Data Visualization in Notebooks

David Koop
Northern Illinois University
Also Visualizations

Histogram showing distribution of values with steps.

Graph illustrating changes over steps with max, mean, and min values.
"Scientific" and "Information" Visualization

- Two subfields of visualization
- **Scientific visualization (SciVis)** deals with data where the spatial position is given with data
  - Usually continuous data
  - Often displaying physical phenomena
  - Techniques like isosurfacing, volume rendering, vector field vis
- **Information visualization (InfoVis)** deals with data that has no set spatial representation; the designer chooses how to visually represent data
SciVis
InfoVis
"The purpose of visualization is insight, not pictures"

– B. Schneiderman
Data Analysis

Data
Data Analysis

Data  Computation
Data Analysis

Data -> Computation -> Output
Data Analysis

Data → Computation → Output → Perception & Cognition
Data Analysis

Data → Computation → Output → Perception & Cognition → Knowledge

- Data
- Computation
- Output
- Perception & Cognition
- Knowledge
Data Analysis

Data -> Computation -> Output -> Perception & Cognition -> Knowledge

- Data
- Computation
- Output
- Perception & Cognition
- Knowledge
Data Analysis

Data → Computation → Output → Perception & Cognition → Knowledge

Explore
Data Analysis

Data → Computation → Output → Perception & Cognition → Knowledge

Explore
Outputs Often Become Inputs
Visualization Landscape

- **Apps:** ParaView, VisIt

- **Domain-Specific Apps:** VMD, VAPOR

- **APIs:** VTK, ITK

- **Also… Data Analysis Tools/Libraries**
  - JavaScript: D3, Observable Plot
  - R: ggplot
  - Python: matplotlib, altair, bokeh, …
  - Matlab
  - GNUPlot
Different Types of Notebooks, Many Similarities
Different Types of Notebooks, Many Similarities
Different Types of Notebooks, Many Similarities
Focus on Jupyter Notebooks

Open a CSV file using Pandas

```
In [17]: import pandas
df = pandas.read_csv('..data/iris.csv')
df.head(5)
```

```
Out[17]:
   sepal_length  sepal_width  petal_length  petal_width  species
0       5.1         3.5          1.4          0.2       setosa
1       4.9         3.0          1.4          0.2       setosa
2       4.7         3.2          1.3          0.2       setosa
3       4.6         3.1          1.5          0.2       setosa
4       5.0         3.6          1.4          0.2       setosa
```

```
In [20]: from IPython.display import GeoJSON
GeoJSON(s, layer_options={"minZoom": 11})
```
The Lorenz Differential Equations

Before we start, we import some preliminary libraries. We will also import (below) the accompanying \texttt{lorenz.py} file, which contains the actual solver and plotting routine.

\begin{verbatim}
from ipywidgets import interactive, fixed
\end{verbatim}

We explore the Lorenz system of differential equations:

\begin{align*}
  \dot{x} &= \sigma(y-x) \\
  \dot{y} &= \rho x - y - xz \\
  \dot{z} &= -\beta z + xy
\end{align*}

Let's change ($\sigma$, $\beta$, $\rho$) with ipywidgets and examine the trajectories.

\begin{verbatim}
from lorenz import solve_lorenz
w=interactive(solve_lorenz,sigma=(0.0,50.0),rho=(0.0,50.0))
w
sigma  10.00
beta  2.67
\end{verbatim}
The Lorenz Differential Equations

Before we start, we import some preliminary libraries. We will also import (below) the accompanying `lorenz.py` file, which contains the actual solver and plotting routine.

```python
from ipywidgets import interactive, fixed
```

We explore the Lorenz system of differential equations:

\[
\begin{align*}
\dot{x} &= \sigma(y - x) \\
\dot{y} &= \rho x - y - xz \\
\dot{z} &= -\beta z + xy
\end{align*}
\]

Let's change \((\sigma, \beta, \rho)\) with ipywidgets and examine the trajectories.

```python
from lorenz import solve_lorenz
w=interactive(solve_lorenz,sigma=(0.0,50.0),rho=(0.0,50.0))
w
```

<table>
<thead>
<tr>
<th>sigma</th>
<th>beta</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2.67</td>
</tr>
</tbody>
</table>
Notebooks Are Great for Exploration

In[1]:

Out[1]:

In[2]:

Out[2]:
f(Out[1], ...)

Out[2]:
Notebooks Are Great for Exploration

- Digestible Blocks of Code

In[1]:

Out[1]:

In[2]:

Out[2]:

f(Out[1], ...)

