SEOUL NATIONAL UNIVERSITY
Graduate School of Data Science

M3239.003100: Data Analysis and Visualization
Lecture 2

Data Science Pipeline and Data Collection

Hyunwoo Park
Graduate School of Data Science
Seoul National University

Agenda

- Data Science Pipeline
- Data Collection
- Data Parsing
- Simple Exercise

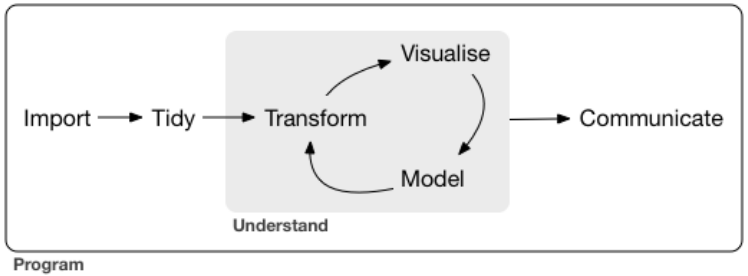
- Class Survey (Due 9/13 before class)
  - Submitted via Google Forms
- Things to do
  - Install Stata
  - Install the Anaconda distribution of Python 3
  - Launch Jupyter Notebook
  - Get to know the terminal interface
  - Pick a text/code editor

Data Science Pipeline

Data science pipeline

- A typical process that a data science project goes through

Collection	Obtaining Data
Cleaning	Cleaning Data
Integration	Visualizing Data
Analysis	Modeling Data
Visualization	Interpreting Data
Presentation	Revisiting Model
Dissemination	



What a typical data science project looks like
https://r4ds.had.co.nz/introduction.html

"What's Data Science Pipeline?"
https://www.geeksforgeeks.org/whats-data-science-pipeline/

Polo Chau's
Analytics Building Blocks
https://poloclub.github.io/cse6242-2018fall-campus/slides/CSE6242-010-AnalyticsBuildingBlocks.pdf

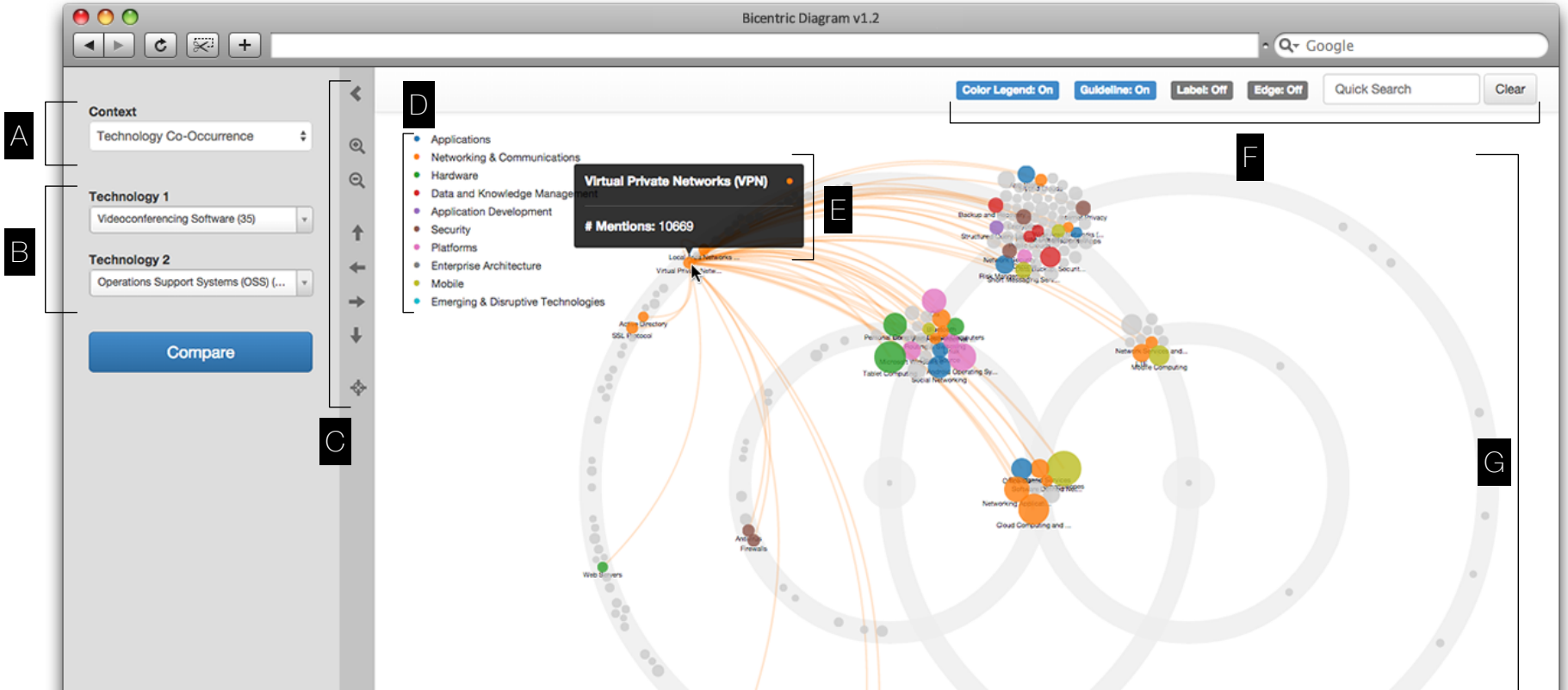
Common theme

- You may skip some steps.
- You can go back and forth. (Blocks are linked by two-way street.)

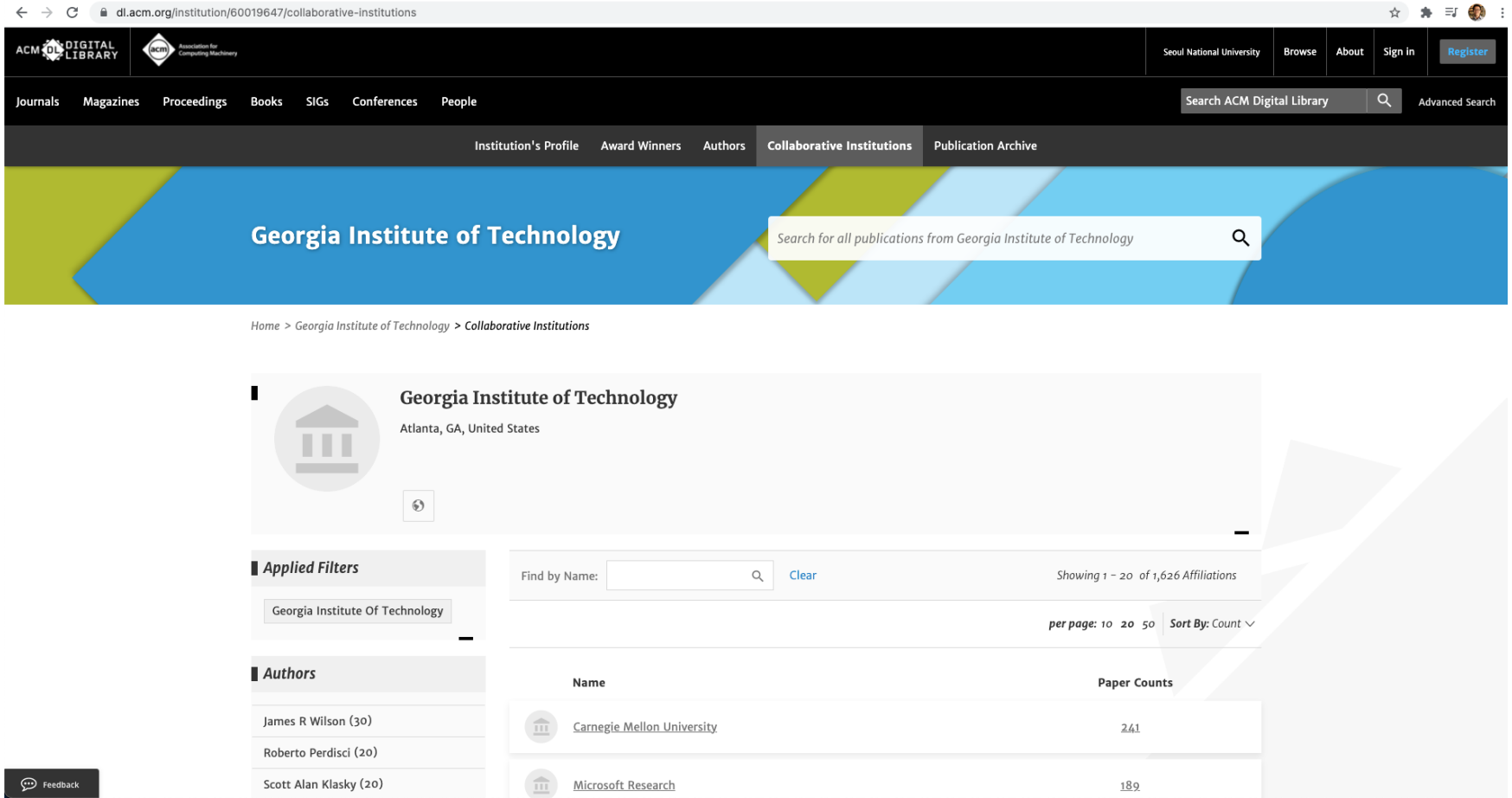
- For example,
  - Data types inform visualization design.
  - Data size informs the choice of algorithms.
  - Analysis motivates more data collection/cleaning/integration.
  - Visualization motivates more data cleaning.
  - Visualization challenges algorithm assumptions. (e.g., users may find results do not make sense.)

