=1)
```

Acima, definimos uma lista que contém os nomes de todas as colunas que queremos dropar.

Quando inspecionarmos novamente, veremos que as colunas indesejadas foram removidas:

```
>>> df.head()

   Identifier      Place of Publication Date of Publication  \
0         206                    London         1879 [1878]
1         216  London; Virtue & Yorston                1868
```

```
2         218                   London               1869
3         472                   London               1851
4         480                   London               1857

               Publisher                                                Title  \
0       S. Tinsley & Co.                  Walter Forbes. [A novel.] By A. A
1           Virtue & Co.  All for Greed. [A novel. The dedication signed...
2  Bradbury, Evans & Co.  Love the Avenger. By the author of “All for Gr...
3          James Darling  Welsh Sketches, chiefly ecclesiastical, to the...
4   Wertheim & Macintosh  [The World in which I live, and my place in it...

      Author                                          Flickr URL
0      A. A.  http://www.flickr.com/photos/britishlibrary/ta...
1  A., A. A.  http://www.flickr.com/photos/britishlibrary/ta...
2  A., A. A.  http://www.flickr.com/photos/britishlibrary/ta...
3  A., E. S.  http://www.flickr.com/photos/britishlibrary/ta...
4  A., E. S.  http://www.flickr.com/photos/britishlibrary/ta...
```

Alternativamente, também poderíamos remover as colunas passando-as diretamente para o parâmetro em vez de especificar separadamente os rótulos a serem removidos e o eixo onde os Pandas devem procurar os rótulos:

```
>>> df.drop(columns=to_drop, inplace=True)
```

Alterando o Índice de um DataFrame

Pandas amplia a funcionalidade das matrizes NumPy para permitir cortes e rotulagem mais versáteis.

Em muitos casos, é útil usar um campo de identificação exclusivamente valorizado dos dados como seu índice.

Por exemplo, no conjunto de dados usado na seção anterior, pode-se esperar que, quando um bibliotecário pesquisa um registro, eles possam inserir o identificador único (valores na coluna) para um livro:

```
>>> df['Identifier'].is_unique
True
```

Vamos substituir o índice existente por esta coluna usando:

```
>>> df = df.set_index('Identifier')
>>> df.head()
                Place of Publication Date of Publication  \
206                           London         1879 [1878]
216         London; Virtue & Yorston                1868
218                           London                1869
472                           London                1851
480                           London                1857

                        Publisher  \
```

```
206              S. Tinsley & Co.
216                  Virtue & Co.
218         Bradbury, Evans & Co.
472                 James Darling
480          Wertheim & Macintosh

                                                    Title     Author  \
206                        Walter Forbes. [A novel.] By A. A      A. A.
216         All for Greed. [A novel. The dedication signed...  A., A. A.
218         Love the Avenger. By the author of “All for Gr...  A., A. A.
472         Welsh Sketches, chiefly ecclesiastical, to the...  A., E. S.
480         [The World in which I live, and my place in it...  A., E. S.

                                                  Flickr URL
206         http://www.flickr.com/photos/britishlibrary/ta...
216         http://www.flickr.com/photos/britishlibrary/ta...
218         http://www.flickr.com/photos/britishlibrary/ta...
472         http://www.flickr.com/photos/britishlibrary/ta...
480         http://www.flickr.com/photos/britishlibrary/ta...
```

```
>>> df.loc[206]
Place of Publication                                          London
Date of Publication                                      1879 [1878]
Publisher                                             S. Tinsley & Co.
Title                                  Walter Forbes. [A novel.] By A. A
Author                                                         A. A.
Flickr URL             http://www.flickr.com/photos/britishlibrary/ta...
Name: 206, dtype: object
```

Você deve ter notado que reatribuímos a variável ao objeto devolvido pelo método set_index. Isso porque, por padrão, **o método retorna uma cópia modificada do nosso objeto e não faz as alterações diretamente no objeto**. Podemos evitar isso definindo o parâmetro:df = df.set_index(...)inplace

