Programming Principles in Python (CSCI 503/490)

Visualization

Dr. David Koop
String Methods

• Can do many of the same methods used for single strings on entire columns
• Requires `.str` prefix before calling the method
  - violations.value.str.strip().str.split(' - Comments: ')
• Also helps when extracting from a list
  - comments.str[1]
Support for Datetime

- Python has datetime library to support dates and times
- pandas has a Timestamp data type that functions somewhat similarly
- Pandas can convert timestamps
  - `pd.to_datetime`: versatile, can often guess format
- Like string methods, also a `.dt` accessor for datetime methods/properties
- With a timestamp, filtering based on datetimes becomes easier
  - `df[df['Inspection Date'] > '2021']`
Method chaining in pandas

- Tom Augspurger's post
- Effective Pandas book by Matt Harrison
- Functions written for chaining, and pipe allows custom functions

```python
def read(fp):
    df = (pd.read_csv(fp)
          .rename(columns=str.lower)
          .drop('unnamed: 36', axis=1)
          .pipe(extract_city_name)
          .pipe(time_to_datetime, ['dep_time', 'arr_time',
                                   'crs_arr_time', 'crs_dep_time'])
          .assign(fl_date=lambda x: pd.to_datetime(x['fl_date']),
                  dest=lambda x: pd.Categorical(x['dest']),
                  origin=lambda x: pd.Categorical(x['origin']),
                  tail_num=lambda x: pd.Categorical(x['tail_num']),
                  unique_carrier=lambda x: pd.Categorical(x['unique_carrier']),
                  cancellation_code=lambda x: pd.Categorical(x['cancellation_code'])))
    return df
```

D. Koop, CSCI 503/490, Spring 2023
Assignment 8

• Last Assignment
• Data and Visualization
• Pokemon Data
Data Exploration through Visualization
Transportation Data - NYC MTA
## MTA Fare Data Exploration

<table>
<thead>
<tr>
<th>REMOTE</th>
<th>STATION</th>
<th>FF</th>
<th>SEN/DIS</th>
<th>7-D AFAS UNL</th>
<th>D AFAS/RMF</th>
<th>JOINT RR TKT</th>
<th>7-D UNL</th>
<th>30-D UNL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R011 42ND STREET &amp; 8TH AVENUE</td>
<td>00228985</td>
<td>00008471</td>
<td>00000441</td>
<td>00001455</td>
<td>00000134</td>
<td>00033341</td>
<td>00071255</td>
</tr>
<tr>
<td>2</td>
<td>R170 14TH STREET-UNION SQUARE</td>
<td>00224603</td>
<td>0011051</td>
<td>00000827</td>
<td>00003026</td>
<td>00000660</td>
<td>00089367</td>
<td>00199841</td>
</tr>
<tr>
<td>3</td>
<td>R046 42ND STREET &amp; GRAND CENTRAL</td>
<td>00207758</td>
<td>00007908</td>
<td>00000323</td>
<td>00001183</td>
<td>00003001</td>
<td>00040759</td>
<td>00096613</td>
</tr>
<tr>
<td>4</td>
<td>R012 34TH STREET &amp; 8TH AVENUE</td>
<td>00188311</td>
<td>00006490</td>
<td>00000498</td>
<td>00001279</td>
<td>00003622</td>
<td>00035527</td>
<td>00067483</td>
</tr>
<tr>
<td>5</td>
<td>R293 34TH STREET - PENN STATION</td>
<td>00168768</td>
<td>00006155</td>
<td>00000523</td>
<td>00001065</td>
<td>00005031</td>
<td>00030645</td>
<td>00054376</td>
</tr>
<tr>
<td>6</td>
<td>R033 42ND STREET/TIMES SQUARE</td>
<td>00159382</td>
<td>00005945</td>
<td>00000378</td>
<td>00001205</td>
<td>00000690</td>
<td>00058931</td>
<td>00078644</td>
</tr>
<tr>
<td>7</td>
<td>R022 34TH STREET &amp; 6TH AVENUE</td>
<td>00156008</td>
<td>00006276</td>
<td>00000487</td>
<td>00001543</td>
<td>00000712</td>
<td>00058910</td>
<td>00110466</td>
</tr>
<tr>
<td>8</td>
<td>R084 59TH STREET/COLUMBUS CIRCLE</td>
<td>00155262</td>
<td>00009484</td>
<td>00000589</td>
<td>00002071</td>
<td>00000542</td>
<td>00053397</td>
<td>00113966</td>
</tr>
<tr>
<td>9</td>
<td>R020 47-50 STREETS/ROCKEFELLER</td>
<td>00143500</td>
<td>00006402</td>
<td>00000384</td>
<td>00001159</td>
<td>00000723</td>
<td>00037978</td>
<td>00090745</td>
</tr>
<tr>
<td>10</td>
<td>R179 86TH STREET-LEXINGTON AVE</td>
<td>00142169</td>
<td>00010367</td>
<td>00000470</td>
<td>00001839</td>
<td>00000271</td>
<td>00050328</td>
<td>00125250</td>
</tr>
<tr>
<td>11</td>
<td>R023 34TH STREET &amp; 6TH AVENUE</td>
<td>00134052</td>
<td>00005005</td>
<td>00000348</td>
<td>00001112</td>
<td>00000649</td>
<td>00031531</td>
<td>00075040</td>
</tr>
<tr>
<td>12</td>
<td>R029 PARK PLACE</td>
<td>00121614</td>
<td>00004311</td>
<td>00000287</td>
<td>00000931</td>
<td>00000792</td>
<td>00025404</td>
<td>00065362</td>
</tr>
<tr>
<td>13</td>
<td>R047 42ND STREET &amp; GRAND CENTRAL</td>
<td>00100742</td>
<td>00004273</td>
<td>00000185</td>
<td>00000704</td>
<td>00001241</td>
<td>00022808</td>
<td>00068216</td>
</tr>
</tbody>
</table>
MTA Fare Data Exploration
MTA Fare Data Exploration
MTA Fare Data Exploration
MTA Fare Data Exploration
MTA Fare Data Exploration