An example: Bicentric Diagrams

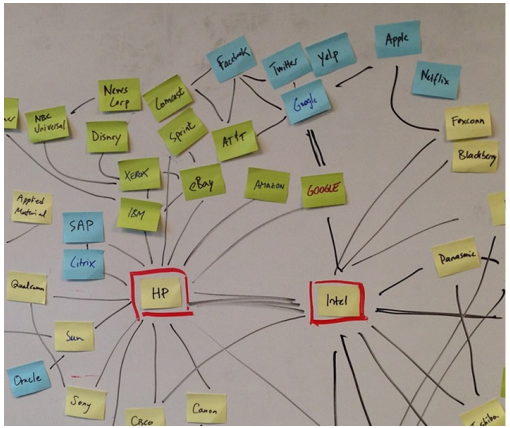
- https://youtu.be/TFBf3Nm5XQ8

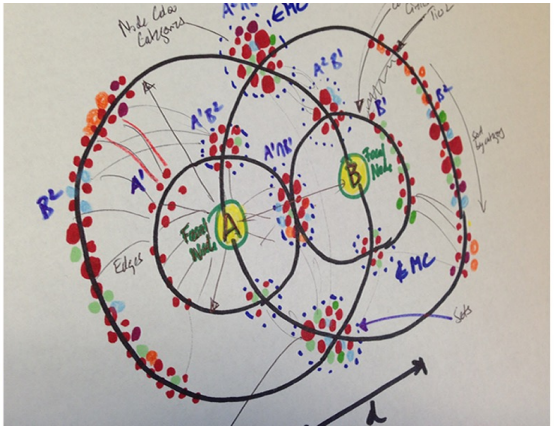


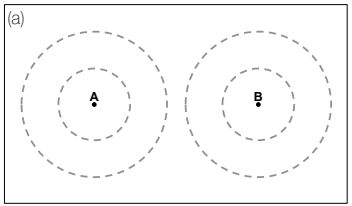
Data Collection / Cleaning / Integrating

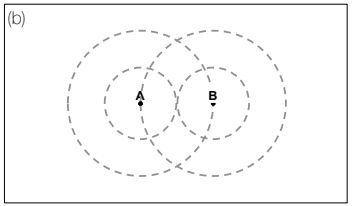


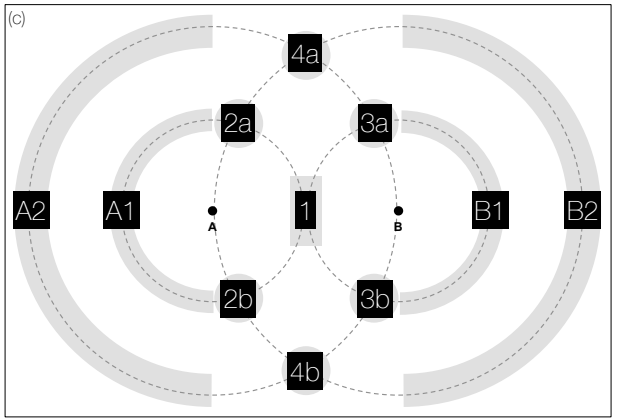
Analysis / Visualization











Presentation / Dissemination

- http://bicentric.herokuapp.com/

Decision Support Systems 84 (2016) 64–77

Contents lists available at ScienceDirect

Decision Support Systems

journal homepage: www.elsevier.com/locate/dss

Bicentric diagrams: Design and applications of a graph-based relational set visualization technique

Hyunwoo Park [a,*,1], Rahul C. Basole [b,1]

[a] Tennenbaum Institute, Georgia Institute of Technology, 85 Fifth Street NW, Atlanta, GA 30308, USA
[b] School of Interactive Computing & Tennenbaum Institute, Georgia Institute of Technology, 85 Fifth Street NW, Atlanta, GA 30308, USA

ARTICLE INFO

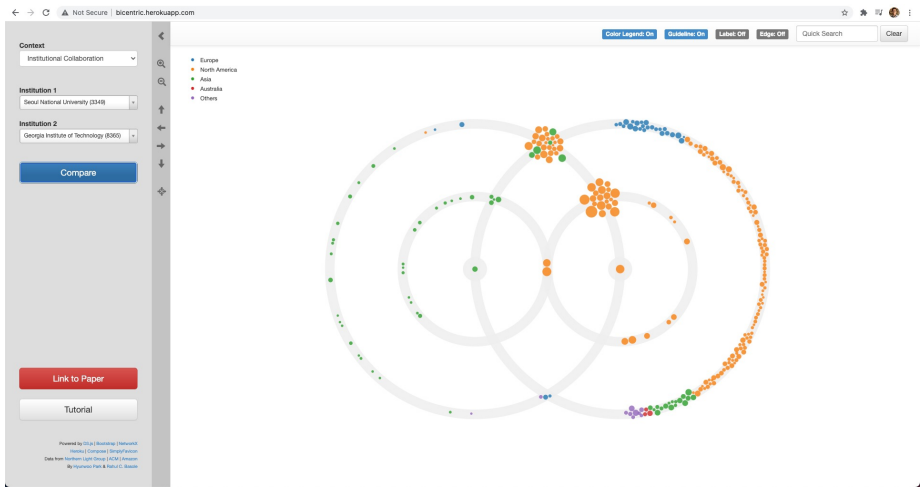
Article history:
Received 20 February 2015
Received in revised form 22 December 2015
Accepted 2 February 2016
Available online 12 February 2016

Keywords:
Bicentric diagram
Set visualization
Network visualization
Strength of ties

ABSTRACT

In an era where data on social, economic, and physical networks are proliferating at a rapid pace, the ability to understand the underlying complex structural connections, discover prominent entities, and identify clusters is becoming increasingly important. It is also well-established that interactive visualizations can amplify human cognition and augment decision making. Motivated by a practical need articulated by corporate decision makers and limitations of existing visual representations, this research presents our journey in designing and implementing bicentric diagrams, a novel graph-based set visualization technique. A bicentric diagram enables simultaneous identification of sets, set relationships, and set member reach in integrated egonetworks of two focal entities. Our technique builds on the well-established sociological theory of tie strength to visually group and position nodes. We illustrate the broad applicability of bicentric diagrams with examples from four diverse sample domains: university collaboration, technology co-occurrence, health app purchases, and interfirm alliance networks. We assess the value of our technique using an expert-based evaluation approach. The paper concludes with implications and a discussion of opportunities for implementation in real-world decision support settings.

© 2016 Elsevier B.V. All rights reserved.

