Captura de dados na Web, expressões regulares e visualização de dados: fazendo tudo em Python

Um Pequeno Projeto do Mundo Real para Aprender Três Habilidades Científicas em Dados Inestimáveis

Will Koehrsen Seguir
28 de abril de 2018 · 7 minutos de leitura

Tal como acontece com projetos mais interessantes, este começou com uma pergunta simples meio seriamente: quanto custa pagar cinco minutos do tempo do meu presidente da faculdade? Depois de uma chance de conversar com o presidente da minha escola (CWRU), fiquei imaginando o quanto minha conversa me custou.

Minha busca levou a este artigo , que, juntamente com o salário do meu presidente, tinha esta tabela mostrando os salários dos presidentes de faculdades particulares em Ohio:

- Grant Cornwell, College of Wooster (left in 2015): $911,651
- Marvin Krislov, Oberlin College (left in 2016): $829,913
- Mark Roosevelt, Antioch College, (left in 2015): $507,672
- Laurie Joyner, Wittenberg University (left in 2015): $463,504
- Richard Giese, University of Mount Union (left in 2015): $453,800
- Sean Decatur,Kenyon College: $451,698
- Adam Weinberg, Denison University: $435,322
- Daniel Dibiasio, Ohio Northern University: $414,716
- Denvy Bowman, Capital University (left in 2016): $388,570
- Anne Steele, Muskingum University (left in 2016): $384,233
- Kathy Krendl, Otterbein University: $378,035
- Rockwell Jones, Ohio Wesleyan University: $366,625
- Robert Helmer, Baldwin Wallace University: $365,616
- Robert Huntington, Heidelberg University: $300,005
- Lori Varlotta, Hiram College: $293,336
- Joseph Bruno, Marietta College (left in 2016): $288,295
- W. Richard Merriman Jr., University of Mount Union (started in June 2015): $221,761

Embora eu pudesse ter encontrado a resposta para o meu presidente, (SPOILER ALERT, é US $ 48 / cinco minutos), e fiquei satisfeito, eu queria levar a ideia adiante usando esta tabela. Eu estava procurando por uma chance de praticar scraping de web e expressões regulares em Python e decidi que este era um grande projeto curto.

Embora quase certamente fosse mais rápido inserir manualmente os dados no Excel, eu não teria tido a oportunidade inestimável de praticar algumas habilidades! A ciência de dados trata da solução de problemas usando um conjunto diversificado de ferramentas, e a raspagem da web e expressões regulares são duas áreas nas quais preciso trabalhar (sem mencionar que fazer plotagens é sempre divertido). O resultado foi um projeto muito curto - mas completo - mostrando como podemos reunir essas três técnicas para resolver um problema de ciência de dados.

O código completo deste projeto está disponível como um Notebook Jupyter no Google Collaboratory (este é um novo serviço que estou experimentando, onde você pode compartilhar e colaborar em Notebooks Jupyter na nuvem. Parece o futuro!) Para editar o bloco de anotações , abra em Colaboratory, selecione file> salve uma cópia no drive e então você pode fazer qualquer alteração e rodar o Notebook.

. . .

Raspagem da web

Enquanto a maioria dos dados usados em classes e livros didáticos só aparece pronto para uso em um formato limpo, na realidade, o mundo não joga tão bem. Obter dados geralmente significa sujar as mãos, nesse caso, puxar dados (também conhecidos como raspagem) da web. O Python tem ótimas ferramentas para fazer isso, ou seja, a biblioteca de `requests` para recuperar o conteúdo de uma página da Web e o `bs4` (BeautifulSoup) para extrair as informações relevantes.

Essas duas bibliotecas costumam ser usadas juntas da seguinte maneira: primeiro, fazemos uma solicitação GET para um site. Em seguida, criamos um objeto Beautiful Soup a partir do conteúdo que é retornado e o analisamos usando vários métodos.

```
  # solicitações para buscar o html do site
  pedidos de importação


  # Faça o pedido GET para um URL
  r = requests.get ('
http://www.cleveland.com/metro/index.ssf/2017/12/case_weste
rn_reserve_university_president_barbara_snyders_base_salary
_and_bonus_pay_tops_among_private_colleges_in_ohio.html' )
```

```
  # Extrair o conteúdo
  c = r.content


  de bs4 import BeautifulSoup
```

```
  # Criar um objeto de sopa
  sopa = BeautifulSoup (c)
```