```
df.set_index('Identifier', inplace=True)
```

Arrumando campos nos dados

Até agora, removemos colunas desnecessárias e mudamos o índice do nosso para algo mais sensato. Nesta seção, limparemos colunas específicas e as levaremos a um formato uniforme para obter uma melhor compreensão do conjunto de dados e impor consistência.

Após a inspeção, todos os tipos de dados são atualmente o dtype, que é aproximadamente análogo ao em Python nativo.

Encapsula qualquer campo que não possa ser perfeitamente adequado como dados numéricos ou categóricos. Isso faz sentido, já que estamos trabalhando com dados que inicialmente são um monte de strings confusas:

```
>>> df.get_dtype_counts()
object    6
```

Um campo onde faz sentido impor um valor numérico é a data de publicação para que possamos fazer cálculos na estrada:

```
>>> df.loc[1905:, 'Date of Publication'].head(10)
Identifier
1905           1888
1929    1839, 38-54
2836        [1897?]
2854           1865
2956        1860-63
2957           1873
3017           1866
3131           1899
4598           1814
4884           1820
Name: Date of Publication, dtype: object
```

Um livro em particular pode ter apenas uma data de publicação. Portanto, precisamos fazer o seguinte:

- Remova as datas extras em parênteses quadrados, onde quer que presentes: 1879 [1878]
- Converta as faixas de data para a sua "data de início", onde quer que a presente: 1860-63; 1839, 38-54
- Remova completamente as datas sobre as as que não temos certeza e substitua-as por NumPy's : [1897?]`NaN`
- Converta a sequência em valor do NumPy`nan` `NaN`

Expressões Regulares

Sintetizando esses padrões, podemos realmente aproveitar uma única **expressão regular** para extrair o ano de publicação:

```
regex = r'^(\d{4})'
```

A expressão regular acima é feita para encontrar qualquer quatro dígitos no início de uma sequência, o que é suficiente para o nosso caso.

O acima é uma *sequência bruta* (o que significa que uma barra invertida não é mais um personagem de fuga), que é a prática padrão com expressões regulares.

O representa qualquer dígito, e repete esta regra quatro vezes. O personagem corresponde ao início de uma sequência, e os parênteses denotam um grupo de captura, que sinaliza para Pandas que queremos extrair essa parte do regex. (Queremos evitar casos em que comece a sequência.)`\d` `{4}` `^` `^` `[`

Vamos ver o que acontece quando executamos este regex em nosso conjunto de dados:

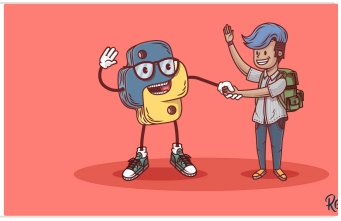
```
>>> extr = df['Date of Publication'].str.extract(r'^(\d{4})', expand=False)
>>> extr.head()
Identifier
206    1879
216    1868
218    1869
472    1851
480    1857
Name: Date of Publication, dtype: object
```

Expressões Regulares

Regular Expressions: Regexes in Python (Part 1) - Real Python

In this tutorial, you'll explore regular expressions, also known as regexes, in Python. A regex is a special sequence of characters that defines a pattern for complex string-matching functionality. Earlier

https://realpython.com/regex-python/



Tecnicamente, esta coluna ainda tem dtype, mas podemos facilmente obter sua versão numérica com:objectpd.to_numeric

```
>>> df['Date of Publication'] = pd.to_numeric(extr)
>>> df['Date of Publication'].dtype
dtype('float64')
```

Isso resulta em cerca de um em cada dez valores faltando, que é um pequeno preço a pagar por agora ser capaz de fazer cálculos sobre os valores válidos restantes:

```
>>> df['Date of Publication'].isnull().sum() / len(df)
0.11717147339205986
```

Combinando métodos com NumPy para Colunas Limpas

Este atributo é uma maneira de acessar operações rápidas de sequência de cordas em Pandas que imitam em grande parte operações em strings python nativas ou expressões regulares compiladas: `df['Date of Publication'].str.split().replace().capitalize()`

Para limpar o campo, podemos combinar métodos Pandas com a função do NumPy, que é basicamente uma forma vetorializada da macro do Excel. Tem a seguinte sintaxe:

```
>>> np.where(condition, then, else)
```

Aqui, ou é um objeto semelhante a uma matriz ou uma máscara booleana. É o valor a ser usado se avaliar , e é o valor a ser usado de outra forma. `conditionthenconditionTrueelse`

Essencialmente, pega cada elemento no objeto utilizado, verifica se esse elemento em particular avalia no contexto da condição, e retorna um contendo, dependendo do que se aplica. `.where()conditionTruendarraythenelse`

Ele pode ser aninhado em uma declaração se-then composto, permitindo calcular valores com base em múltiplas condições:

```
>>> np.where(condition1, x1,
        np.where(condition2, x2,
            np.where(condition3, x3, ...)))
```

Vamos fazer uso dessas duas funções para limpar, já que esta coluna tem objetos de corda. Aqui está o conteúdo da coluna:Place of Publication

```
>>> df['Place of Publication'].head(10)
Identifier
206                                 London
216               London; Virtue & Yorston
218                                 London
472                                 London
480                                 London
481                                 London
519                                 London
667     pp. 40. G. Bryan & Co: Oxford, 1898
874                                London]
1143                                London
Name: Place of Publication, dtype: object
```

Vemos que para algumas linhas, o local de publicação é cercado por outras informações desnecessárias. Se fôssemos olhar para mais valores, veríamos que este é o caso apenas de algumas linhas que têm seu lugar de publicação como 'Londres' ou 'Oxford'.

Vamos dar uma olhada em duas entradas específicas:

```
>>> df.loc[4157862]
Place of Publication                                  Newcastle-upon-Tyne
Date of Publication                                                  1867
Publisher                                                    T. Fordyce
Title                  Local Records; or, Historical Register of rema...
Author                                                       T.  Fordyce
Flickr URL             http://www.flickr.com/photos/britishlibrary/ta...
Name: 4157862, dtype: object

>>> df.loc[4159587]
Place of Publication                                  Newcastle upon Tyne
Date of Publication                                                  1834
Publisher                                                Mackenzie & Dent
Title                  An historical, topographical and descriptive v...
Author                                               E. (Eneas) Mackenzie
Flickr URL             http://www.flickr.com/photos/britishlibrary/ta...
Name: 4159587, dtype: object
```

Estes dois livros foram publicados no mesmo lugar, mas um tem hífens em nome do lugar, enquanto o outro não.

Para limpar esta coluna em uma varredura, podemos usar para obter uma máscara booleana. `str.contains()`

Limpamos a coluna da seguinte forma:

```
>>> pub = df['Place of Publication']
>>> london = pub.str.contains('London')
>>> london[:5]
Identifier
206    True
216    True
218    True
472    True
480    True
Name: Place of Publication, dtype: bool

>>> oxford = pub.str.contains('Oxford')
```

Nós os combinamos com:np.where

```
df['Place of Publication'] = np.where(london, 'London',
                                      np.where(oxford, 'Oxford',
                                               pub.str.replace('-', ' ')))

>>> df['Place of Publication'].head()
Identifier
206    London
216    London
218    London
472    London
480    London
Name: Place of Publication, dtype: object
```

Aqui, a função é chamada em uma estrutura aninhada, sendo um dos booleanos obtidos com . O método funciona de forma semelhante à palavra-chave incorporada usada para encontrar a ocorrência de uma entidade em uma iterável (ou substring em uma sequência). `np.where` `condition` `Series` `str.contains()` `contains()`

A substituição a ser usada é uma string representando nosso local de publicação desejado. Também substituímos hífens por um espaço e reatribuímos à coluna em nosso. `str.replace()` `DataFrame`

Embora haja mais dados sujos neste conjunto de dados, discutiremos apenas essas duas colunas por enquanto.

Vamos dar uma olhada nas cinco primeiras entradas, que parecem muito mais nítidas do que quando começamos:

```
>>> df.head()
          Place of Publication Date of Publication              Publisher  \
206                     London                1879        S. Tinsley & Co.
216                     London                1868           Virtue & Co.
218                     London                1869  Bradbury, Evans & Co.
472                     London                1851          James Darling
480                     London                1857   Wertheim & Macintosh

                                                  Title    Author  \
206                       Walter Forbes. [A novel.] By A. A        AA
216         All for Greed. [A novel. The dedication signed...   A. A A.
218         Love the Avenger. By the author of “All for Gr...   A. A A.
472         Welsh Sketches, chiefly ecclesiastical, to the...   E. S A.
480         [The World in which I live, and my place in it...   E. S A.

                                            Flickr URL
206         http://www.flickr.com/photos/britishlibrary/ta...
216         http://www.flickr.com/photos/britishlibrary/ta...
218         http://www.flickr.com/photos/britishlibrary/ta...
472         http://www.flickr.com/photos/britishlibrary/ta...
480         http://www.flickr.com/photos/britishlibrary/ta...
```

Limpeza de todo o conjunto de dados usando a funçãoapplymap

Em certas situações, você verá que a "sujeira" não está localizada em uma coluna, mas está mais espalhada.

Existem alguns casos em que seria útil aplicar uma função personalizada a cada célula ou elemento de um DataFrame. O método Pandas é semelhante à função de `map` embutido e simplesmente aplica uma função a todos os elementos em um . `.applymap()DataFrame`

Vamos dar uma olhada em um exemplo. Criaremos um arquivo "university_towns.txt": `DataFrame`

```
head Datasets/univerisity_towns.txt
Alabama[edit]
```

```
Auburn (Auburn University)[1]
Florence (University of North Alabama)
Jacksonville (Jacksonville State University)[2]
Livingston (University of West Alabama)[2]
Montevallo (University of Montevallo)[2]
Troy (Troy University)[2]
Tuscaloosa (University of Alabama, Stillman College, Shelton State)[3][4]
Tuskegee (Tuskegee University)[5]
Alaska[edit]
```

Vemos que temos nomes periódicos de estado seguidos pelas cidades universitárias daquele estado: . Se olharmos para a forma como os nomes de estado são escritos no arquivo, veremos que todos eles têm o substring "[editar]" neles. `StateA TownA1 TownA2 StateB TownB1 TownB2...`

Podemos aproveitar esse padrão criando uma *lista de* `tuplas (estado, cidade)` e embrulhando essa lista em um : `DataFrame`

```
>>> university_towns = []
>>> with open('Datasets/university_towns.txt') as file:
...     for line in file:
...         if '[edit]' in line:
...             # Remember this `state` until the next is found
...             state = line
...         else:
...             # Otherwise, we have a city; keep `state` as last-seen
...             university_towns.append((state, line))

>>> university_towns[:5]
[('Alabama[edit]\n', 'Auburn (Auburn University)[1]\n'),
 ('Alabama[edit]\n', 'Florence (University of North Alabama)\n'),
 ('Alabama[edit]\n', 'Jacksonville (Jacksonville State University)[2]\n'),
 ('Alabama[edit]\n', 'Livingston (University of West Alabama)[2]\n'),
 ('Alabama[edit]\n', 'Montevallo (University of Montevallo)[2]\n')]
```

Podemos embrulhar esta lista em um DataFrame e definir as colunas como "Estado" e "RegionName". Pandas pegarão cada elemento da lista e definirão o valor esquerdo e o valor certo. `State` `RegionName`

O DataFrame resultante é assim:

```
>>> towns_df = pd.DataFrame(university_towns,
...                         columns=['State', 'RegionName'])

>>> towns_df.head()
 State                                      RegionName
0  Alabama[edit]\n                   Auburn (Auburn University)[1]\n
1  Alabama[edit]\n            Florence (University of North Alabama)\n
2  Alabama[edit]\n  Jacksonville (Jacksonville State University)[2]\n
3  Alabama[edit]\n       Livingston (University of West Alabama)[2]\n
4  Alabama[edit]\n         Montevallo (University of Montevallo)[2]\n
```