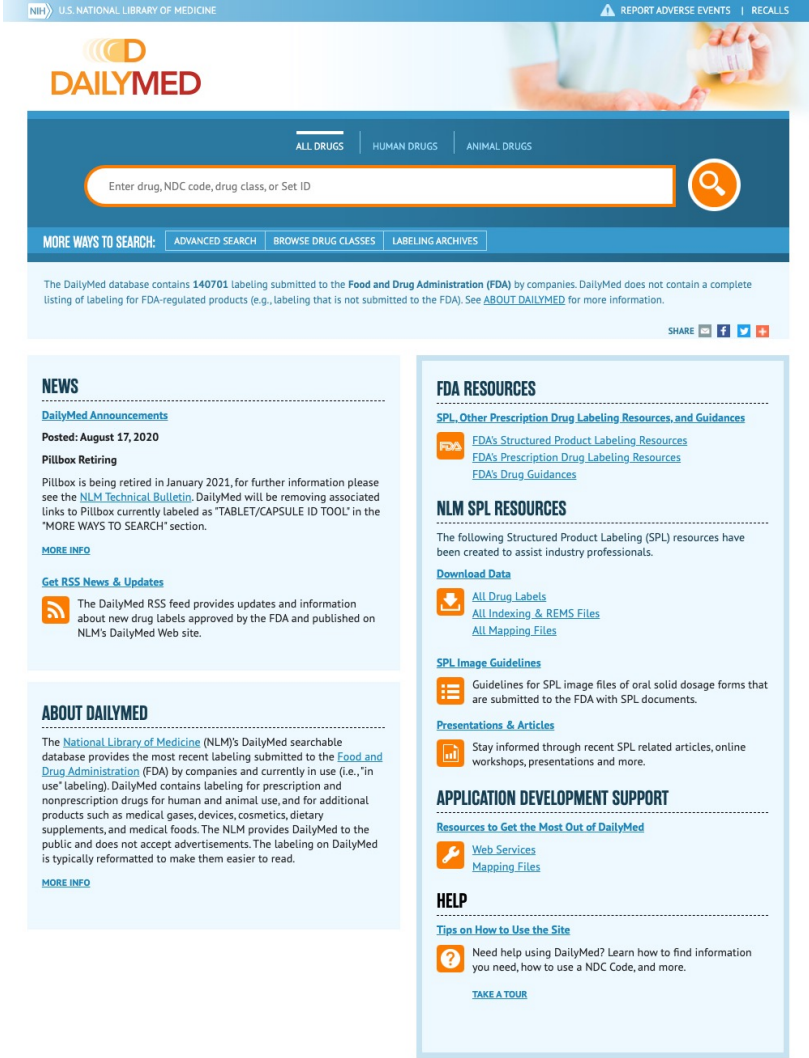


Data Collection

Three ways you can collect data

1. Bulk data download – Easy
2. Application Programming Interface (API) – Medium
3. Scraping or crawling – Hard

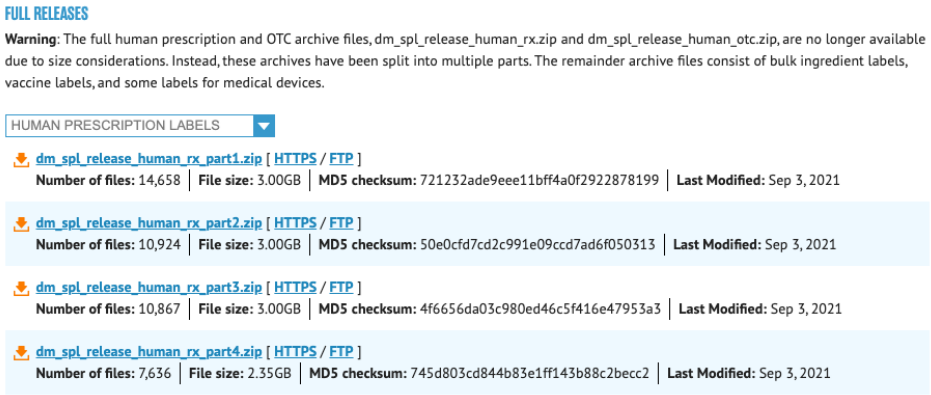
- Combining multiple sources of data is one way to develop a unique dataset.
- Let me use this website as an example. https://dailymed.nlm.nih.gov/



1) Bulk data download

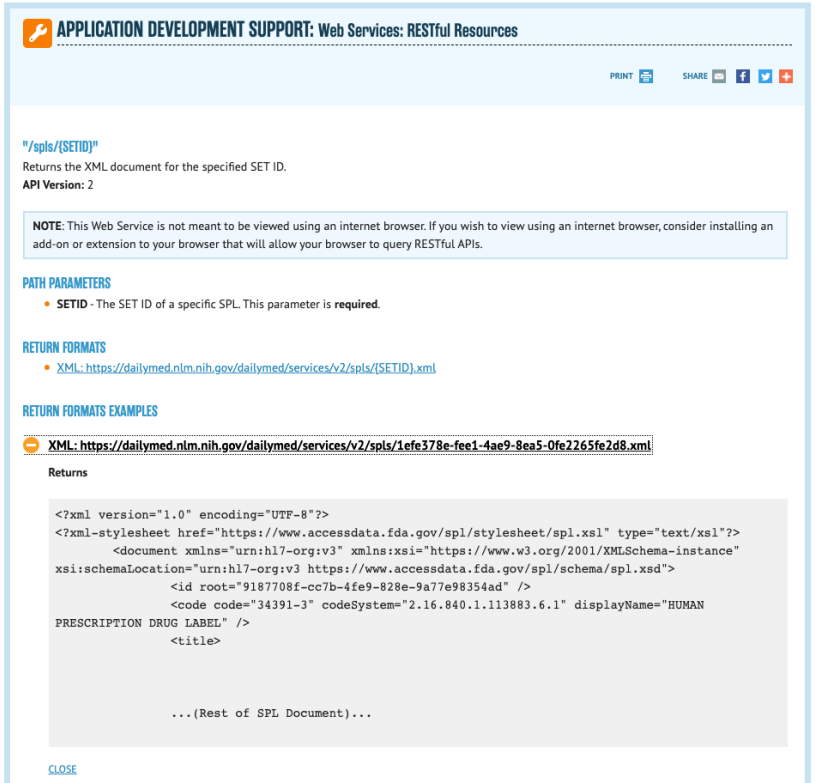
- Data is money these days. Proprietary data is often not available for download.
- Bulk data download is often offered by non-profit organizations or governments.
- Because anyone can obtain such data, your analysis or visual analytics system for such data may not be seen novel and interesting by itself.
- This type of data may provide some supplementary information for your main dataset.
- Examples include:
  - StackOverflow dump, Wikipedia, Data.gov, etc.
  - https://poloclub.github.io/cse6242-2019spring-campus/#datasets
  - http://va.gatech.edu/courses/cs4460/resources/ ["Data Sources" section]
  - Please feel free to post a data source in Korea on the eTL Q&A board.

- https://dailymed.nlm.nih.gov/dailymed/spl-resources-all-drug-labels.cfm



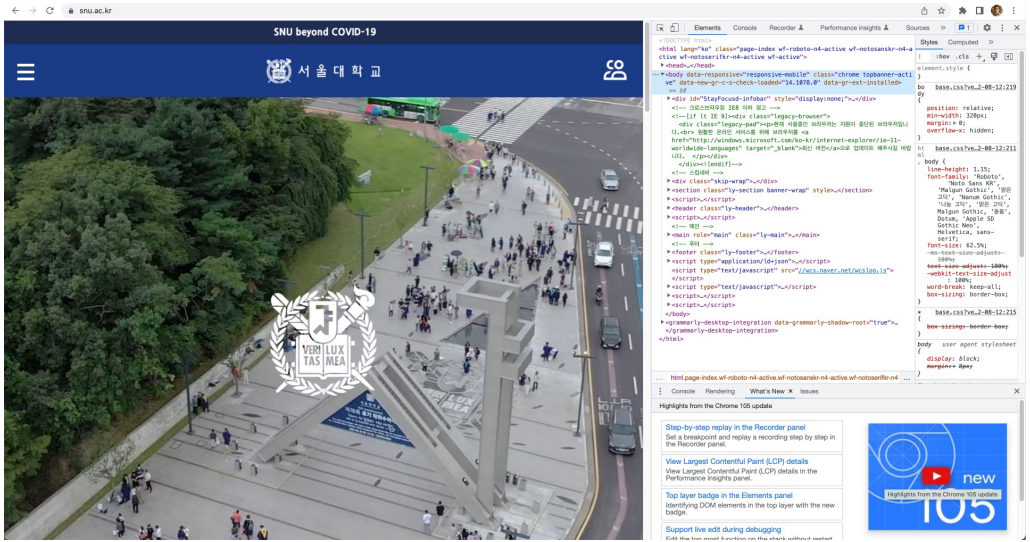
2) API (Application Programming Interface)

- Some web sites expose API endpoints to be accessed.
- API endpoints typically return a structured response in the XML or JSON format.
- For developers, it is easier because data parsing and cleaning work is reduced.
- Instead, the web site retains a greater control and monitoring over what data to be shared and who accesses the data.
- API is often intended for providing real-time data for third-party applications.
- You may need to get an API key to authenticate your request.
- Rate limiting or IP monitoring is typically in place. (a certain number of requests per day or per hour)



3) Scraping or crawling

- The most brute-force and messy way to get data is to scrape a web site.
- You will get a bunch of HTML files and parse/clean them out to get structured data.
- Figuring out the structure of URLs and getting the list of IDs to scrape are the first step in scraping.