O objeto de sopa resultante é bastante intimidante:

```
                <!-- Article -->
    <div class="entry-content" id="entryContent">
        <p>CLEVELAND, Ohio - <a href="http://www.case.edu/">Case Weste
<p>Nationally, Snyder's total compensation of $1.154 million, which ca
<p>She is among 58 presidents to earn $1 million or more. The previous
<p>The Chronicle's <a href="https://www.chronicle.com/interactives/exe
<p>Former University of Dayton President Daniel Curran, who left in 20
<p>Following is total compensation for other presidents at private col
<ul>
<li>Grant Cornwell, College of Wooster (left in 2015): $911,651</li>
<li>Marvin Krislov, Oberlin College (left in 2016):  $829,913</li>
<li>Mark Roosevelt, Antioch College, (left in 2015): $507,672</li>
<li>Laurie Joyner, Wittenberg University (left in 2015): $463,504</li>
<li>Richard Giese, University of Mount Union (left in 2015): $453,800<
<li>Sean Decatur,Kenyon College: $451,698</li>
<li>Adam Weinberg, Denison University: $435,322</li>
```

Nossos dados estão em algum lugar, mas precisamos extraí-los. Para selecionar nossa mesa a partir da sopa, precisamos encontrar os seletores CSS corretos. Uma maneira de fazer isso é ir até a página da Web e inspecionar o elemento. Nesse caso, também podemos apenas observar a sopa e ver que nossa tabela está em uma tag HTML `<div>` com o atributo `class = "entry-content"` . Usando essa informação e o método `.find` do nosso objeto de sopa, podemos extrair o conteúdo do artigo principal.

```
  # Encontre o elemento na página da web
  main_content = soup.find ('div', attrs = {'class':
'conteúdo de entrada'})
```

Isso retorna outro objeto de sopa que não é suficientemente específico. Para selecionar a tabela, precisamos encontrar a tag `<ul>` (veja a imagem acima). Nós também queremos lidar apenas com o texto na tabela, então usamos o atributo `.text` da sopa.

```
  # Extrair as informações relevantes como texto
  content = main_content.find ('ul'). texto
```

```
('\n'
 'Grant Cornwell, College of Wooster (left in 2015): $911,651\n'
 'Marvin Krislov, Oberlin College (left in 2016): \xa0$829,913\n'
 'Mark Roosevelt, Antioch College, (left in 2015): $507,672\n'
 'Laurie Joyner, Wittenberg University (left in 2015): $463,504\n'
 'Richard Giese, University of Mount Union (left in 2015): $453,800\n'
 'Sean Decatur,Kenyon College: $451,698\n'
 'Adam Weinberg, Denison University: $435,322\n'
 'Daniel Dibiasio, Ohio Northern University: $414,716\n'
 'Denvy Bowman, Capital University (left in 2016): $388,570\n'
 'Anne Steele, Muskingum University (left in 2016): $384,233\n'
 'Kathy Krendl, Otterbein University: \xa0$378,035\n'
 'Rockwell Jones, Ohio Wesleyan University: $366,625\n'
 'Robert Helmer, Baldwin Wallace University: $365,616\n'
 'Robert Huntington, Heidelberg University: $300,005\n'
 'Lori Varlotta, Hiram College: $293,336\n'
 'Joseph Bruno, Marietta College (left in 2016): $288,295\n'
 'W. Richard Merriman Jr., University of Mount Union (started in June 2015): '
 '$221,761\n')
```

Agora temos o texto exato da tabela como uma string, mas claramente não é muito útil para nós ainda! Para extrair partes específicas de uma string de texto, precisamos seguir para expressões regulares. Eu não tenho espaço neste artigo (nem tenho a experiência!) Para explicar completamente expressões regulares, então aqui eu só dou uma breve visão geral e mostro os resultados. Ainda estou aprendendo e descobri que a única maneira de melhorar é a prática. Sinta-se à vontade para ler este caderno para alguma prática, e confira a documentação do Python para começar (a documentação é geralmente seca, mas *extremamente* útil).

Expressões regulares

A idéia básica de expressões regulares é definir um padrão (a expressão regular ou regex) que queremos combinar em uma string de texto e, em seguida, pesquisar na string para retornar correspondências. Alguns desses padrões parecem muito estranhos porque contêm tanto o conteúdo que queremos combinar quanto caracteres especiais que mudam como o padrão é interpretado. Expressões regulares surgem o tempo todo ao analisar informações de strings e são uma ferramenta vital para aprender pelo menos em um nível básico!