East 161st Street and River Avenue

[Bar chart showing fare data for East 161st Street and River Avenue from 08-02 to 11-01]
MTA Fare Data Exploration

East 161st Street and River Avenue

New York Yankees

AUGUST

SUN MON TUE WED THU FRI SAT SD SD SD SD

4 11 18 25 32
5 12 19 26 33
6 13 20 27 34
7 14 21 28 35
8 15 22 29 36
9 16 23 30 37
10 17 24 31 38

ALL GAMES ARE EASTERN TIME.

September

SUN MON TUE WED THU FRI SAT SD SD SD SD

1 8 15 22 29
2 9 16 23 30
3 10 17 24 31
4 11 18 25 1
5 12 19 26 2
6 13 20 27 3
7 14 21 28 4

ALL GAMES ARE EASTERN TIME.

2013 REGULAR SEASON SCHEDULE

D. Koop, CSCI 503/490, Spring 2023
“Computer-based visualization systems provide visual representations of datasets designed to help people carry out tasks more effectively” — T. Munzner
Why do we visualize data?

- Why Graphics?
- Figures are richer; provide more information with less clutter and in less space.
- Figures provide the gestalt effect: they give an overview; make structure more visible.
- Figures are more accessible, easier to understand, faster to grasp, more comprehensible, more memorable, more fun, and less formal.

List adapted from: [Stasko et al. 1998]

[T. Nørretranders]
Why Visual?

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>II</th>
<th>III</th>
<th>IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
</tr>
<tr>
<td>10.0</td>
<td>8.04</td>
<td>10.0</td>
<td>9.14</td>
<td>10.0</td>
</tr>
<tr>
<td>8.0</td>
<td>6.95</td>
<td>8.0</td>
<td>8.14</td>
<td>8.0</td>
</tr>
<tr>
<td>13.0</td>
<td>7.58</td>
<td>13.0</td>
<td>8.74</td>
<td>13.0</td>
</tr>
<tr>
<td>9.0</td>
<td>8.81</td>
<td>9.0</td>
<td>8.77</td>
<td>9.0</td>
</tr>
<tr>
<td>11.0</td>
<td>8.33</td>
<td>11.0</td>
<td>9.26</td>
<td>11.0</td>
</tr>
<tr>
<td>14.0</td>
<td>9.96</td>
<td>14.0</td>
<td>8.10</td>
<td>14.0</td>
</tr>
<tr>
<td>6.0</td>
<td>7.24</td>
<td>6.0</td>
<td>6.13</td>
<td>6.0</td>
</tr>
<tr>
<td>4.0</td>
<td>4.26</td>
<td>4.0</td>
<td>3.10</td>
<td>4.0</td>
</tr>
<tr>
<td>12.0</td>
<td>10.84</td>
<td>12.0</td>
<td>9.13</td>
<td>12.0</td>
</tr>
<tr>
<td>7.0</td>
<td>4.82</td>
<td>7.0</td>
<td>7.26</td>
<td>7.0</td>
</tr>
<tr>
<td>5.0</td>
<td>5.68</td>
<td>5.0</td>
<td>4.74</td>
<td>5.0</td>
</tr>
</tbody>
</table>

