DataFest Prep

Part 3 of n

A data exploration checklist

- ✓ Get acquainted (with your data)
- ✓ Get curious (about your data)
- ✓ Get it loaded
- ✓ Identify the parts
- ✓ Examine each part (in isolation)
- ✓ Clean and transform (as necessary)
- ✓ Summarize each part (briefly)
- ✓ Formulate questions (for the next phase)

Teams

The goal today is **to actually choose DataFest teams for reals**.

These teams can be partially or entirely the ones you had in previous Mavens meetings, or brand new.

Every team should have 4–5 members.

Getting reacquainted

1. Find that project directory/folder in which you stored the data and your program(s).

   (...or create one if you're new)

2. Find that text document you used to store notes about what you learn
   - (maybe it's a plain text file)
   - (maybe it's a Word doc)
   - (maybe it's on Google Docs)

3. Open your tool of choice (Python, R, Excel) and make sure you can still run your code

Catching up...

`cs.umw.edu/~stephen/CPSCFeedback.csv`

`cs.umw.edu/~stephen/clean.py`

Clean and transform (as necessary)

Problems remaining from last time:

- Multi-choice columns ("`Why`" and "`Styles`")
- Ordinal variables in disguise ("`Time`," "`Level`", and "`Programming`")

Clean and transform (as necessary)

Consider the following questions. Are they easy to answer, or difficult, with the data in this present format?

1. "What percentage of students are aware of the CPSC Honor Code?"
2. "Are students who prefer more collaboration less informed about the CPSC Honor Code?"
3. "What percentage of respondents are upperclassmen?"
4. "What percentage chose CPSC for love?"
5. "What percentage prefer two or more *different* styles?"
6. "Are the ones who chose CPSC for love better programmers than the others?"
7. "Are people becoming better programmers as they continue in the major?"

Clean and transform (as necessary)

Without writing any code yet, discuss amongst yourselves how you might format/structure the data to be able to answer these questions.

You have 4 minutes. Go.

Clean and transform (as necessary)

Okay, go do that.

You have 9 minutes. Go.

What I did

```
# Add an ID column.
coded['id'] = range(len(coded))

# Separate "why" multi-choice
why_ids = []
why_vals = []

for i in range(len(coded)):

        whys = coded.iloc[i].Why.split(",")


    for why in whys:
        why_ids.append(i)
        why_vals.append(why)

whys = pd.DataFrame({'id':why_ids,'why':why_vals})
```

(and similar for "`Styles`")

What I did

```
# Add an ID column.
coded['id'] = range(len(coded))

# Separate "why" multi-choice
why_ids = []
why_vals = []

for i in range(len(coded)):
    if type(coded.iloc[i].Why) is str:
        whys = coded.iloc[i].Why.split(",")
    else:
        whys = []
    for why in whys:
        why_ids.append(i)
        why_vals.append(why)

whys = pd.DataFrame({'id':why_ids,'why':why_vals})
```

(and similar for "`Styles`")

What I did

My "`whys`" and "`styles`" DataFrames now look like this:

```
>>> print(whys.head())
   id    why
0   0   love
1   0  money
2   1  Other
3   2   love
4   2  money
```

```
>>> print(styles.head())
   id         style
0   0            XP
1   0  incremental
2   0      quizzes
3   1            XP
4   1  incremental
```

What I did

```
newlevel = coded.Level.copy()
newlevel[newlevel.isnull()] = 0
newlevel = newlevel.astype(int)
coded['Level'] = newlevel
```

(Now the `Level` column is integer, and can be meaningfully compared.)

What I did

```
replacements = {
    'Above Average':2,
    'Below Average':0,
    'Average':1
}

for old, new in replacements.items():
    coded.replace(old, new, regex=True,inplace=True)

newprogramming = coded.Programming.copy()
newprogramming[newprogramming.isnull()] = 0
newprogramming = newprogramming.astype(int)
coded['Programming'] = newprogramming
```

(Now the `Programming` column is integer, and can be meaningfully compared.)

What I did

```
replacements = {
    'More than 8':8,
    '6-8':6,
    '4-6':4,
    '2-4':2,
    'Less than 2':0
}

for old, new in replacements.items():
    coded.replace(old, new, regex=True,inplace=True)

newtime = coded.Time.copy()
newtime[newtime.isnull()] = 0
newtime = newtime.astype(int)
coded['Time'] = newtime
```

(Now the `Time` column is integer, and can be meaningfully compared.)

What I did

```
coded = coded[['id','Collaborate','Honor','Time','Level','Programming']]
```

(Don't need the messy `Why` or `Styles` columns anymore.)

What I did

We now have:

```
>>> print(coded.head())
   id    Collaborate Honor  Time  Level  Programming
    0  Collaboration   Yes     8    200            1
    1   Individually   Yes     2    200            2
    2   Individually    No     2    100            1
    3  Collaboration    No     6    200            0
    4  Collaboration   Yes     4    400            2

>>> print(whys.head())
   id    why
    0   love
    0  money
    1  Other
    2   love
    2  money

>>> print(styles.head())
   id        style
    0           XP
    0  incremental
    0      quizzes
    1           XP
    1  incremental
```

...to be continued...