Existem 3 informações que precisamos extrair da tabela de texto:

1. Os nomes dos presidentes
2. Os nomes das faculdades

3. Os salários

Primeiro é o nome. Nesta expressão regular, faço uso do fato de que cada nome está no início de uma linha e termina com uma vírgula. O código abaixo cria um padrão de expressão regular e, em seguida, pesquisa na sequência para localizar todas as ocorrências do padrão:

```
  # Criar um padrão para combinar nomes
  name_pattern = re.compile (r '^ ([AZ] {1}. +?) (?:,)',
flags = re.M)

  # Encontre todas as ocorrências do padrão
  names = name_pattern.findall (conteúdo)
```

```
['Grant Cornwell',
 'Marvin Krislov',
 'Mark Roosevelt',
 'Laurie Joyner',
 'Richard Giese',
 'Sean Decatur',
 'Adam Weinberg',
 'Daniel Dibiasio',
 'Denvy Bowman',
 'Anne Steele',
 'Kathy Krendl',
 'Rockwell Jones',
 'Robert Helmer',
 'Robert Huntington',
 'Lori Varlotta',
 'Joseph Bruno',
 'W. Richard Merriman Jr.']
```

Como eu disse, o padrão é bem complexo, mas faz exatamente o que queremos! Não se preocupe com os detalhes do padrão, mas pense no processo geral: primeiro defina um padrão e, em seguida, pesquise uma string para encontrar o padrão.

Repetimos o procedimento com as faculdades e o salário:

```
  # Faça o patttern da escola e extraia escolas
  school_pattern = re.compile (r '(?:, |, \ s) ([AZ] {1}.
*?) (?: \ s \ (|: |,)')
  schools = school_pattern.findall (conteúdo)
```

```
  # Padrão para combinar com os salários
  salary_pattern = re.compile (r '\ $. +')
  salaries = salary_pattern.findall (conteúdo)
```

```
['College of Wooster',                ['$911,651',
 'Oberlin College',                    '$829,913',
 'Antioch College',                    '$507,672',
 'Wittenberg University',              '$463,504',
 'University of Mount Union',          '$453,800',
 'Kenyon College',                     '$451,698',
 'Denison University',                 '$435,322',
 'Ohio Northern University',           '$414,716',
 'Capital University',                 '$388,570',
 'Muskingum University',               '$384,233',
 'Otterbein University',               '$378,035',
 'Ohio Wesleyan University',           '$366,625',
 'Baldwin Wallace University',         '$365,616',
 'Heidelberg University',              '$300,005',
 'Hiram College',                      '$293,336',
 'Marietta College',                   '$288,295',
 'University of Mount Union']          '$221,761']
```

Infelizmente, o salário está em um formato que nenhum computador entenderia como números. Felizmente, isso nos dá a chance de praticar o uso de uma compreensão de lista do Python para converter os salários das seqüências em números. O código a seguir ilustra como usar o fatiamento, `split` e `join` string, tudo dentro de uma compreensão de lista para alcançar os resultados desejados:

```
  # Salários bagunçados
  salários = ['$ 876,001', '$ 543,903', '$ 2453,896']

  # Converter salários em números em uma compreensão de
lista
  [int (''. join (s [1:]. split (','))) para s em salários]
```

```
  [876001, 543903, 2453896]
```

Aplicamos essa transformação aos nossos salários e finalmente temos todas as informações que queremos. Vamos colocar tudo em um dataframe `pandas` . Neste ponto, insiro manualmente as informações para minha faculdade (CWRU) porque não estava na tabela principal. É importante saber quando é mais eficiente fazer as coisas à mão do que escrever um programa complicado (embora esse artigo inteiro seja contra esse ponto!).

	College	President	salary
0	CWRU	Barbara Synder	1154000.0
1	College of Wooster	Grant Cornwell	911651.0
2	Oberlin College	Marvin Krislov	829913.0
3	Antioch College	Mark Roosevelt	507672.0
4	Wittenberg University	Laurie Joyner	463504.0
5	University of Mount Union	Richard Giese	453800.0
6	Kenyon College	Sean Decatur	451698.0

Subconjunto do Dataframe

Visualização

Este projeto é indicativo de ciência de dados porque a maior parte do tempo foi gasto coletando e formatando os dados. No entanto, agora que temos um conjunto de dados limpo, conseguimos fazer alguns gráficos! Podemos usar `matplotlib` e `seaborn` para visualizar os dados.

Se não estamos muito preocupados com a estética, podemos usar o método de plotagem de quadros de dados embutidos para mostrar rapidamente os resultados:

```
# Faça um gráfico de barras horizontal
df.plot (tipo = 'barh', x = 'Presidente', y = 'salário')
```