[F. J. Anscombe]
Why Visual?

<table>
<thead>
<tr>
<th>I</th>
<th>II</th>
<th>III</th>
<th>IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>10.0</td>
<td>8.04</td>
<td>10.0</td>
<td>9.14</td>
</tr>
<tr>
<td>8.0</td>
<td>6.95</td>
<td>8.0</td>
<td>8.14</td>
</tr>
<tr>
<td>13.0</td>
<td>7.58</td>
<td>13.0</td>
<td>8.74</td>
</tr>
<tr>
<td>9.0</td>
<td>8.81</td>
<td>9.0</td>
<td>8.77</td>
</tr>
<tr>
<td>11.0</td>
<td>8.33</td>
<td>11.0</td>
<td>9.26</td>
</tr>
<tr>
<td>14.0</td>
<td>9.96</td>
<td>14.0</td>
<td>8.10</td>
</tr>
<tr>
<td>6.0</td>
<td>7.24</td>
<td>6.0</td>
<td>6.13</td>
</tr>
<tr>
<td>4.0</td>
<td>4.26</td>
<td>4.0</td>
<td>3.10</td>
</tr>
<tr>
<td>12.0</td>
<td>10.84</td>
<td>12.0</td>
<td>9.13</td>
</tr>
<tr>
<td>7.0</td>
<td>4.82</td>
<td>7.0</td>
<td>7.26</td>
</tr>
<tr>
<td>5.0</td>
<td>5.68</td>
<td>5.0</td>
<td>4.74</td>
</tr>
</tbody>
</table>

Mean of x: 9
Variance of x: 11
Mean of y: 7.50
Variance of y: 4.122
Correlation: 0.816

[F. J. Anscombe]
Why Visual?

[F. J. Anscombe]
Why Visual?

Mean of $x$ 9
Variance of $x$ 11
Mean of $y$ 7.50
Variance of $y$ 4.122
Correlation 0.816

[F. J. Anscombe]
Visualization Goals

• "The purpose of visualization is **insight**, not pictures" – B. Shneiderman

• Identify patterns, trends
• Spot outliers
• Find similarities, correlation
Visual Pop-out
Visual Pop-out
Visual Pop-out
Visual Perception Limitations
Visual Perception Limitations
The Python Visualization Landscape
matplotlib

- **Strengths:**
  - Designed like Matlab
  - Many rendering backends
  - Can reproduce almost any plot
  - Proven, well-tested

- **Weaknesses:**
  - API is imperative
  - Not originally designed for the web
  - Dated styles
Altair

- Declarative Visualization
  - Specify **what** instead of how
  - Separate specification from execution
- Based on VegaLite which is browser-based
- Strengths:
  - Declarative visualization
  - Web technologies
- Drawbacks:
  - Moving data between Python and JS
  - Sometimes longer specifications
Matplotlib History

• "In the beginning was matplotlib" – J. VanderPlas
• Started by John D. Hunter, a neurobiologist ~2003
• John tragically passed away in 2012, community-led now

• Before Python, John had Perl scripts that called C++ mathematical programs that wrote data files that were plotted using Matlab (then gnuplot)
• Sought a solution that was Matlab users would be more comfortable with
  - Imports "hidden" by importing into the global namespace
  - `pylab` mode: match terminology of Matlab (at the cost of overriding core python functions/definitions)
Lots of Changes Since

- `pylab` is "strongly discouraged nowadays and deprecated." [docs]
- Stateful plotting using `pyplot` still exists, but...
- Also object-oriented methods to build and customize plots now
- Integrated output in JupyterLab
- Many derivative libraries (e.g. `seaborn`) that build on matplotlib core
- Can use more directly from pandas
matplotlib tutorials

