Programming Principles in Python (CSCI 503/490)

Visualization

Dr. David Koop
Exploring Data through Visualization
Exploring Data through Visualization
Why do we visualize data?

Total Bandwidth
(millions of bits per second)

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]

via A. Lex

T. Nørretranders
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>

[F. J. Anscombe]
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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.0</td>
<td>8.04</td>
<td>10.0</td>
<td>10.0</td>
<td>8.0</td>
</tr>
<tr>
<td>8.0</td>
<td>6.95</td>
<td>8.0</td>
<td>8.0</td>
<td>6.77</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>

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?

[D. Koop, CSCI 503/490, Spring 2022]

[F. J. Anscombe]
Why Visual?

<table>
<thead>
<tr>
<th></th>
<th>Mean of x</th>
<th>Variance of x</th>
<th>Mean of y</th>
<th>Variance of y</th>
<th>Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>9</td>
<td>11</td>
<td>7.50</td>
<td>4.122</td>
<td>0.816</td>
</tr>
<tr>
<td>x2</td>
<td>9</td>
<td>11</td>
<td>7.50</td>
<td>4.122</td>
<td>0.816</td>
</tr>
<tr>
<td>x3</td>
<td>9</td>
<td>11</td>
<td>7.50</td>
<td>4.122</td>
<td>0.816</td>
</tr>
<tr>
<td>x4</td>
<td>9</td>
<td>11</td>
<td>7.50</td>
<td>4.122</td>
<td>0.816</td>
</tr>
</tbody>
</table>

[F. J. Anscombe]
Quiz
Question 1

- Which best describes a pandas Series?
  (a) a numpy array
  (b) an immutable list
  (c) two paired numpy arrays
  (d) multiple numpy arrays that share the same index
Question 2

• \( \text{pd.Series}({'a': 1, 'b': 2, 'c': 3}) + \text{pd.Series}({'c': 3, 'a': 1, 'b': 2}) \) = ?

(a) \( \text{pd.Series}({'a': 4, 'b': 3, 'c': 5}) \)
(b) \( \text{pd.Series}({'c': 4, 'a': 3, 'b': 5}) \)
(c) \( \text{pd.Series}({'a': 2, 'b': 4, 'c': 6}) \)
(d) None of the above
Question 3

• What is the shape of np.array([[1, 2, 3], [4, 5, 6]])?
  (a) (2, 3)
  (b) (6,)
  (c) (2, -1)
  (d) (3, 2)
Question 4

- What is the purpose of data visualization, according to the quote from class?
  (a) insight
  (b) pictures
  (c) sleep
  (d) efficiency
Question 5

• Which pandas method does the split in split-apply-combine?
  (a) pivot
  (b) melt
  (c) groupby
  (d) split
Visualization Goals

- "The purpose of visualization is **insight**, not pictures" – B. Schneiderman

- Identify patterns, trends
- Spot outliers
- Find similarities, correlation
The Python Visualization Landscape
The Python Visualization Landscape

D. Koop, CSCI 503/490, Spring 2022
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
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
Data is Encoded via Visual Channels

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

- **Color**

- **Shape**

- **Tilt**

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

[Munzner (ill. Maguire), 2014]
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'])`
Many different types of charts
Anatomy of a Figure

- **Figure**
- **Axes**
- **Title**
- **Legend**

- **Axes**
  - **X label**
  - **Y label**

- **Legend**
  - Blue line: Signal
  - Red line: Scatter plot

- **Markers**
  - Circle markers

- **Spines**

- **Grid**

- **Legend**
  - Legend icons

- **Title**

- **Axes**
  - **X axis label**
  - **Y axis label**

- **Minor tick**
- **Major tick**
- **Tick label**

[D. Koop, CSCI 503/490, Spring 2022]
pandas Integration

- Can call many of these methods directly from pandas
- Handled through `kind` kwarg or `.plot` accessor
- It will try to guess a reasonable visualization, but may fail:
  - `fruit.plot()`
- Instead, specify x and y and other parameters:
  - `fruit.plot(kind='bar', x='name', y='price')`
  - `plt.bar(x='name', height='price', data=fruit) # SIMILAR`
  - `fruit.plot.scatter(x='price', y='count', c='name') # ERROR`
  - `colors = {'Apple': 'red', 'Orange': 'orange',
               'Banana': 'yellow', 'Pear': 'green'}`
  - `fruit.plot.scatter(x='price', y='count',
                        c=fruit['name'].map(colors))`
Extensions & Other Directions

• Seaborn:
  
  - import seaborn as sns
  
  sns.scatterplot(x='price', y='count', hue='name', data=fruit)
Assignment 8

• Illinois Employment Data (same as A7)
• Data Manipulation using pandas
• Visualization using matplotlib and altair
• Due next Thursday
Final Exam

- Monday, May 9, 2:00-3:50pm in PM 153
- **More** comprehensive than Test 2
- Expect questions from topics covered on Test 1 and 2
- Expect questions from the last four weeks of class (concurrency, data, visualization, machine learning)
- Similar format
History of Vega-Lite & Altair

- "Grammar of Graphics", L. Wilkinson
- "A Layered Grammar of Graphics", H. Wickham
- ggplot: plotting library for R
- Vega: similar idea for Javascript/JSON (U. Washington, A. Satyanarayan)
  - "Declarative language for creating, saving, and sharing interactive visualization designs"
  - More focus on interaction and reactive signals
  - Separation between specification and runtime
- Vega-Lite: higher-level language than Vega (U. Washington, D. Moritz)
  - uses carefully designed rules to default settings
History of Vega-Lite & Altair