Out[2]:
Notebooks Are Great for Exploration

• Digestible Blocks of Code
• Rich, Inline Outputs (inc. widgets)
Notebooks Are Great for Exploration

- Digestible Blocks of Code
- Rich, Inline Outputs (inc. widgets)
- Reuse Existing Outputs
Notebooks Are Great for Exploration

- Digestible Blocks of Code
- Rich, Inline Outputs (inc. widgets)
- Reuse Existing Outputs
- Non-linear editing

```
In[2]:
Out[2]:
```

```
In[3]:
Out[3]:
```

```
In[2]: f(Out[1], ...)
Out[2]:
```
Support for Rapid Exploration

- Flexible environment
  - Edit any cell whenever you want
  - Execute whichever cells you want
- Inline views of outputs
  - No context switch
  - Easily compare and trace outputs
- Explore data in situ
  - Notebooks run in browser
  - Kernels can run remotely
Notebooks Are Great for Explanation

Gene Expression Data

We obtained gene expression data from the Cancer Cell Line Encyclopedia (CCLE) for 37 lung cancer cell lines assayed by our collaborators at CST. This independent dataset can be used to find novel correlations between differentially expressed genes and PTMs as well as determine whether lung cancer cell lines behave similarly in gene-expression-space and PTM-space. The gene expression data was processed in the CST_Data_Processing.ipynb notebook that kept the top 1000 genes with the greatest variance across the cell lines, and Z-score normalized the genes across the cell lines to highlight differential expression across the lung cancer cell lines.

In [4]:
net.load_file(’../lung_cellline_3.1.16/lung_cl_all_pmt/precalc_processed/CST_CCLS_exp.txt’)
print(’Expression data shape: ’ + str(net.dat[’mat’].shape))

Expression data shape: (1000, 37)

In [5]:
net.set_cat_color(’row’, 1, ’Data-Type: Exp’, ’yellow’)
net.cluster(views=[])
net.widget()
Support for Clear Explanation

- Textual explanation: markdown cells
- Graphical explanation: inline figures
- Interactive explanation: widgets
- Publishing: Web pages, LaTeX, etc.
- Structure: clear, linear cell layout
- Reproducible
```
[1]: import pandas as pd
[2]: df = pd.read_csv('penguins_size.csv')
[3]: df = df.dropna(subset=['culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm', 'body_mass_g'])
[4]: df.columns
[5]: df = df.rename(columns={'culmen_length_mm': 'culmen_length (mm)', 'culmen_depth_mm': 'body_mass (g)'})
[6]: df['body_mass (g)'].max()
[7]: df.groupby('island')['body_mass (g)'].median()
```

**Exemplar Notebook**

---

```
Name: body mass (g), dtype: float64
```

---

```
Torgersen    3700.0
5 400.0
MAL E
```

---

```
Biscoe       4775.0
49. 9
5 0.9
MAL E
```

---

```
Ch instrap
... ...
342 Adelie Torgersen NaN ... NaN NaN
343 Chinstrap Dream 470 ... 3700.0 FEMALE
```

---

```
0 Chinstrap Dream 50.9 ... 3550.0 Male
1 Gentoo Biscoe 473 ... 4725.0 NaN
... ... ... ... ... ...
341 Gentoo Biscoe 49.9 ... 5400.0 Male
343 Chinstrap Dream 470 ... 3700.0 Female
```

---

```
343 Chinstrap Dream 470 ... 3700.0 Female
```

---

```
Biscoe       4775.0
island
```

---

```
Ch instrap
... ...
```

---

```
sex
```

---

```
Torgersen
3700.0
```

---

```
Biscoe       4775.0
island
```

---

```
name: body mass (g), dtype: float64
```
Exemplar Notebook

```python
import pandas as pd

df = pd.read_csv('penguins_size.csv')

df.columns

Index(['species', 'island', 'culmen_length_mm', 'culmen_depth_mm',
       'flipper_length_mm', 'body_mass_g', 'sex'],
      dtype='object')

df = df.rename(columns={'culmen_length_mm': 'culmen length (mm)',
                         'culmen_depth_mm': 'culmen length (mm)',
                         'body_mass_g': 'body mass (g)'}

df.columns

Index(['species', 'island', 'culmen_length_mm', 'culmen_depth_mm',
       'flipper_length_mm', 'body_mass_g', 'sex'],
      dtype='object')

df = df.dropna(subset=['culmen_length_mm', 'culmen_depth_mm',
                        'flipper_length_mm', 'body_mass_g'])

df['body mass (g)'].max()

6300.0

df.groupby('island')['body mass (g)'].median()

island
  Biscoe  4775.0
  Dream   3687.5
  Torgersen  3700.0
Name: body mass (g), dtype: float64
```
```python
import pandas as pd

df = pd.read_csv('penguins_size.csv')
```