Popular Python packages for scraping

- Requests
  - https://docs.python-requests.org/en/latest/user/quickstart/
- Scrapy
  - https://docs.scrapy.org/en/latest/
- Selenium
  - Not part of Anaconda distribution
  - https://selenium-python.readthedocs.io/getting-started.html#simple-usage

- Note
  - You may have to log in with username and password to get the data you want depending on the website.
  - You may need to change the “user agent” of your requests to trick the server.
  - Be courteous to the server load. Space out your requests using something like “time.sleep(1).”
  - Otherwise, your IP may be blocked from the server.

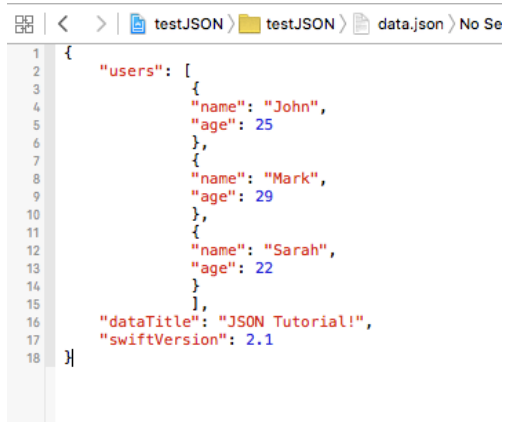
Data Parsing

XML (or HTML) and JSON

- XML (or HTML)

```
<?xml version="1.0" encoding="UTF-8"?><?xml-stylesheet href="
https://www.accessdata.fda.gov/spl/stylesheet/spl.xsl" type="text/xsl"?>
<document xmlns="urn:hl7-org:v3" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="urn:hl7-org:v3
https://www.accessdata.fda.gov/spl/schema/spl.xsd">
   <id root="874057c0-a437-4a08-9105-38c1b2f3307b"/>
   <code code="34391-3" displayName="HUMAN PRESCRIPTION DRUG LABEL"
   codeSystem="2.16.840.1.113883.6.1"/>
   <title>
      <content styleCode="bold">These highlights do not include all the
      information needed to use HUMIRA safely and effectively. See full
      prescribing information for HUMIRA.</content>
      <br/>
      <content styleCode="bold">
      <0xa0><0xa0><0xa0><0xa0><0xa0><0xa0><0xa0></content>
      <br/>
      <content styleCode="bold">HUMIRA</content>
      <content styleCode="bold">
         <sup>®</sup>
      </content>
      <content styleCode="bold"> (adalimumab) injection, for
      subcutaneous use</content>
      <br/>
      <content styleCode="bold">Initial U.S. Approval: 2002</content>
      <br/>
   </title>
   <effectiveTime value="20210224"/>
   <setId root="608d4f0d-b19f-46d3-749a-7159aa5f933d"/>
   <versionNumber value="2135"/>
   <author>
      <time/>
      <assignedEntity>
         <representedOrganization>
            <id extension="078458370" root="1.3.6.1.4.1.519.1"/>
            <name>AbbVie Inc.</name>
            <assignedEntity>
               <assignedOrganization/>
            </assignedEntity>
         </representedOrganization>
      </assignedEntity>
```

- JSON



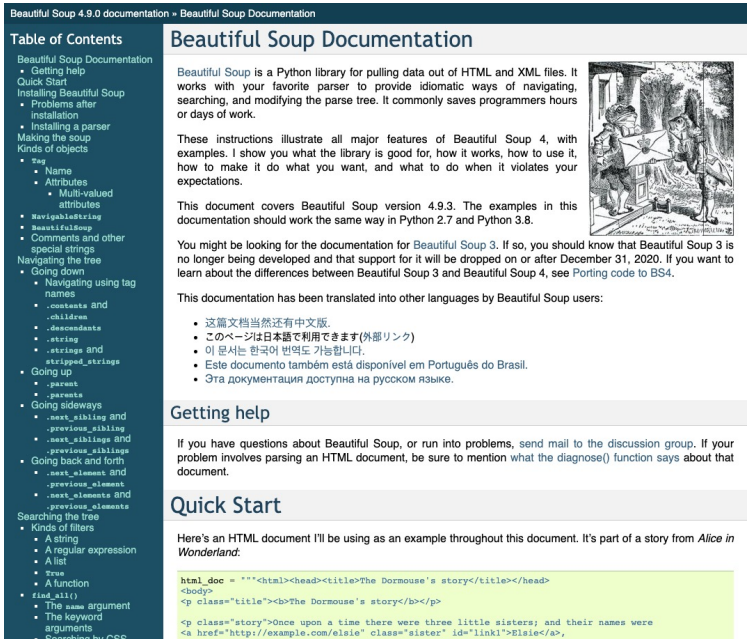
testJSON › testJSON › data.json › No Se

```
{
    "users": [
                {
                "name": "John",
                "age": 25
                },
                {
                "name": "Mark",
                "age": 29
                },
                {
                "name": "Sarah",
                "age": 22
                }
                ],
    "dataTitle": "JSON Tutorial!",
    "swiftVersion": 2.1
}
```

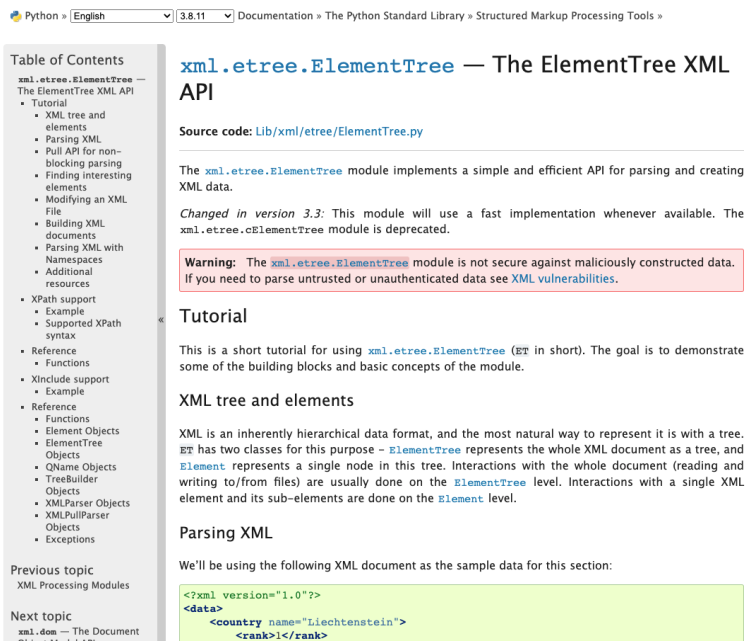
- Basics
  - https://www.geeksforgeeks.org/xml-basics/
  - https://www.geeksforgeeks.org/javascript-json/
  - https://www.geeksforgeeks.org/difference-between-json-and-xml/

Parsing XML (and HTML)

- BeautifulSoup (package name: bs4)
  - https://www.crummy.com/software/BeautifulSoup/bs4/doc/



- ElementTree (built-in)
  - https://docs.python.org/3.8/library/xml.etree.elementtree.html



Parsing JSON

- json (built-in)