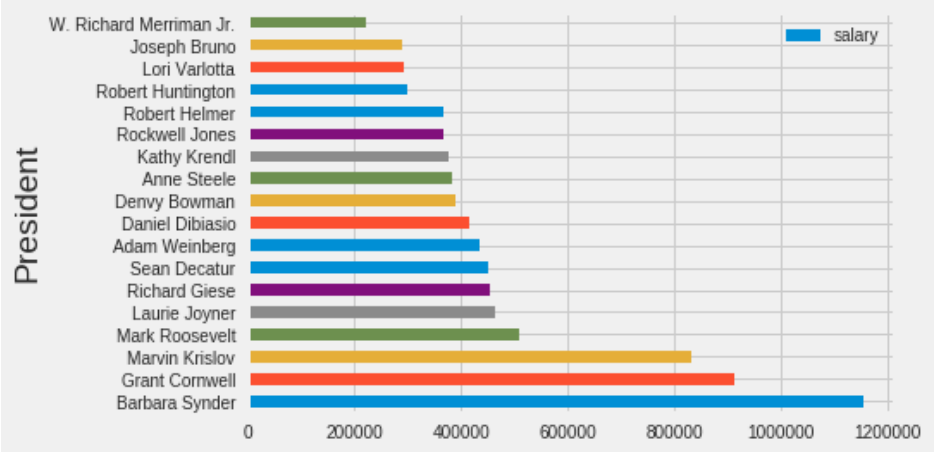
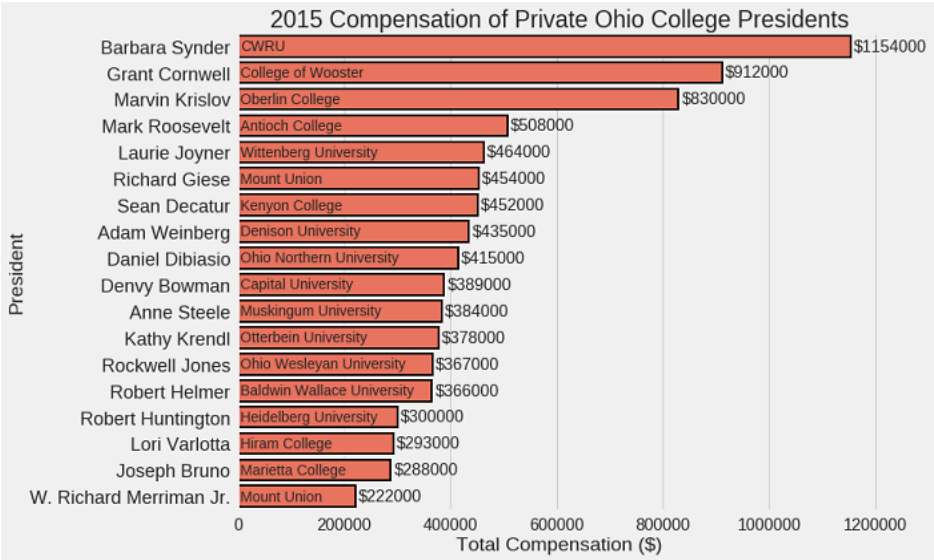


Gráfico padrão usando o método de plotagem de quadros de dados

Para obter uma trama melhor, temos que fazer algum trabalho. Plotagem de código em Python, como expressões regulares, pode ser um pouco complexo, e leva um pouco de prática para se acostumar. Principalmente, aprendo construindo respostas em sites como o Stack Overflow ou lendo a documentação oficial .

Depois de um pouco de trabalho, obtemos a seguinte plotagem (veja o caderno para detalhes):



Melhor trama usando seaborn

Muito melhor, mas isso ainda não responde à minha pergunta original! Para mostrar quanto os estudantes estão pagando por 5 minutos do tempo de seu presidente, podemos converter os salários em $ / cinco minutos, considerando 2000 horas de trabalho por ano.

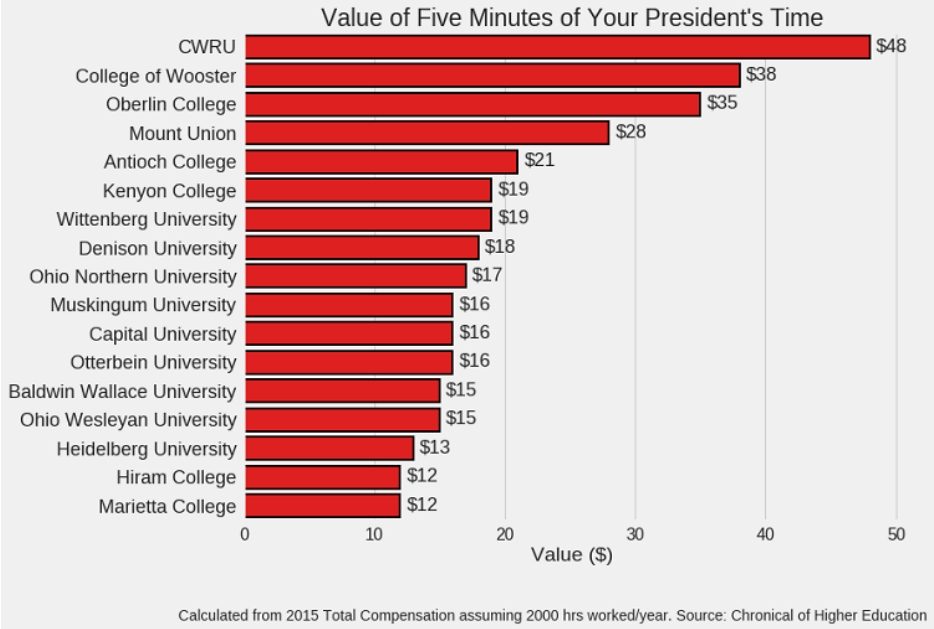


Figura final

Este não é necessariamente um enredo digno de publicação, mas é uma boa maneira de finalizar um pequeno projeto.

