QSS20: Modern Statistical Computing

Session 03: Catch Up

Goals for today's session

- **Housekeeping**
- Review notebook focused on pandas questions
- Review notebook covering plotting (relevant for question 2 graphs)
- Loops versus functions (relevant for problem 2.3.3 bandwidth question)

Housekeeping

- **Problem set deadline extended until Sunday 11:59 PM** (canvas announcement yesterday)
- Heterogeneity in backgrounds
- Please fill out Slack profile with picture otherwise we will bother you via DM!
- ggplot2 datacamp assignment - 100% optional and nongraded (optional modules will have March 15th deadline); just if you want plotting practice
- MLK day and next week

Goals for today's session

- Housekeeping
- **Review notebook focused on pandas questions**
- **Review notebook covering plotting (relevant for question 2 graphs)**
- Loops versus functions (relevant for problem 2.3.3 bandwidth question)

Notebook focused on Pandas questions

Open up the following notebook...

00_classquestions.ipynb

Notebook focused on plotting code

Open up the following notebook...

01_example_plotting.ipynb

Goals for today's session

- Housekeeping
- Review notebook focused on pandas questions
- Review notebook covering plotting (relevant for question 2 graphs)
- **Loops versus functions (relevant for problem 2.3.3 bandwidth question)**

Common task

- Do something repeatedly to something else (e.g., do something to every row in a dataset; transform every column)
- Within pandas, the built in methods like `df.mean()`, `df.col.str.contains()` etc only get us so far
- Often may want to iterate over rows, check for something or do something, and store the result
    - **Example:** pset question 2.3.3 on (1) focusing on a policy change in december 2016; (2) iterate over different ways to define the "before changeänd äfter change"windows, (3) return a dataframe with each before/after, and (4) rowbind the results

To make more concrete: we have a couple example crime reports

CCN	WARD	OFFENSE	REPORT_DAT
20165648	6	MOTOR VEHICLE THEFT	2020/11/19 21:25:50+00
20123250	2	MOTOR VEHICLE THEFT	2020/08/29 01:00:25+00

For each of these two crimes, we want to see if there are any reported crimes with (1) same ward and (2) reported within 1000 minutes of the first crime. We want to pull all those crimes and rank them by time proximity.

Blank-ish notebook with loop solution to task but no function solution

Open up the following notebook...

02_loopsfunctions.ipynb

For both approaches, define crimes to look for and crimes to look within

```
## two examples
CCN_examples = ['20165648', '20123250']

## crimes to look 4 matches for
crimes_lookfor = dc_crim_2020.loc[dc_crim_2020.CCN.astype(str).isin
                 (CCN_examples),
                 ['CCN', 'WARD', 'OFFENSE', 'report_dt']].copy()

## crimes to look for matches within
other_crimes = dc_crim_2020[~dc_crim_2020.CCN.astype(str).isin(
                 CCN_examples)].copy()
```

Approach 1: loop through crimes

```
## create empty container to store results
store_matches = {}
## loop through two example crimes
for i in range(0, crimes_lookfor.shape[0]):
    ## extract row
    one_row = crimes_lookfor.iloc[i]

    ## first, subset to crimes in same ward
    same_wards = other_crimes[other_crimes.WARD == one_row.WARD].
    copy()

    ## second, with those same-ward crimes, filter to crimes within
    ## 20 minutes after focal crime
    cutoff = one_row.report_dt +  timedelta(minutes=1000)
    same_wards_sametime = same_wards[
                        (same_wards.report_dt >=  one_row.report_dt) &
                        (same_wards.report_dt <= cutoff)].copy()

    ## third, store the results
    store_matches[str(one_row.CCN)] = same_wards_sametime

## finally, rowbind results into one df
all_matches = pd.concat(store_matches)
```

How might we transition this to a function?

General structure of a function

```python

## what is the function called?
## and what are its inputs?
def do_something(search_for: pd.DataFrame,
                 search_in: pd.DataFrame):

    ## here i'm doing things
    ## (similar to meat of a for loop)
    my_output = search_in[search_in.something ==
                          search_for.something].copy()
    ## and so on...

    ## here i'm returning things
    ## (similar to last part of a loop, though don't need
    ## a within-function container)
    return(my_output)
```

Approach 2: first define a function

```python
def proximate_crimes(search_for,
                     search_in,
                     bandwidth = 1000):

    ## first, subset to crimes in same ward
    same_wards = search_in[search_in.WARD ==
                           search_for.WARD].copy()

    ## second, with those same-ward crimes, construct indicator
    ## for reported within 20 minutes
    cutoff = search_for.report_dt +  timedelta(minutes=bandwidth)
    same_wards_sametime = same_wards[
                  (same_wards.report_dt >= search_for.report_dt) &
                  (same_wards.report_dt <= cutoff)].copy()

    ## add col for focal match
    same_wards_sametime['focal_crime'] = search_for.CCN

    ## return
    return(same_wards_sametime)
```

Approach 2: applying the function

```python
## apply to a single crime
one_match = proximate_crimes(search_for = crimes_lookfor.iloc[0],
                             search_in = other_crimes)

## iterate over the crimes, apply, and rowbind results
all_matches_list = [proximate_crimes(search_for =
                     crimes_lookfor.iloc[i],
                     search_in = other_crimes)
                     for i in
                     range(0, crimes_lookfor.shape[0])]
all_matches = pd.concat(all_matches_list)
```