<table>
<thead>
<tr>
<th>species</th>
<th>island</th>
<th>culmen_length_mm</th>
<th>body_mass_g</th>
<th>sex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chirntrap</td>
<td>Dream</td>
<td>50.9</td>
<td>3550.0</td>
<td>MALE</td>
</tr>
<tr>
<td>Gentoo</td>
<td>Biscoe</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>Chirntrap</td>
<td>Dream</td>
<td>470.0</td>
<td>3700.0</td>
<td>FEMALE</td>
</tr>
</tbody>
</table>

```python
df.dropna(subset=[
    'culmen_length_mm',
    'culmen_depth_mm',
    'flipper_length_mm',
    'body_mass_g',
])
```

<table>
<thead>
<tr>
<th>species</th>
<th>island</th>
<th>culmen_length_mm</th>
<th>body_mass_g</th>
<th>sex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chirntrap</td>
<td>Dream</td>
<td>50.9</td>
<td>3550.0</td>
<td>MALE</td>
</tr>
<tr>
<td>Gentoo</td>
<td>Biscoe</td>
<td>473.0</td>
<td>4725.0</td>
<td>NaN</td>
</tr>
<tr>
<td>Chirntrap</td>
<td>Dream</td>
<td>470.0</td>
<td>3700.0</td>
<td>FEMALE</td>
</tr>
</tbody>
</table>

```python
df['culmen_length_mm'].max()
```

6300.0

```python
df.groupby('island')['body_mass_g'].median()
```

<table>
<thead>
<tr>
<th>island</th>
<th>body_mass_g</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biscoe</td>
<td>4775.0</td>
</tr>
<tr>
<td>Dream</td>
<td>3687.5</td>
</tr>
<tr>
<td>Torgersen</td>
<td>3700.0</td>
</tr>
</tbody>
</table>

Name: body_mass_g, dtype: float64
```python
# Import pandas library
import pandas as pd

# Read CSV file
df = pd.read_csv('penguins_lter.csv')

# Rename columns
df = df.rename(columns={'culmen_length_mm': 'culmen length (mm)', 'culmen_depth_mm': 'body mass (g)', 'flipper_length_mm': 'flipper length (mm)'}

# Possible KeyError: 'body_mass_g'

# Use dropna method to handle missing values
df = df.dropna(subset=['culmen_length_mm', 'culmen_length_mm', 'flipper_length_mm', 'body_mass_g'])

# Display columns
print(df.columns)

# Use groupby method
study_name = df.groupby('study name')['body mass (g)'].median()

# Display max of 'body mass (g)'
print(df['body mass (g)'].max())

# Display median of 'body mass (g)'
print(study_name)
```

**Key Points:**
- The code attempts to rename columns with existing column names, which could cause a `KeyError`.
- The `dropna` function is used to handle missing values, which can help in avoiding `KeyError`.
- The `groupby` function is used to group data by study name and calculate the median of the body mass column.

**Traceback:**
```
[1]: 
[2]: 
[3]: 
[4]: 
[5]: 
[6]: 
[7]: 
[8]: 
[9]: 
[10]: 

KeyError: ['culmen_length_mm', 'culmen_length_mm', 'flipper_length_mm', 'body_mass_g']
```

**Error Message:**
```
KeyError: ['culmen_length_mm', 'culmen_length_mm', 'flipper_length_mm', 'body_mass_g']
```
### Confusing Notebook

```
[1]: import pandas as pd
[5]: df = pd.read_csv('penguins_lter.csv')

```

```
<table>
<thead>
<tr>
<th>study name</th>
<th>sample number</th>
<th>species</th>
<th>culmen length (mm)</th>
<th>body mass (g)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAL0708</td>
<td>1</td>
<td>Adelie</td>
<td>39.1</td>
<td>3750.0</td>
</tr>
<tr>
<td>PAL0708</td>
<td>1</td>
<td>Gentoo</td>
<td>46.1</td>
<td>45000.0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>342</td>
<td>151</td>
<td>Adelie</td>
<td>36.0</td>
<td>3700.0</td>
</tr>
<tr>
<td>343</td>
<td>152</td>
<td>Adelie</td>
<td>41.5</td>
<td>4000.0</td>
</tr>
</tbody>
</table>

```

```
[6]: df = df.dropna(subset=['culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm', 'body_mass_g'])

```

```
Traceback (most recent call last)
---
KeyError: ['culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm', 'body_mass_g']
```

```
[7]: df.columns

```

```
[7]: Index(['study name', 'sample number', 'species', 'region', 'sex', 'culmen length (mm)', 'culmen depth (mm)', 'flipper length (mm)', 'body mass (g)'], dtype='object')

```

```
[8]: df['body_mass_g'].max()
[8]: 6300.0

```

```
[9]: df.groupby('study name')['body_mass_g'].median()