Conclusões

A maneira mais eficaz de aprender habilidades técnicas é fazendo. Embora todo esse projeto possa ter sido feito manualmente inserindo valores no Excel, gosto de ter uma visão de longo prazo e pensar em como as habilidades aprendidas aqui ajudarão no futuro. O processo de aprendizagem é mais importante que o resultado final e, neste projeto, pudemos ver como usar 3 habilidades críticas para a ciência de dados:

1. Captura de Web: Recuperando dados online
2. Expressões regulares: Analisando nossos dados para extrair informações
3. Visualização: mostrando todo o nosso trabalho duro

Agora, vá até lá e comece seu próprio projeto e lembre-se: não precisa mudar o mundo para valer a pena.

Eu agradeço feedback e discussão e pode ser alcançado no Twitter @koehrsen_will .

Web Scraping, Regular Expressions, and Data Visualization: Doing it all in Python

A Small Real-World Project for Learning Three Invaluable Data Science Skills

Will Koehrsen Follow
Apr 28, 2018 · 7 min read

As with most interesting projects, this one started with a simple question asked half-seriously: how much tuition do I pay for five minutes of my college president's time? After a chance pleasant discussion with the president of my school (CWRU), I wondered just how much my conversation had cost me.

My search led to this article, which along with my president's salary, had this table showing the salaries of private college presidents in Ohio:

- Grant Cornwell, College of Wooster (left in 2015): $911,651
- Marvin Krislov, Oberlin College (left in 2016): $829,913
- Mark Roosevelt, Antioch College, (left in 2015): $507,672
- Laurie Joyner, Wittenberg University (left in 2015): $463,504
- Richard Giese, University of Mount Union (left in 2015): $453,800
- Sean Decatur,Kenyon College: $451,698
- Adam Weinberg, Denison University: $435,322
- Daniel Dibiasio, Ohio Northern University: $414,716
- Denvy Bowman, Capital University (left in 2016): $388,570
- Anne Steele, Muskingum University (left in 2016): $384,233
- Kathy Krendl, Otterbein University: $378,035
- Rockwell Jones, Ohio Wesleyan University: $366,625
- Robert Helmer, Baldwin Wallace University: $365,616
- Robert Huntington, Heidelberg University: $300,005
- Lori Varlotta, Hiram College: $293,336
- Joseph Bruno, Marietta College (left in 2016): $288,295
- W. Richard Merriman Jr., University of Mount Union (started in June 2015): $221,761

While I could have found the answer for my president, (SPOILER ALERT, it's $48 / five minutes), and been satisfied, I wanted to take the idea further using this table. I had been looking for a chance to practice web scraping and regular expressions in Python and decided this was a great short project.

Although it almost certainly would have been faster to manually enter the data in Excel, then I would not have had the invaluable opportunity to practice a few skills! Data science is about solving problems using a diverse set of tools, and web scraping and regular expressions are two areas I need some work on (not to mention that making plots is always fun). The result was a very short—but complete—project showing how we can bring together these three techniques to solve a data science problem.

The complete code for this project is available as a Jupyter Notebook on Google Colaboratory (this is a new service I'm trying out where you can share and collaborate on Jupyter Notebooks in the cloud. It feels like the future!) To edit the notebook, open it up in Colaboratory, select file > save a copy in drive and then you can make any changes and run the Notebook.

. . .

Web Scraping

While most data used in classes and textbooks just appears ready-to-use in a clean format, in reality, the world does not play so nice. Getting data usually means getting our hands dirty, in this case pulling (also known as scraping) data from the web. Python has great tools for doing this, namely the `requests` library for retrieving content from a webpage, and `bs4` (BeautifulSoup) for extracting the relevant information.

