Programming Principles in Python (CSCI 503/490)

Visualization & Machine Learning

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

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?

\[\begin{align*}
\text{\(Y_1\)} & \quad \text{\(X_1\)} \\
\text{\(Y_2\)} & \quad \text{\(X_2\)} \\
\text{\(Y_3\)} & \quad \text{\(X_3\)} \\
\text{\(Y_4\)} & \quad \text{\(X_4\)}
\end{align*}\]

[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

---

[D. Koop, CSCI 503/490, Fall 2022]
Visualization Goals

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

• Identify patterns, trends
• Spot outliers
• Find similarities, correlation
Data is Encoded via Visual Channels

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

- **Color**

- **Shape**

- **Tilt**

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

[Munzner (ill. Maguire), 2014]
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
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))
Assignment 8

• Due Friday
• Data and Visualization
• Port of Entry Data
• Part 1a: Month includes Year—the whole Data column (e.g. Jan 2022)
Final Exam

- Tuesday, December 6, **12:00-1:50pm** in PM 253
- More comprehensive than Test 2
- Expect questions from topics covered on Test 1 and 2
- Expect questions from the last four weeks of class (data, visualization, machine learning)
- Similar format
The Python Visualization Landscape
The Python Visualization Landscape

- seaborn
- pandas
- ggpy
- scikit-plot
- yellowbrick
- networkx
- basemap
- pythreejs
- bqplot
- bokeh
- toyplot
- matplotlib
- datashader
- d3js
- violin
- matplotlib
- Altair
- vega-lite
- vega
- plotnine
- cufflinks
- plotly
- ipyvolume
- ipyleaflet
- pythreejs
- d3po
- mpld3
- altair
- glueviz
- galpy
- galpy
- pyglet
- GR Framework
- mayavi
- vispy
- glumpy
- vincent
- d3js
- vega
- pandas
- seaborn
- pygal
- chaco
- PyQtGraph

+ ipyparaview
+ pydeck

D. Koop, CSCI 503/490, Fall 2022
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)
Specifying the Type

\[ \text{zoo\_area}:O \]

\[ \text{zoo\_area}:N \]
### 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
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
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

Seattle Weather: 2012-2015

count of records

weather

precipitation

precipitation

interaction
Weather Selection: Rain vs. Sun

Seattle Weather: 2012-2015

Weather Selection: Rain vs. Sun

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

Seattle Weather: 2012-2015

- Maximum Daily Temperature (°C)
- Date

Count of Records

- weather
  - sun
  - fog
  - drizzle
  - rain
  - snow

- precipitation
  - 0
  - 10
  - 20
  - 30
  - 40
  - 50

D. Koop, CSCI 503/490, Fall 2022
Machine Learning in Python
Tasks Machine Learning can Help With

• Identifying the zip code from handwritten digits on an envelope

• Detecting fraudulent activity in credit card transactions
• Identifying topics in a set of blog posts
• Grouping customers with similar preferences

[A. Müller & S. Guido, Introduction to Machine Learning with Python, J. Steppan (MNIST image)]
When to Use Machine Learning?

• ML is used when:
  - Human expertise does not exist (navigating on Mars)
  - Humans can’t explain their expertise (speech recognition)
  - Models must be customized (personalized medicine)
  - Models are based on huge amounts of data (genomics)

• ML isn’t always useful:
  - Calculating payroll…

[E. Alpaydin via E. Eaton]
Questions when building a machine learning solution

• What question(s) am I trying to answer? Do I think the data collected can answer that question?
• What is the best way to phrase my question(s) as a machine learning problem?
• Have I collected enough data to represent the problem I want to solve?
• What features of the data did I extract, and will these enable the right predictions?
• How will I measure success in my application?
Machine Learning Workflow Overview

1. Should I use ML on this problem?
   - Is there a pattern to detect? Can I solve it analytically? Do I have data?
2. Gather and organize data.
   - Preprocessing, cleaning, visualizing.
3. Establishing a baseline.
4. Choosing a model, loss, regularization, …
5. Optimization (could be simple, could be a Phd…).
6. Hyperparameter search.
7. Analyze performance & mistakes, and iterate back to step 4 (or 2).
Machine Learning

- Traditional Programming

Data → Computer → Output
Program → Computer

- Machine Learning

Data → Computer → Program
Output → Computer
Machine Learning

• Every machine learning algorithm has three components:
  - Representation
  - Evaluation
  - Optimization
Representation

- Decision trees
- Sets of rules / Logic programs
- Instances
- Graphical models (Bayes/Markov nets)
- Neural networks
- Support vector machines
- Model ensembles
- Etc.
Evaluation

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Cost / Utility
- Margin
- Entropy
- K-L divergence
- Etc.
Optimization

- Combinatorial optimization
  - E.g.: Greedy search
- Convex optimization
  - E.g.: Gradient descent
- Constrained optimization
  - E.g.: Linear programming
Types of Learning

• Supervised (inductive) learning
  - Training data includes desired outputs

• Unsupervised learning
  - Training data does not include desired outputs

• Semi-supervised learning
  - Training data includes a few desired outputs

• Reinforcement learning
  - Rewards from sequence of actions
Areas of Machine Learning

- Supervised learning
  - Decision tree induction
  - Rule induction
  - Instance-based learning
  - Bayesian learning
  - Neural networks
  - Support vector machines
  - Model ensembles
  - Learning theory

- Unsupervised learning
  - Clustering
  - Dimensionality reduction
Supervised & Unsupervised Tasks

• Identifying the zip code from handwritten digits on an envelope (supervised)

• Detecting fraudulent activity in credit card transactions (supervised)

• Identifying topics in a set of blog posts (unsupervised)

• Grouping customers with similar preferences (unsupervised)

[A. Müller & S. Guido, Introduction to Machine Learning with Python, J. Steppan (MNIST image)]
Supervised Learning
Supervised Learning: Learned Algorithm (Fit)
Supervised Learning: Prediction
Supervised Learning: Prediction
Unsupervised Learning: Input
Unsupervised Learning: Output
Scikit-Learn

- Started as a Google Summer of Code project! (D. Cournapeau, 2007)
- Rewritten by scientists at INRIA (France) in 2010
- Written in Python using numpy, some optimizations using C (cython)
- The "gold standard" for machine learning in python
scikit-learn Principles

• Consistency: all objects share consistent, documented interface
• Inspection: parameters and parameter values determined by learning algorithms are stored and exposed as public attributes
• Non-proliferation of classes: only learning algos are classes, not datasets or parameters; easier to combine with other libraries
• Composition: create and reuse building blocks
• Sensible defaults: user-defined parameters should have meaningful defaults
scikit-learn entities

- Data: numpy matrices (also pandas series, data frames), process batches
- Estimators: all supervised & unsupervised algs implement `common` interface
  - estimator initialization does not do learning, only attaches parameters
  - `fit` does the learning, learned parameters exposed with trailing underscore
- Predictor: extends estimator with `predict` method
  - also provides `score` method to return value indicating prediction quality
- Transformer: help modify or filter data before learning
  - Preprocessing, feature selection, feature extraction, and dimensionality reduction via `transform` method
  - Can combine `fit` and `transform` via `fit_transform`
Penguin Example
Deep Learning

- Deep learning is tied to neural networks, attempting to mimic how human neurons work together
- Hierarchical with multiple layers
- Usually takes advantage of GPUs
- Frameworks:
  - pytorch
  - TensorFlow
  - keras
  - theano