- Altair: Python interface to Vega-Lite (J. VanderPlas)
  - "spend more time understanding your data and its meaning"
  - Specify the what, minimize the amount of code directing the how
  - Python can write JSON specification just as well as any other language
  - Bindings make it more Python-friendly, integrate with pandas, add support for Jupyter, etc.
Basic Example

- import altair as alt
  import pandas as pd
  data = pd.DataFrame({'x': [1,3,4,6,10], 'y': [1,5,2,7,3]})
  alt.Chart(data).mark_line().encode(x='x', y='y')

- Easiest to use data from a pandas data frame
  - Another option is a csv or json file
  - Can support geo_interface, too
- Chart is the basic unit
- Mark: .mark_*() indicates the geometry created for each data item
- Encode: .encode() allows visual properties to be set to data attributes
Visual Marks

- **Marks** are the basic graphical elements in a visualization
- Marks classified by dimensionality:
  - **Points**
  - **Lines**
  - **Areas**
- Also can have surfaces, volumes
- Think of marks as a mathematical definition, or if familiar with tools like Adobe Illustrator or Inkscape, the path & point definitions
- Altair: area, bar, circle, geoshape, image, line, point, rect, rule, square, text, tick
  - Also compound marks: boxplot, errorband, errorbar
Encode via Visual Channels

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

- **Color**

- **Shape**

- **Tilt**

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

[Munzner (ill. Maguire), 2014]
Easily Explore Different Encodings

```python
# data = pd.DataFrame({
    'age': [1, 3, 4, 6, 10],
    'weight': [20, 50, 25, 55, 125],
    'zoo_area': [1, 3, 3, 1, 2],
    'num_scoops': [3, 2, 4, 2, 3]
})
alt.Chart(data).mark_point(
    filled=True, size=50,
    stroke='black', strokeWidth=1).encode(
    x='age',
    y='weight',
    color='zoo_area'
)
```
Problem: zoo_area is not a continuous value, nor is it ordered in any way!
Data Attributes and Altair Types

➡️ Categorical

➡️ Ordered

➡️ Ordinal

➡️ Quantitative

[Munzner (ill. Maguire), 2014]
Data Attributes and Altair Types

- Categorical data = Nominal (N)
- Ordinal data = Ordinal (O)
- Quantitative data = Quantitative (Q)
- Temporal data = Temporal (T)

[Munzner (ill. Maguire), 2014]
Specifying the Type

\[ \text{zoo\_area:O} \quad \text{zoo\_area:N} \]
### Different Channels for Different Attribute Types

<table>
<thead>
<tr>
<th><strong>Magnitude Channels: Ordered Attributes</strong></th>
<th><strong>Identity Channels: Categorical Attributes</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Position on common scale</td>
<td>Spatial region</td>
</tr>
<tr>
<td>Position on unaligned scale</td>
<td>Color hue</td>
</tr>
<tr>
<td>Length (1D size)</td>
<td>Motion</td>
</tr>
<tr>
<td>Tilt/angle</td>
<td>Shape</td>
</tr>
<tr>
<td>Area (2D size)</td>
<td></td>
</tr>
<tr>
<td>Depth (3D position)</td>
<td></td>
</tr>
<tr>
<td>Color luminance</td>
<td></td>
</tr>
<tr>
<td>Color saturation</td>
<td></td>
</tr>
<tr>
<td>Curvature</td>
<td></td>
</tr>
<tr>
<td>Volume (3D size)</td>
<td></td>
</tr>
</tbody>
</table>

Altair will use its rules to pick whether to use color hue or saturation based on the type.

[Munzner (ill. Maguire), 2014]
Multiple Views in Visualization
Multiple Views in Visualization

[Improvise, Weaver, 2004]
Multiple Views in Visualization

- Sepal length
- Sepal width
- Petal length
- Petal width

[Image of scatter plots showing multiple views of the iris dataset]

[M. Bostock]
Altair Supports Concatenation, Layering, & Repetition

• Layering:
  - + Operator

• Concatenation:
  - Horizontal: | operator
  - Vertical: & operator

• Repetition
  - Use of .repeat for layout
  - Reference repeated variables in the encoding
Visualization

[Rock 'N' Roll is Here to Pay, R. Garofalo, 1977 (via Tufte)]
Also Visualization, but with Interaction

[Music Timeline, Google Research (no working version)]
Interaction

- Grammar of Graphics, why not Grammar of Interaction?
- Vega-Lite/Altair is about interactive graphics
- Types of Interactions:
  - Selection
  - Zoom
  - Brushing
Selection

- Selection is often used to initiate other changes
- User needs to select something to drive the next change
- What can be a selection target?
  - Items, links, attributes, (views)
- How?
  - mouse click, mouse hover, touch
  - keyboard modifiers, right/left mouse click, force
- Selection modes:
  - Single, multiple
  - Contiguous?
Highlighting

- Selection is the user action
- Feedback is important!
- How? Change selected item's visual encoding
  - Change color: want to achieve visual popout
  - Add outline mark: allows original color to be preserved
  - Change size (line width)
  - Add motion: marching ants
Highlighting

- Selection is the user action
- Feedback is important!
- How? Change selected item's visual encoding
  - Change color: want to achieve visual popout
  - Add outline mark: allows original color to be preserved
  - Change size (line width)
  - Add motion: marching ants
Altair's Interactive Charts

Weather Selection: Rain vs. Sun

Seattle Weather: 2012-2015

D. Koop, CSCI 503/490, Spring 2022
Date Selection: July-September Sun