These two libraries are often used together in the following manner: first, we make a GET request to a website. Then, we create a Beautiful Soup object from the content that is returned and parse it using several methods.

```
# requests for fetching html of website
import requests


# Make the GET request to a url
r =
requests.get('http://www.cleveland.com/metro/index.ssf/2017
/12/case_western_reserve_university_president_barbara_snyde
rs_base_salary_and_bonus_pay_tops_among_private_colleges_in
_ohio.html')
```

```
# Extract the content
c = r.content


from bs4 import BeautifulSoup
```

```
# Create a soup object
soup = BeautifulSoup(c)
```

The resulting soup object is quite intimidating:

```
                <!-- Article -->
    <div class="entry-content" id="entryContent">
        <p>CLEVELAND, Ohio - <a href="http://www.case.edu/">Case Weste
<p>Nationally, Snyder's total compensation of $1.154 million, which ca
<p>She is among 58 presidents to earn $1 million or more. The previous
<p>The Chronicle's <a href="https://www.chronicle.com/interactives/exe
<p>Former University of Dayton President Daniel Curran, who left in 20
<p>Following is total compensation for other presidents at private col
<ul>
<li>Grant Cornwell, College of Wooster (left in 2015): $911,651</li>
<li>Marvin Krislov, Oberlin College (left in 2016):  $829,913</li>
<li>Mark Roosevelt, Antioch College, (left in 2015): $507,672</li>
<li>Laurie Joyner, Wittenberg University (left in 2015): $463,504</li>
<li>Richard Giese, University of Mount Union (left in 2015): $453,800<
<li>Sean Decatur,Kenyon College: $451,698</li>
<li>Adam Weinberg, Denison University: $435,322</li>
```

Our data is in there somewhere, but we need to extract it. To select our table from the soup, we need to find the right CSS selectors. One way to do this is by going to the webpage and inspecting the element. In this case, we can also just look at the soup and see that our table resides under a `<div>` HTML tag with the attribute `class = "entry-content"` . Using this info and the `.find` method of our soup object, we can pull out the main article content.

```
# Find the element on the webpage
main_content = soup.find('div', attrs = {'class': 'entry-
content'})
```

This returns another soup object which is not quite specific enough. To select the table, we need to find the `<ul>` tag (see above image). We also want to deal with only the text in the table, so we use the `.text` attribute of the soup.

```
# Extract the relevant information as text
content = main_content.find('ul').text
```

```
('\n'
 'Grant Cornwell, College of Wooster (left in 2015): $911,651\n'
 'Marvin Krislov, Oberlin College (left in 2016): \xa0$829,913\n'
 'Mark Roosevelt, Antioch College, (left in 2015): $507,672\n'
 'Laurie Joyner, Wittenberg University (left in 2015): $463,504\n'
 'Richard Giese, University of Mount Union (left in 2015): $453,800\n'
 'Sean Decatur,Kenyon College: $451,698\n'
 'Adam Weinberg, Denison University: $435,322\n'
 'Daniel Dibiasio, Ohio Northern University: $414,716\n'
 'Denvy Bowman, Capital University (left in 2016): $388,570\n'
 'Anne Steele, Muskingum University (left in 2016): $384,233\n'
 'Kathy Krendl, Otterbein University: \xa0$378,035\n'
 'Rockwell Jones, Ohio Wesleyan University: $366,625\n'
 'Robert Helmer, Baldwin Wallace University: $365,616\n'
 'Robert Huntington, Heidelberg University: $300,005\n'
 'Lori Varlotta, Hiram College: $293,336\n'
 'Joseph Bruno, Marietta College (left in 2016): $288,295\n'
 'W. Richard Merriman Jr., University of Mount Union (started in June 2015): '
 '$221,761\n')
```

We now have the exact text of the table as a string, but clearly is it not of much use to us yet! To extract specific parts of a text string, we need to move on to regular expressions. I don't have space in this article (nor do I have the experience!) to completely explain regular expressions, so here I only give a brief overview and show the results. I'm still learning myself, and I have found the only way to get better is practice. Feel free to go over this notebook for some practice, and check out the Python `re` documentation to get started (documentation is usually dry but *extremely* helpful).

Regular Expressions

The basic idea of regular expressions is we define a pattern (the "regular expression" or "regex") that we want to match in a text string and then search in the string to return matches. Some of these patterns look pretty strange because they contain both the content we want to match and special characters that change how the pattern is interpreted. Regular expressions come up all the time when parsing string information and are a vital tool to learn at least at a basic level!

There are 3 pieces of info we need to extract from the text table:

1. The names of the presidents
2. The names of the colleges
3. The salaries

First up is the name. In this regular expression, I make use of the fact that each name is at the start of a line and ends with a comma. The code below creates a regular expression pattern, and then searches through the string to find all occurrences of the pattern:

```
# Create a pattern to match names
name_pattern = re.compile(r'^([A-Z]{1}.+?)(?:,)', flags =
re.M)


# Find all occurrences of the pattern
names = name_pattern.findall(content)
```

```
['Grant Cornwell',
 'Marvin Krislov',
 'Mark Roosevelt',
 'Laurie Joyner',
 'Richard Giese',
 'Sean Decatur',
 'Adam Weinberg',
 'Daniel Dibiasio',
 'Denvy Bowman',
 'Anne Steele',
 'Kathy Krendl',
 'Rockwell Jones',
 'Robert Helmer',
 'Robert Huntington',
 'Lori Varlotta',
 'Joseph Bruno',
 'W. Richard Merriman Jr.']
```

Like I said, the pattern is pretty complex, but it does exactly what we want! Don't worry about the details of the pattern, but just think about the general process: first define a pattern, and then search a string to find the pattern.