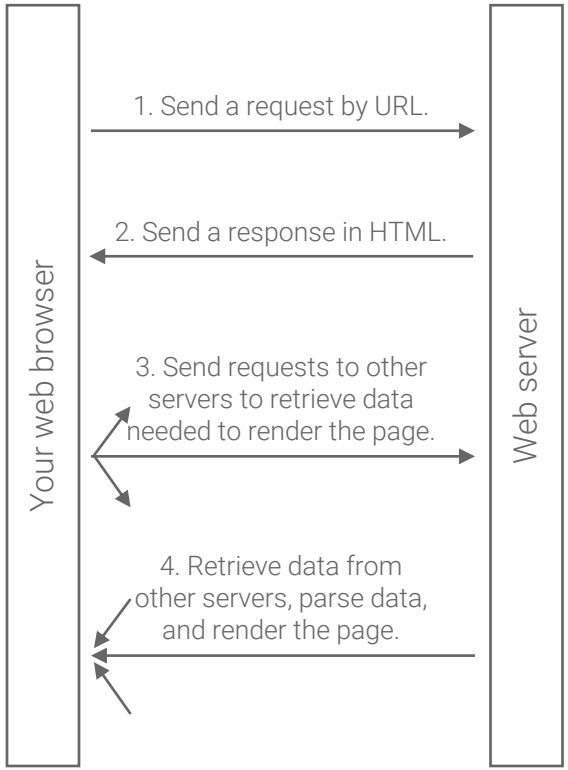
```
1  import json
2  a = json.loads(open('simple.json').read())
3  a['mylist']
4  a['mydict']
5  with open('simple2.json', 'w') as f: f.write(json.dumps(a))
```

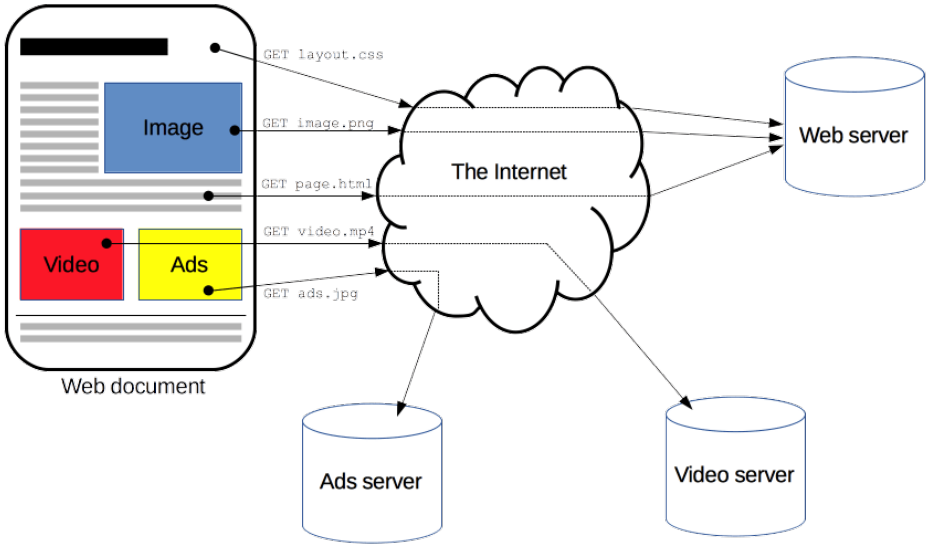
simple.json

```
{
  "mylist": [1, 2, "a", "b"],
  "mydict": {"3": "c", "4": "d"}
}
```

Simple Exercise

Web request flow





https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

Three different situations when scraping

- Scenario 1
  - https://hyunwoopark.com/courses/dav-fall21/scrape1.html

- Scenario 2
  - https://hyunwoopark.com/courses/dav-fall21/scrape2.html

- Scenario 3
  - https://hyunwoopark.com/courses/dav-fall21/scrape3.html

Let's try.

- First example: scrape1.html

```
import requests
r = requests.get('https://hyunwoopark.com/courses/dav-fall21/scrape1.html')
print(r.content.decode('ascii'))
```

```
<!doctype html>
<html>
<body>
<h1>This is a test page for scraping exercise.</h1>
</body>
</html>
```

```
from bs4 import BeautifulSoup as soup
s = soup(r.content)
s.h1.text
```

```
'This is a test page for scraping exercise.'
```

IFRAME and asynchronous DOM update

- Second example

```
u = 'https://hyunwoopark.com/courses' \
+'/dav-fall21/scrape2.html'
r = requests.get(u)
print(r.content.decode('ascii'))
```

```
<!doctype html>
<html>
<head>
  <style type="text/css">
    body { margin: 0px; }
    iframe {
      width: 100%;
      border: 0px;
    }
  </style>
</head>
<body>
<iframe src="scrape1.html"></iframe>
</body>
</html>
```

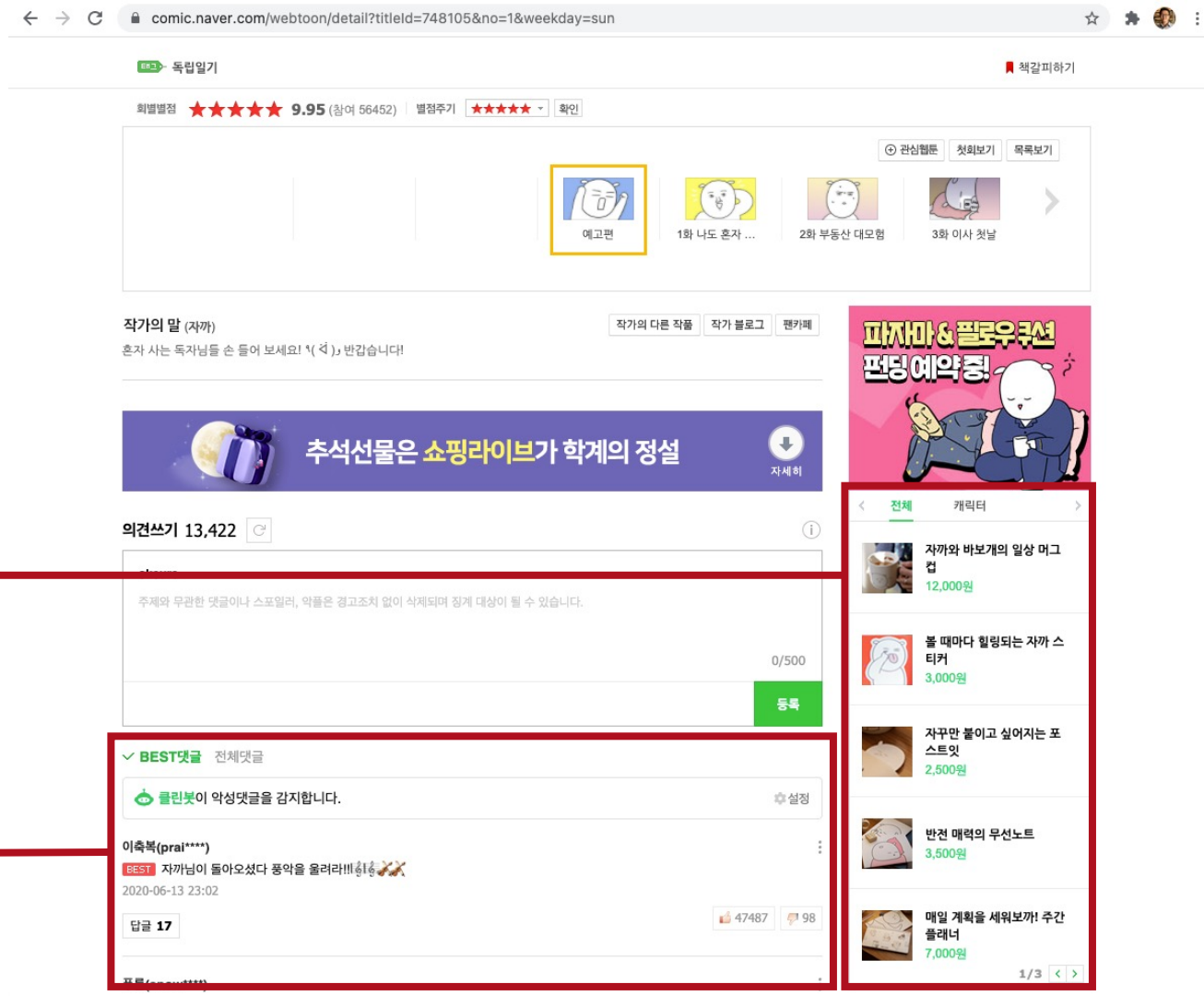
- Third example

```
u = 'https://hyunwoopark.com/courses/dav-fall21/scrape3.html'
r = requests.get(u)
print(r.content.decode('ascii'))
```

```
<!doctype html>
<html>
<head>
  <script type="text/javascript">
    document.addEventListener("DOMContentLoaded", function(event) {
      var h1 = document.createElement("h1");
      h1.innerHTML = "This is a test page for scraping exercise.";
      document.getElementsByTagName('body')[0].appendChild(h1);
    });
  </script>
</head>
<body>
</body>
</html>
```

