QSS20: Modern Statistical Computing

Unit 14: Recap & work session

Goals for today

- Recap of web-scraping
- Recap of SQL
- Upcoming deadlines
- Final project work session

Goals for today

- **Recap of web-scraping**
- Recap of SQL
- Upcoming deadlines
- Final project work session

Recap of web-scraping

Tips:

- Grabbing information from web pages is *web-scraping*. Systematically finding what pages to scrape is *web-crawling*, which usually uses *link extraction* to find & follow links in certain domains
- Most people do *narrow crawling*—grabbing specific information from a few pages with similar structure. To do *broad crawling* over a range of websites, use `scrapy` for flexible, non-blocking (i.e., fast) operations
- To maximize *extensibility* or resilience to new sites/HTML layouts, use an HTML exclusion list to scrape visible text (rather than relying on specific HTML/CSS; BS code, spider code)

Useful general commands:

```
scrapy startproject schools # create scrapy project
scrapy shell 'http://quotes.toscrape.com' # shell to test scrapy
scrapy genspider -t crawl broad site.com # create broad spider
    from CrawlSpider template
scrapy crawl broad -o output.json # run broad spider, save JSON
```

Code for broad spider in broad.py

```python
import scrapy
from scrapy.linkextractors import LinkExtractor
from scrapy.spiders import CrawlSpider, Rule
from schools.items import SchoolsItem

class BroadSpider(CrawlSpider):
    name = 'broad'
    allowed_domains = ['quotes.toscrape.com']
    start_urls = ['http://quotes.toscrape.com/']

    rules = (Rule(
        LinkExtractor(), callback='parse_item', follow=True))

    def parse_item(self, response):
      for quote in response.css('div.quote'):
        item = SchoolsItem() # initialize

        item['text'] = quote.css('span.text::text').get(),
        item['author'] = quote.css('small.author::text').get(),
        item['tags'] = quote.css('div.tags a.tag::text').getall(),

        yield item
```

Where we are

- Recap of web-scraping
- **Recap of SQL**
- Upcoming deadlines
- Final project work session

Recap of SQL

What do you remember?

Recap of SQL: explanation

When to use:

- You walk into an internship/job and they say, "Here's how to authenticate to our database, it's in SQL. You can use that, right?"(very common)
- You have a huge dataset on which to run complex queries → can put in SQL and work from your laptop (requires little memory)
- You want to access a secured, external dataset without downloading → can put in SQL and work from your laptop (like an API)

Tips:

- SQL like condensed version of Pandas; similar but diff. query syntax
- Usually easy to install Python MySQL connector (database setup less easy)
- Use creds `.yaml` file to access connector—or copy into code (less secure)
- Subqueries in `()` are like functions as in `df.apply(func)`
- Use `count(*)` with `group by` to aggregate each slice (e.g., by race x crime)

Recap of SQL: simple examples

Simple ex. w/ basic syntax:

```sql
select CASE_ID, AGE_AT_INCIDENT   --select cols
from caseinit   --tablename
--table ops like joins or subqueries would go here
where AGE_AT_INCIDENT > 40   --condition(s)
```

Ex. creating new col w/ case/when:

```sql
select *,   --take all cols
CASE
    WHEN OFFENSE_CATEGORY = UPDATED_OFFENSE_CATEGORY
    THEN 'Same offense'
    ELSE 'Diff offense'
END as charge_update
from caseinit
```

Recap of SQL: subquery to create col, filter w/ inner join

```
select *,
from caseinit
inner join
    (select CASE_ID as cid, CASE_PARTICIPANT_ID as cpid,
    CASE
        WHEN OFFENSE_CATEGORY = UPDATED_OFFENSE_CATEGORY
        THEN 'Same offense' ELSE 'Diff offense'
    END as charge_update
    from caseinit) as tmp
    on tmp.cid = caseinit.case_ID and
    tmp.cpid = caseinit.CASE_PARTICIPANT_ID
where charge_update = "Diff offense"
```

Equivalent in pandas:

```
tmp = caseinit.copy()
tmp['charge_update'] = np.where(
    tmp.OFFENSE_CATEGORY == tmp.UPDATED_OFFENSE_CATEGORY,
    'Same offense', 'Diff offense')
merged = pd.merge(left=caseinit, right=tmp, how='inner',
    left_on=('CASE_ID', 'CASE_PARTICIPANT_ID'),
    right_on=('cid','cpid'))
merged = merged[merged.charge_update == 'Diff offense']
```

Recap of SQL: group by offense, add top 5 w/ inner join

```sql
select *
from caseinit
inner join(
     select UPDATED_OFFENSE_CATEGORY as tmp_oc,
     count(*) as count_offense
     from caseinit
     where RACE in ("Black", "White")
     group by UPDATED_OFFENSE_CATEGORY
     order by count_offense desc
     limit 5) as top5
     on caseinit.UPDATED_OFFENSE_CATEGORY = top5.tmp_oc
```

Equivalent in pandas:

```python
top5 = caseinit[caseinit.RACE.isin(['Black', 'White'])].
     groupby('UPDATED_OFFENSE_CATEGORY').
     agg({'CASE_ID': 'nunique'}).reset_index().
     rename(columns = {'CASE_ID': 'count_offense',
          'UPDATED_OFFENSE_CATEGORY': 'tmp_oc'}).
     sort_values(by='count_offense', ascending=False)[:5]
merged = pd.merge(left=caseinit, right=top5, how='inner',
     left_on='UPDATED_OFFENSE_CATEGORY',
     right_on='tmp_oc')
```

Where we are

- Recap of web-scraping
- Recap of SQL
- **Upcoming deadlines**
- Final project work session

Upcoming deadlines

- Problem set five
    - **Due next Friday, 11-18**
    - Submit in usual way (assign Prof. & TA a GitHub issue)
    - Reminder: Your lowest pset grade will be dropped
- Final project presentation
    - **Delivered in class this coming Monday, 11-14**
    - Please also submit slides before that class: PDF → Canvas, share w/ Prof. on Overleaf
- Final paper
    - **Due Tuesday, 11-22** (last day of finals)
    - How to submit: PDF → Canvas, share w/ Prof. on Overleaf

Goals for today

- Recap of web-scraping
- Recap of SQL
- Upcoming deadlines
- **Final project work session**

Final project work session

Things to work on:

- Slides for final presentation
- Analysis, codebase development, documentation
- Final paper

Places to work:

- This classroom
- Outside in hall
- Rockefeller Center atrium (downstairs)—**please mind your volume**

Feel free to consult/ask questions:

- Prof. Haber
- Other groups/research teams
- The whole class (consult Prof. first)