We repeat the procedure with the colleges and the salary:

```
# Make school patttern and extract schools
school_pattern = re.compile(r'(?:,|,\s)([A-Z]{1}.*?)(?:\s\
(|:|,)')
schools = school_pattern.findall(content)
```

```
# Pattern to match the salaries
salary_pattern = re.compile(r'\$.+')
salaries = salary_pattern.findall(content)
```

```
['College of Wooster',                ['$911,651',
 'Oberlin College',                    '$829,913',
 'Antioch College',                    '$507,672',
 'Wittenberg University',              '$463,504',
 'University of Mount Union',          '$453,800',
 'Kenyon College',                     '$451,698',
 'Denison University',                 '$435,322',
 'Ohio Northern University',           '$414,716',
 'Capital University',                 '$388,570',
 'Muskingum University',               '$384,233',
 'Otterbein University',               '$378,035',
 'Ohio Wesleyan University',           '$366,625',
 'Baldwin Wallace University',         '$365,616',
 'Heidelberg University',              '$300,005',
 'Hiram College',                      '$293,336',
 'Marietta College',                   '$288,295',
 'University of Mount Union']          '$221,761']
```

Unfortunately the salary is in a format that no computer would understand as numbers. Fortunately, this gives us a chance to practice using a Python list comprehension to convert the string salaries into numbers. The following code illustrates how to use string slicing, `split` , and `join` , all within a list comprehension to achieve the results we want:

```
# Messy salaries
salaries = ['$876,001', '$543,903', '$2453,896']
```

```
# Convert salaries to numbers in a list comprehension
[int(''.join(s[1:].split(','))) for s in salaries]
```

```
[876001, 543903, 2453896]
```

We apply this transformation to our salaries and finally have the all info we want. Let's put everything into a `pandas` dataframe. At this point, I

manually insert the information for my college (CWRU) because it was not in the main table. It's important to know when it's more efficient to do things by hand rather than writing a complicated program (although this whole article kind of goes against this point!).

	College	President	salary
0	CWRU	Barbara Synder	1154000.0
1	College of Wooster	Grant Cornwell	911651.0
2	Oberlin College	Marvin Krislov	829913.0
3	Antioch College	Mark Roosevelt	507672.0
4	Wittenberg University	Laurie Joyner	463504.0
5	University of Mount Union	Richard Giese	453800.0
6	Kenyon College	Sean Decatur	451698.0

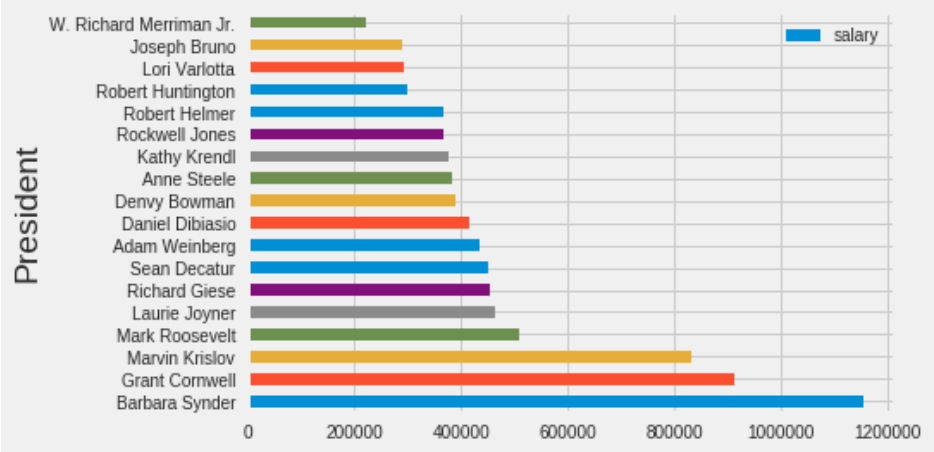
Subset of Dataframe

Visualization

This project is indicative of data science because the majority of time was spent collecting and formatting the data. However, now that we have a clean dataset, we get to make some plots! We can use both `matplotlib` and `seaborn` to visualize the data.