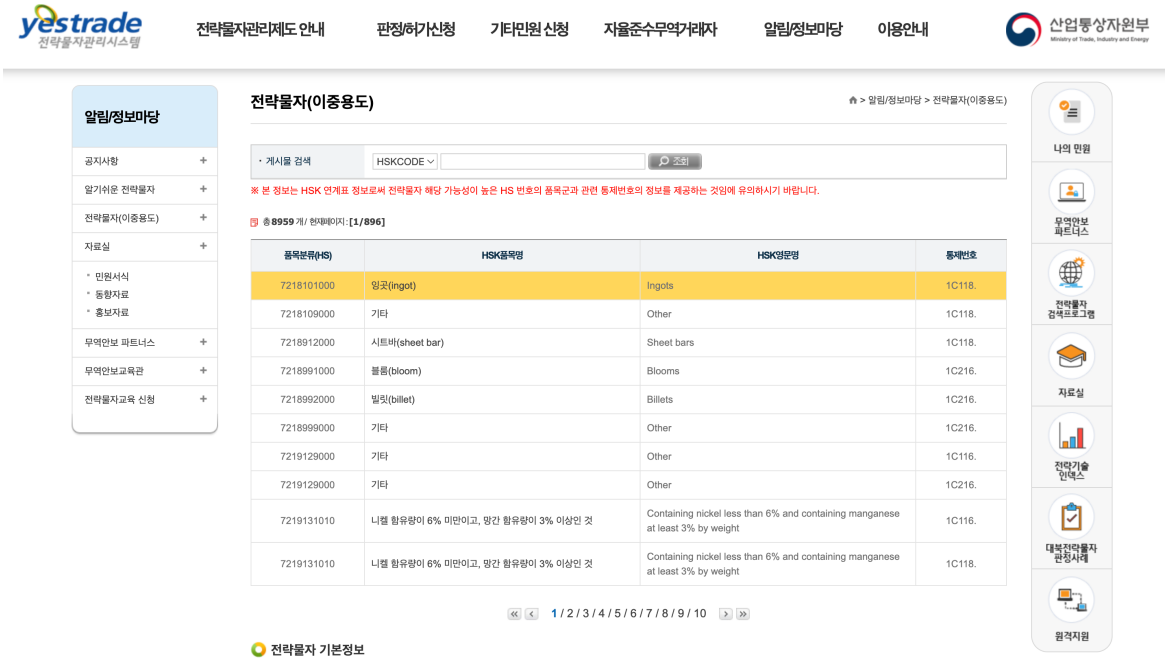
Real-world case

- Related shop items are fetched as JSON asynchronously.
- Comments are in an IFRAME.



Another exercise

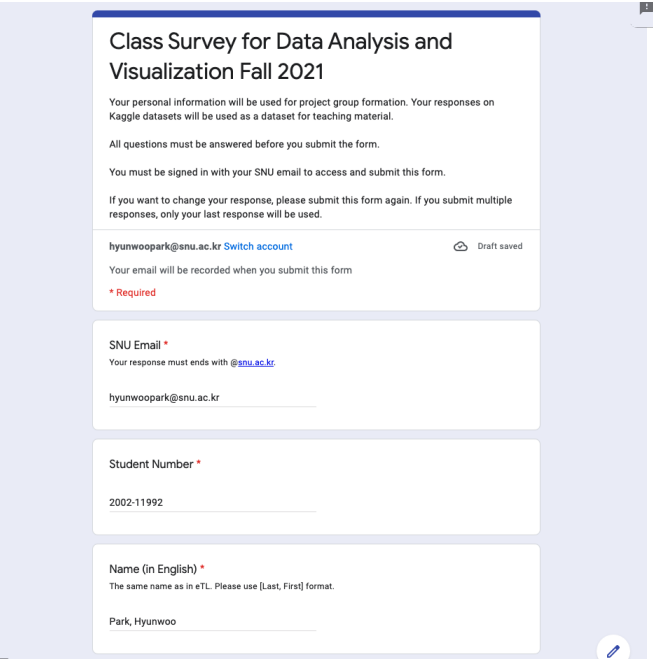
- https://www.yestrade.go.kr/common/common.do?jPath=/im/imCt010L&CURRENT_MENU_CODE=MENU1120

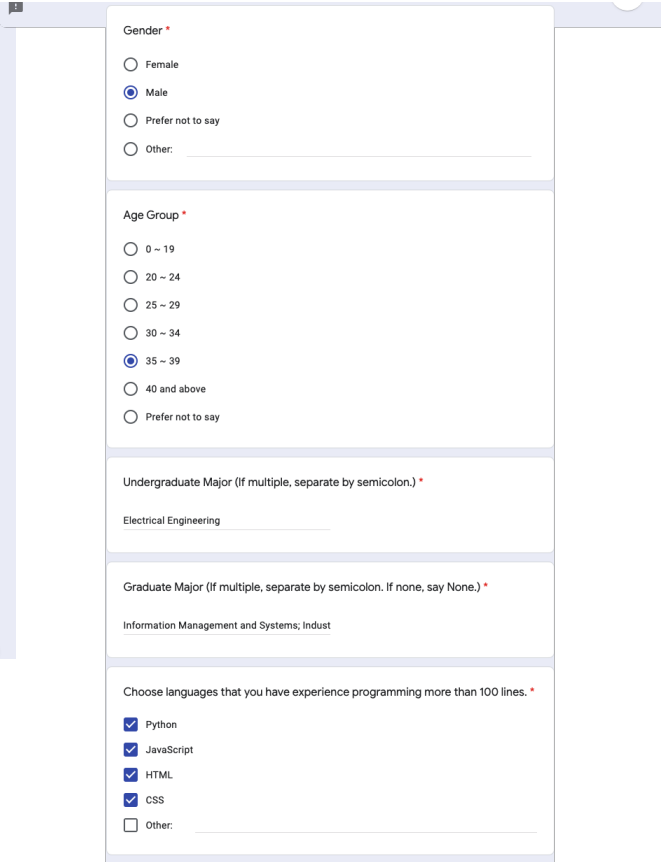


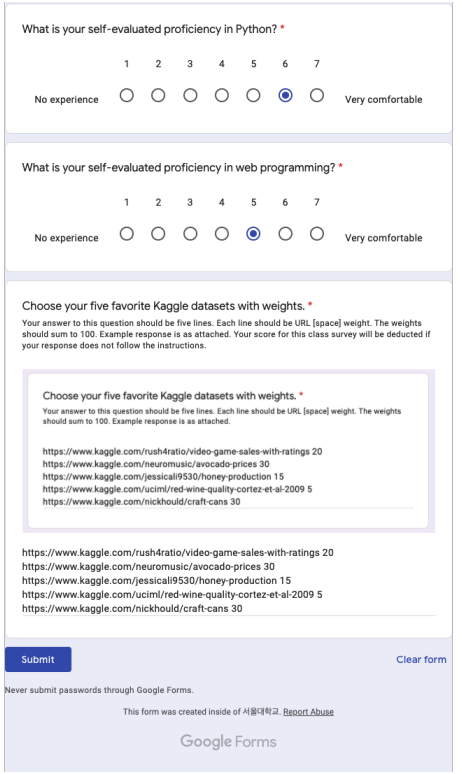
Class Survey
Due 9/13
before class 2:00pm KST