- https://github.com/rougier/matplotlib-tutorial
Basic Example

- import matplotlib.pyplot as plt
  plt.plot([1,5,2,7,3])

- Default is line plot

- x-values are implicit (range(5))

- Can add x-values
  - plt.plot([1,3,4,6,10],[1,5,2,7,3])

- Can change type of plot
  - plt.scatter([1,3,4,6,10],[1,5,2,7,3])
  - plt.plot([1,3,4,6,10],[1,5,2,7,3],'o') # format string
Plot Formats

• Can specify color, marker, and linestyle in format string
  - `plt.plot([1,3,4,6,10],[1,5,2,7,3],'ro—')`

• Can also specify these via keyword arguments:
  - `plt.plot([1,3,4,6,10],[1,5,2,7,3],
                color='red', marker='s', linestyle='dashed')`

• Other keyword arguments, too:
  - `plt.plot([1,3,4,6,10],[1,5,2,7,3],
                color='red', marker='s', linestyle='dashed',
                linewidth=3, markersize=12)`
<table>
<thead>
<tr>
<th>string</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>'b'</td>
<td>blue</td>
</tr>
<tr>
<td>'g'</td>
<td>green</td>
</tr>
<tr>
<td>'r'</td>
<td>red</td>
</tr>
<tr>
<td>'c'</td>
<td>cyan</td>
</tr>
<tr>
<td>'m'</td>
<td>magenta</td>
</tr>
<tr>
<td>'y'</td>
<td>yellow</td>
</tr>
<tr>
<td>'k'</td>
<td>black</td>
</tr>
<tr>
<td>'w'</td>
<td>white</td>
</tr>
</tbody>
</table>

**Color shortcuts**

<table>
<thead>
<tr>
<th>string</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>'-'</td>
<td>solid</td>
</tr>
<tr>
<td>'--'</td>
<td>dashed</td>
</tr>
<tr>
<td>' -. '</td>
<td>dash-dot</td>
</tr>
<tr>
<td>':'</td>
<td>dotted</td>
</tr>
</tbody>
</table>

**Line Styles**

<table>
<thead>
<tr>
<th>string</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>'.'</td>
<td>point</td>
</tr>
<tr>
<td>','</td>
<td>pixel</td>
</tr>
<tr>
<td>'o'</td>
<td>circle</td>
</tr>
<tr>
<td>'v'</td>
<td>triangle_down</td>
</tr>
<tr>
<td>'^'</td>
<td>triangle_up</td>
</tr>
<tr>
<td>'&lt;'</td>
<td>triangle_left</td>
</tr>
<tr>
<td>'&gt;'</td>
<td>triangle_right</td>
</tr>
<tr>
<td>'1'</td>
<td>tri_down</td>
</tr>
<tr>
<td>'2'</td>
<td>tri_up</td>
</tr>
<tr>
<td>'3'</td>
<td>tri_left</td>
</tr>
<tr>
<td>'4'</td>
<td>tri_right</td>
</tr>
<tr>
<td>'8'</td>
<td>octagon</td>
</tr>
<tr>
<td>'s'</td>
<td>square</td>
</tr>
</tbody>
</table>

**Markers**

<table>
<thead>
<tr>
<th>string</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>'p'</td>
<td>pentagon</td>
</tr>
<tr>
<td>'P'</td>
<td>plus (filled)</td>
</tr>
<tr>
<td>'*'</td>
<td>star</td>
</tr>
<tr>
<td>'h'</td>
<td>hexagon1</td>
</tr>
<tr>
<td>'H'</td>
<td>hexagon2</td>
</tr>
<tr>
<td>'+'</td>
<td>plus</td>
</tr>
<tr>
<td>'x'</td>
<td>x</td>
</tr>
<tr>
<td>'X'</td>
<td>x (filled)</td>
</tr>
<tr>
<td>'D'</td>
<td>diamond</td>
</tr>
<tr>
<td>'d'</td>
<td>thin_diamond</td>
</tr>
<tr>
<td>'</td>
<td>'</td>
</tr>
<tr>
<td>'_'</td>
<td>hline</td>
</tr>
</tbody>
</table>
Data is Encoded via Visual Channels