If we aren't too concerned about aesthetics, we can use the built in dataframe plot method to quickly show results:

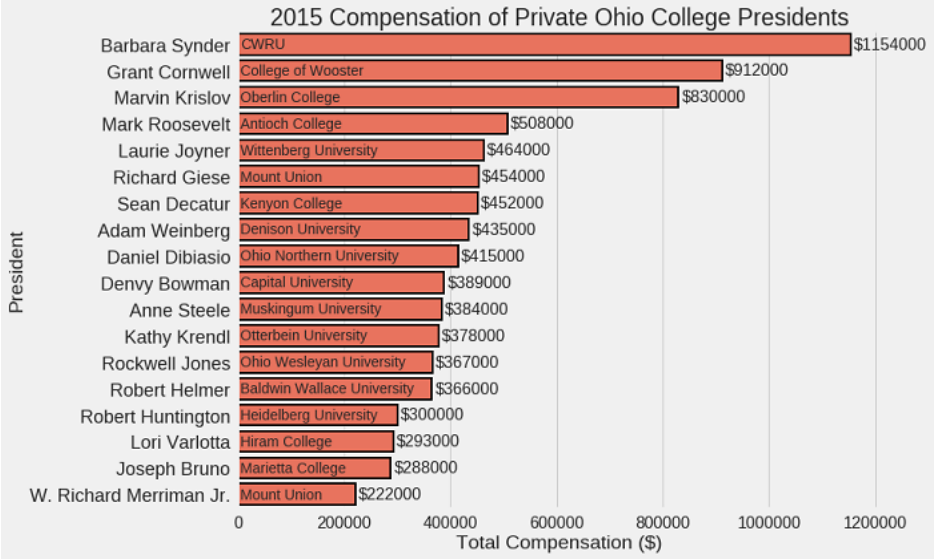
```
# Make a horizontal bar chart
df.plot(kind='barh', x = 'President', y = 'salary')
```



Default plot using dataframe plotting method

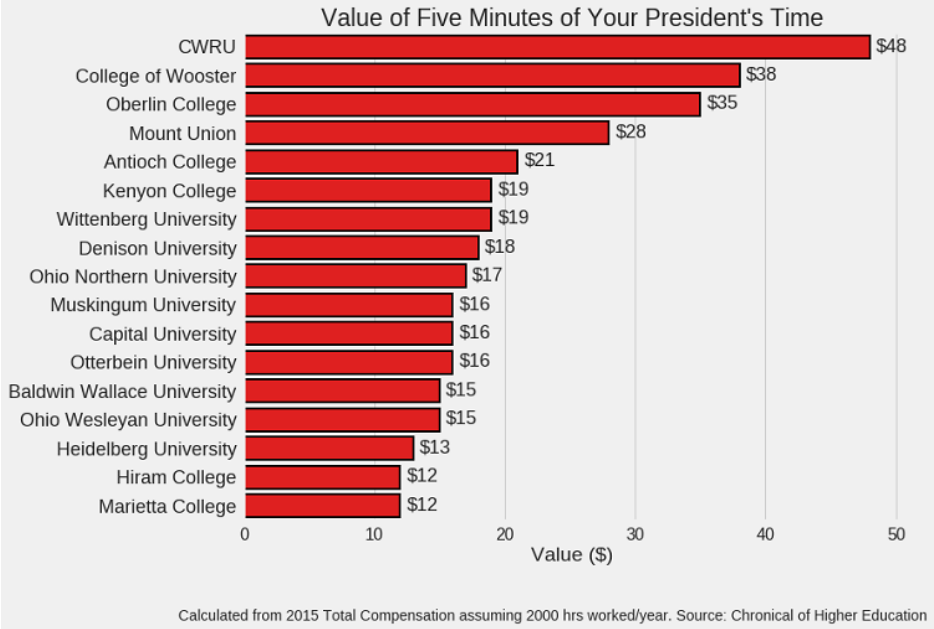
To get a better plot we have to do some work. Plotting code in Python, like regular expressions, can be a little complex, and it takes some practice to get used to. Mostly, I learn by building on answers on sites like Stack Overflow or by reading official documentation.

After a bit of work, we get the following plot (see notebook for the details):



Better Plot using seaborn

Much better, but this still doesn't answer my original question! To show how much students are paying for 5 minutes of their president's time we can convert salaries into $ / five minutes assuming 2000 work hours per year.



Final Figure

This is not necessarily a publication-worthy plot, but it's a nice way to wrap up a small project.

Conclusions

The most effective way to learn technical skills is by doing. While this whole project could have been done manually inserting values into Excel, I like to take the long view and think about how the skills learned here will help in the future. The process of learning is more important than the final result, and in this project we were able to see how to use 3 critical skills for data science:

1. Web Scraping: Retrieving online data
2. Regular Expressions: Parsing our data to extract information
3. Visualization: Showcasing all our hard work

Now, get out there and start your own project and remember: it doesn't have to be world-changing to be worthwhile.

I welcome feedback and discussion and can be reached on Twitter @koehrsen_will.