Example Response

https://forms.gle/SBsgZpFcAciLYRjP8

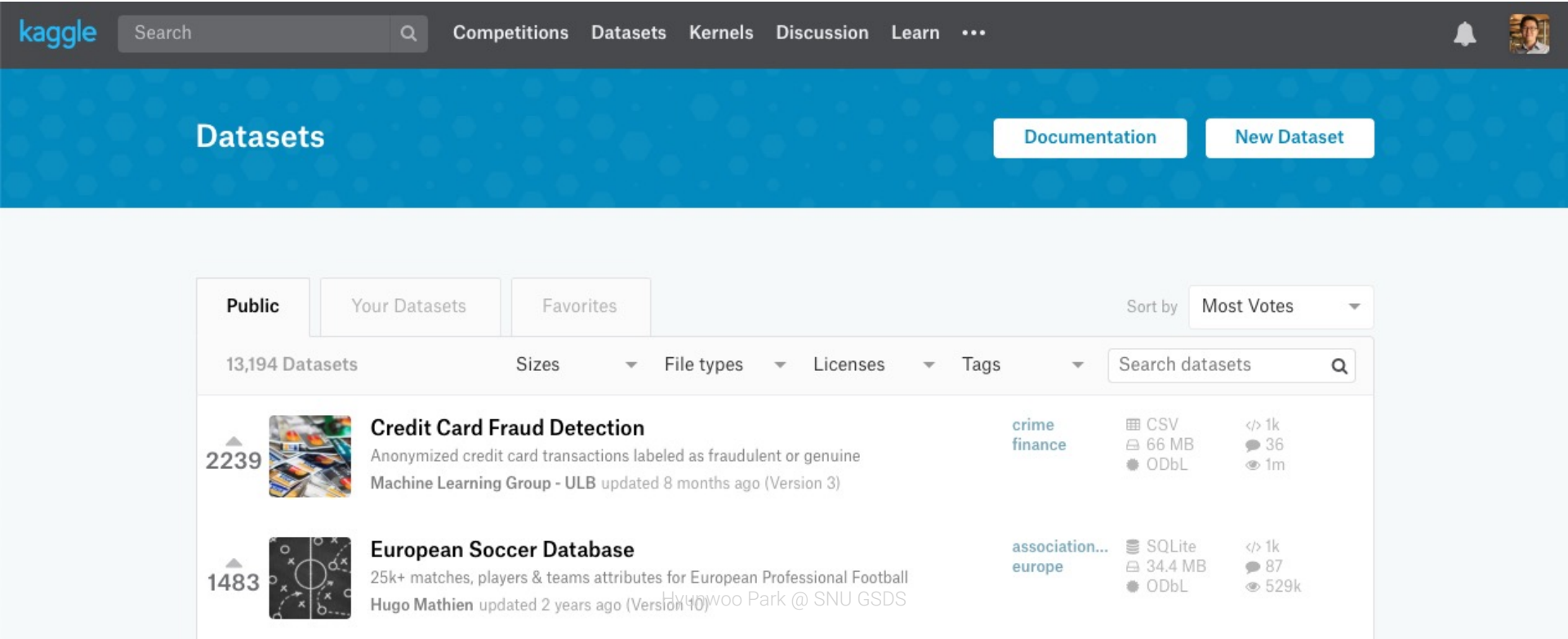






Datasets for this course

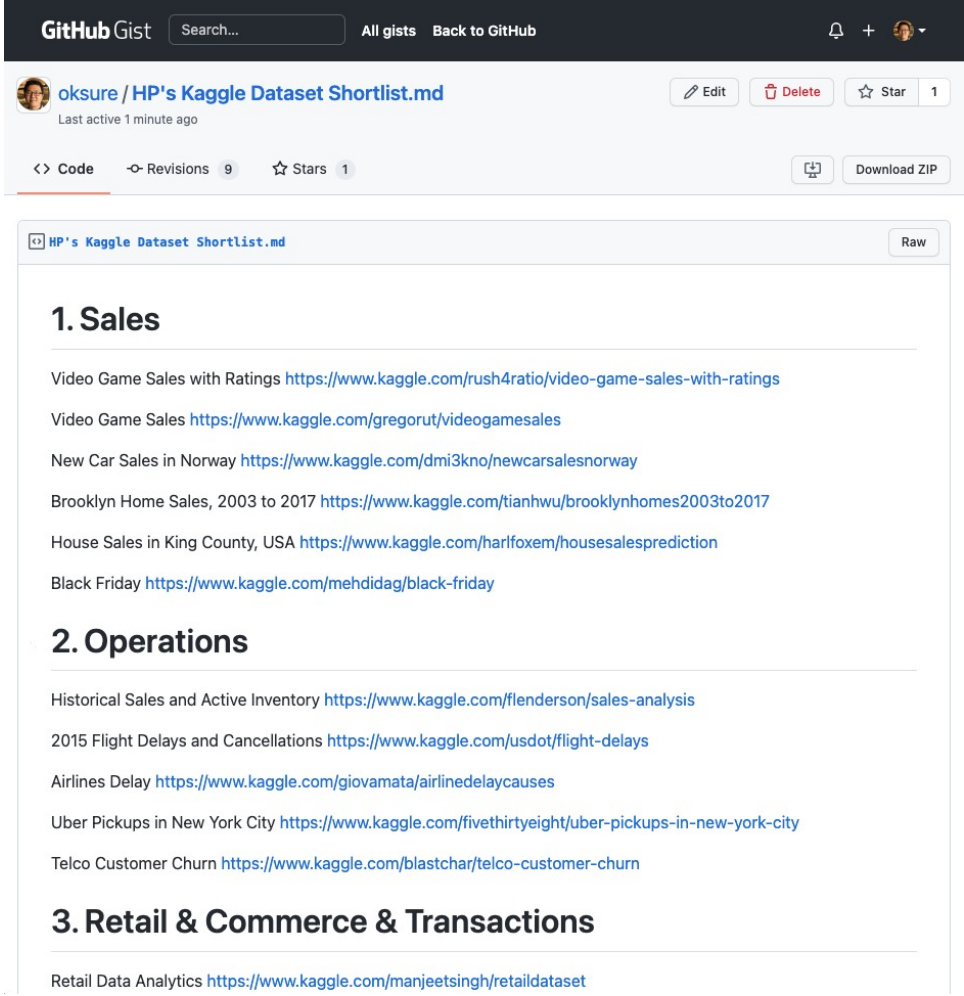
- Kaggle (https://www.kaggle.com/)



Hyunwoo Park @ SNU GSDS

My shortlist datasets

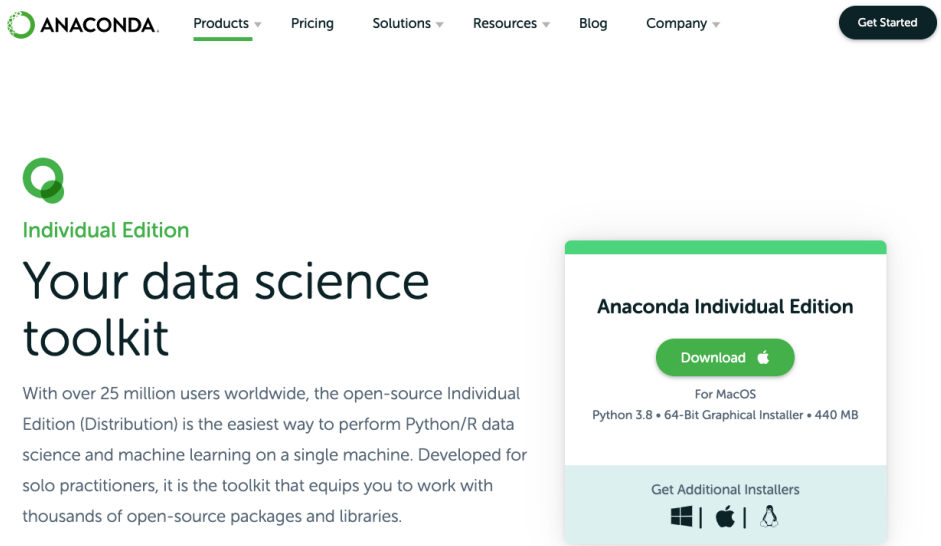
- https://bit.ly/hp-kaggle-datasets
- I browsed public Kaggle datasets and handpicked a set of datasets relevant to business and management.
- This list will be updated after this semester reflecting your input.