- **Position**
  - Horizontal
  - Vertical
  - Both

- **Color**

- **Shape**

- **Tilt**

- **Size**
  - Length
  - Area
  - Volume

[Munzner (ill. Maguire), 2014]
Encoding Data Attributes via Channels

- data = {'age': [1, 3, 4, 6, 10],
  'num_jumps': [1, 5, 2, 7, 3],
  'weight': [20, 50, 25, 55, 25],
  'num_scoops': [3, 2, 4, 2, 3]}
plt.scatter('age', 'num_jumps', c='num_scoops', s='weight', data=data)

- data is a dictionary that contains information about each data item (first animal has age=1, num_jumps=1, weight=20, num_scoops=3)

- x and y are referenced as parts of the array

- s is marker size

- c is color and numbers are mapped to colors
Many different types of charts
Many different types of charts

• Bar chart
  - `plt.bar(['Apple','Banana','Orange'],[0.99,0.50,1.25])`

• Grid Heatmap
  - `plt.pcolormesh(x, y, Z)`

• Pie chart:
  - `plt.pie([20,40,30,10],
            labels=['Apple','Banana','Orange','Pear'])`
Adding Labels

- `plt.xlabel`: set x label
- `plt.ylabel`: set y label
- `plt.title`: set title
- `plt.plot([1,3,4,6,10],[1,5,2,7,3])`
  - `plt.xlabel('Age')`
  - `plt.ylabel('Number of Jumps')`
  - `plt.title('Kangaroo Jumps Today')`
Anatomy of a Figure

Anatomy of a Figure

D. Koop, CSCI 503/490, Spring 2023
Figure and Axes Objects

- pyplot is stateful, functions affect the "current" figure and axes
  - `plt.gcf()`: gets current figure
  - `plt.gca()`: gets current axes
    - Creates one if it doesn't exist!
- This is not aligned with object-based programming ideas
- Most methods in pyplot are translated to methods on the current axes (gca)
- We can instead call these directly, but first need to create them:
  - `fig, ax = plt.subplots()` # "constructor-like" method
    - `ax.scatter([1,3,4,6,10],[1,5,2,7,3])`
Object-Based Plotting

• `fig, ax = plt.subplots()` # "constructor-like" method
  `ax.scatter([1,3,4,6,10],[1,5,2,7,3])`

• Use getters/setters for labels and title
  - `ax.set_xlabel('Age')`
  - `ax.set_ylabel('Number of Jumps')`
  - `ax.set_title('Kangaroo Jumps Today')`

• We can also call methods on the figure:
  - `fig.tight_layout()` # reduce margins
Multiple Figures

• subplots allows multiple axes in the same figure:
  - `fig, ax = plt.subplots(2, 2, figsize=(10, 10))` # rows, then columns

• `ax` is now a 2x2 numpy array

• Can put any type of visualization on each pair of axes
  • `ax[0,0].plot([1,3,4,6,10],[1,5,2,7,3])`
  • `ax[0,1].bar(['Apple','Banana','Orange'],[0.99,0.50,1.25])`
  • `ax[1,0].pcolormesh(x, y, Z)`
  • `ax[1,1].pie([20,40,30,10],
                 labels=['Apple','Banana','Orange','Pear'])`
pandas Integration

• Can call many of these methods directly from pandas
• Handled through \texttt{kind} kwarg or \texttt{.plot} accessor
• It will try to guess a reasonable visualization, but may fail:
  - \texttt{fruit.plot()}

• Instead, specify \texttt{x} and \texttt{y} and other parameters:
  - \texttt{fruit.plot(kind='bar',x='name',y='price')}
  - \texttt{plt.bar(x='name',height='price',data=fruit)} \# SIMILAR
  - \texttt{fruit.plot.scatter(x='price',y='count',c='name')} \# ERROR
  - \texttt{colors = \{'Apple': 'red','Orange': 'orange',}
    \texttt{'Banana': 'yellow','Pear': 'green'}
    \texttt{fruit.plot.scatter(x='price',y='count',}
    \texttt{c=fruit['name'].map(colors))}
Extensions & Other Directions

• Seaborn:

- import seaborn as sns
  sns.scatterplot(x='price', y='count', hue='name', data=fruit)