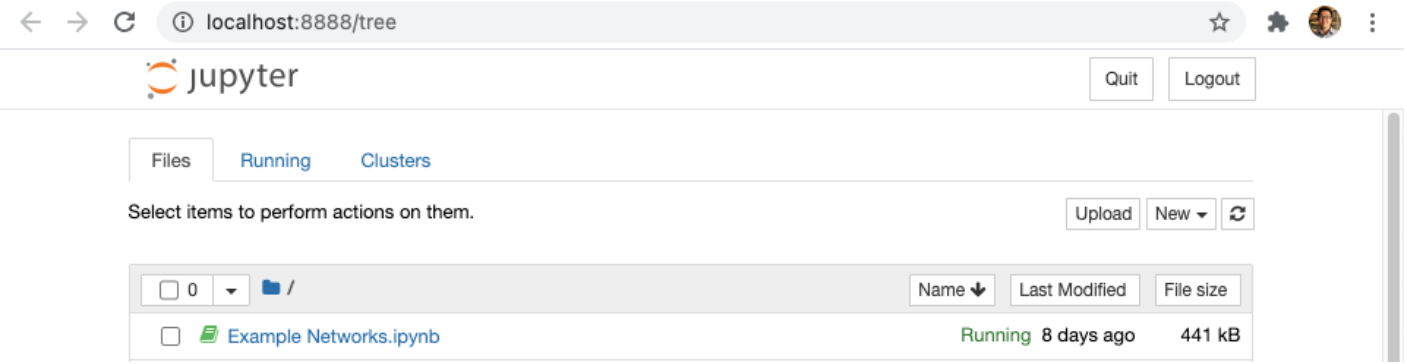
Things To Do

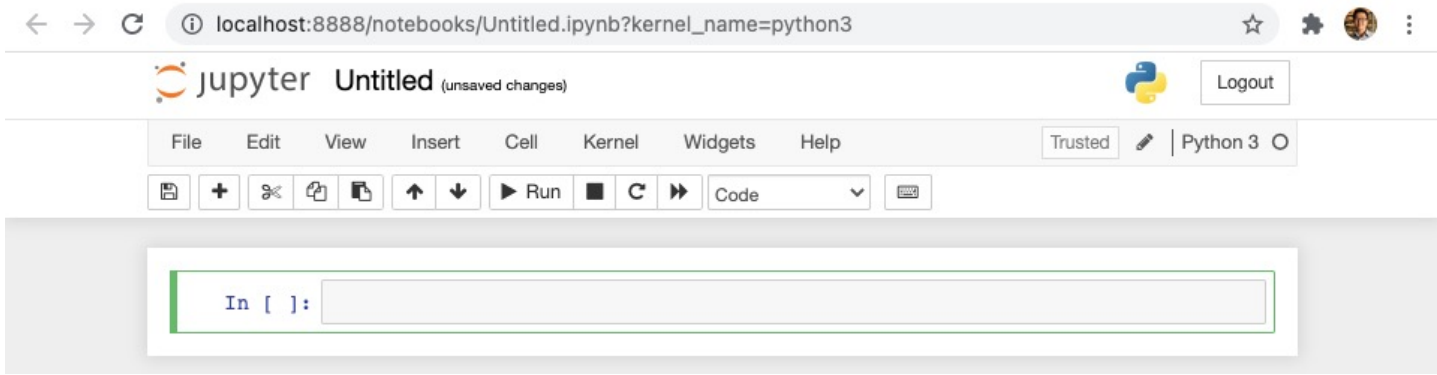
Install Python 3 Anaconda distribution

- https://www.anaconda.com/products/individual
- Python 3.8 (Version 2021.05)
- Included packages:
  - https://docs.anaconda.com/anaconda/packages/py3.8_osx-64/



Launch Jupyter Notebook



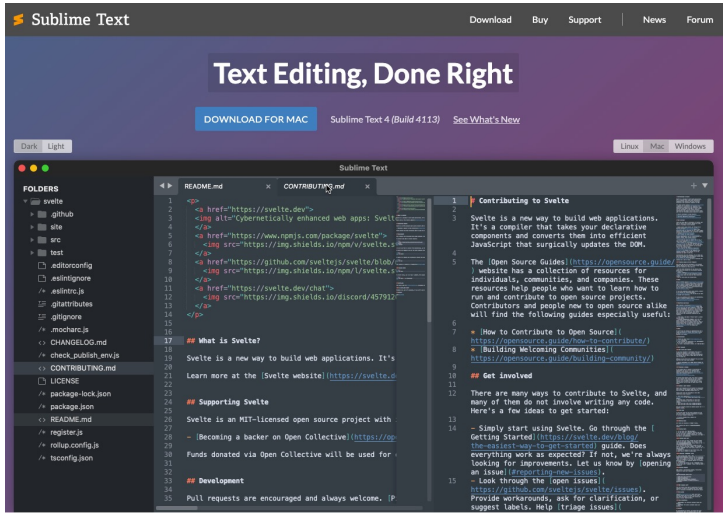


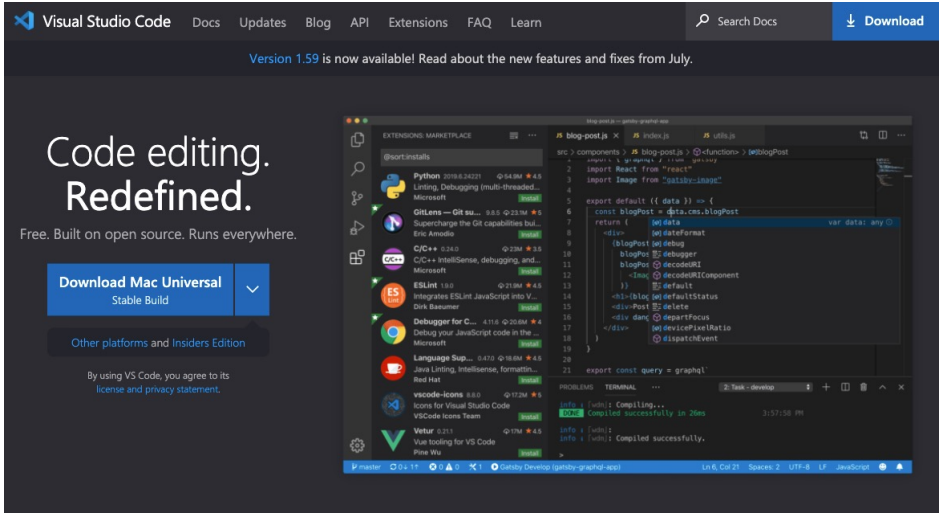
Get to know the terminal interface

```
(base) MacBook-Pro:python_code oksure$ which python
/Users/oksure/opt/anaconda3/bin/python
(base) MacBook-Pro:python_code oksure$ which jupyter
/Users/oksure/opt/anaconda3/bin/jupyter
(base) MacBook-Pro:python_code oksure$ jupyter console
Jupyter console 6.4.0

Python 3.8.8 (default, Apr 13 2021, 12:59:45)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.22.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]:
Do you really want to exit ([y]/n)?
Shutting down kernel
(base) MacBook-Pro:python_code oksure$ [IPKernelApp] WARNING | Parent appears to have exited, shutting down.

(base) MacBook-Pro:python_code oksure$ jupyter notebook
[I 2021-09-02 12:11:37.438 LabApp] JupyterLab extension loaded from /Users/oksure/opt/anaconda3/lib/python3.8/site-packages/jupyterlab
[I 2021-09-02 12:11:37.438 LabApp] JupyterLab application directory is /Users/oksure/opt/anaconda3/share/jupyter/lab
[I 12:11:37.442 NotebookApp] Serving notebooks from local directory: /Users/oksure/Dropbox/Research_SupplyNetwork/python_code
[I 12:11:37.442 NotebookApp] Jupyter Notebook 6.3.0 is running at:
[I 12:11:37.442 NotebookApp] http://localhost:8888/?token=88cbcfda9fdb3afc5d675d6f61b1a7c551e5493321c13a2d
[I 12:11:37.442 NotebookApp]  or http://127.0.0.1:8888/?token=88cbcfda9fdb3afc5d675d6f61b1a7c551e5493321c13a2d
[I 12:11:37.442 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 12:11:37.461 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///Users/oksure/Library/Jupyter/runtime/nbserver-61601-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=88cbcfda9fdb3afc5d675d6f61b1a7c551e5493321c13a2d
     or http://127.0.0.1:8888/?token=88cbcfda9fdb3afc5d675d6f61b1a7c551e5493321c13a2d
[0] 1:bash  2:bash- 3:bash  4:python3.8*                                    "MacBook-Pro.local" 12:11 02-Sep-21
```

Pick your code/text editor

- Sublime Text
  - https://www.sublimetext.com/
- VS Code
  - https://code.visualstudio.com/





Things to do / Reminders

- Class Survey (5%)
  - Due on 9/13 (Tue) before class 2:00pm KST.
- Technical stuff
  - Install Stata
  - Install the Anaconda distribution of Python 3
  - Launch Jupyter Notebook
  - Get to know the terminal interface
  - Pick a text/code editor
- Review
  - Check out the following Python packages.
  - Must: requests, bs4, json
  - Optional: selenium, scrapy, xml
- eTL boards
  - Q&A Board for questions and answers
  - Discussion Board for resources or ideas to share
  - Group Project Board for forming a group and recruiting people
- See you Thursday, 9/8.
  - Data Cleaning / Storing / Merging