```

```
[10]: study name

```

```
PAL0708     3900.0
PAL0809     4200.0
PAL0910     4000.0

```

```
Name: body mass (g), dtype: float64
```

```
```
Confusing Notebook

```python
import pandas as pd

df = pd.read_csv('penguins_lter.csv')

study name | sample number | species | region | culmen length (mm) | culmen depth (mm) | flipper length (mm) | body mass (g) | sex
---|---|---|---|---|---|---|---|---
PAL0708 | 1 | Adelie |  | 39.1 | 3750.0 |  |  |  
PAL0708 | 1 | Gentoo |  | 46.1 | 4500.0 |  |  |  
342 | 151 | Adelie |  | 36.0 | 3700.0 |  |  |  
343 | 152 | Adelie |  | 41.5 | 4000.0 |  |  |  

df = df.dropna(subset=['culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm', 'body_mass_g'])

KeyError Traceback (most recent call last)
Input In [6], in <cell line: 1>()
    1 df = df.dropna(subset=['culmen_length_mm', 'culmen_depth_mm',
    2 'flipper_length_mm', 'body_mass_g'])

KeyError: ['culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm', 'body_mass_g']
```

```python
df.columns

Index(['study name', 'sample number', 'species', 'region', 'culmen length (mm)', 'culmen depth (mm)', 'flipper length (mm)', 'body mass (g)'], dtype='object')

df = df.rename(columns={'culmen_length_mm': 'culmen length (mm)', 'culmen_depth_mm': 'culmen length (mm)', 'body_mass_g': 'body mass (g)'})

species | island | culmen length (mm) | culmen depth (mm) | body mass (g) | sex
---|---|---|---|---|---
Chinstrap | Dream | 50.9 | 3550.0 | MALE
Gentoo | Biscoe | 473 | 4725.0 | NaN
Chinstrap | Dream | 470 | 37000.0 | FEMALE

df['body mass (g)'].max()

6300.0

df.groupby('study name')['body mass (g)].median()

groupby( study name )

<table>
<thead>
<tr>
<th>study name</th>
<th>body mass</th>
<th>region</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAL0708</td>
<td>3900.0</td>
<td></td>
</tr>
<tr>
<td>PAL0809</td>
<td>4200.0</td>
<td></td>
</tr>
<tr>
<td>PAL0910</td>
<td>4000.0</td>
<td></td>
</tr>
</tbody>
</table>

Name: body mass (g), dtype: float64
```python
import pandas as pd

df = pd.read_csv('penguins_lter.csv')

# Grouping and calculating median of body mass by study name
result = df.groupby('study name')['body mass (g)'].median()

# Finding the maximum body mass
max_mass = df['body mass (g)'].max()

# Renaming columns for clarity
df = df.rename(columns={'culmen length_mm': 'culmen length (mm)', 'flipper length_mm': 'flipper length (mm)', 'body_mass_g': 'body mass (g)'})
```

```
<table>
<thead>
<tr>
<th>study name</th>
<th>sample number</th>
<th>species</th>
<th>culmen length (mm)</th>
<th>body mass (g)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAL0708</td>
<td>1</td>
<td>Adelie</td>
<td>39.1</td>
<td>3750.0</td>
</tr>
<tr>
<td>PAL0708</td>
<td>1</td>
<td>Gentoo</td>
<td>46.1</td>
<td>4500.0</td>
</tr>
<tr>
<td>PAL0910</td>
<td>151</td>
<td>Adelie</td>
<td>36.0</td>
<td>3700.0</td>
</tr>
<tr>
<td>PAL0910</td>
<td>152</td>
<td>Adelie</td>
<td>41.5</td>
<td>4000.0</td>
</tr>
<tr>
<td>Ch instrap</td>
<td>Dream</td>
<td></td>
<td>50.9</td>
<td>35500.0</td>
</tr>
<tr>
<td>Ch instrap</td>
<td>Dream</td>
<td>Gentoo</td>
<td>473.0</td>
<td>47250.0</td>
</tr>
<tr>
<td>Ch instrap</td>
<td>Dream</td>
<td>Gentoo</td>
<td>49.9</td>
<td>5400.0</td>
</tr>
<tr>
<td>Ch instrap</td>
<td>Dream</td>
<td>Ch instrap</td>
<td>47.0</td>
<td>37000.0</td>
</tr>
</tbody>
</table>

max_mass: 6300.0
```
Confusing Notebook

[1]:
import pandas as pd

[5]:
df = pd.read_csv('penguins_lter.csv')

[6]:
df = df.dropna(subset=['culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm', 'body_mass_g'])

[7]:
df.columns

[9]:
df['body_mass (g)'].max()

[10]:
df.groupby('study name')['body_mass (g)'].median()

Why is this an error?
Confusing Notebook

```python
[1]: import pandas as pd

[5]: df = pd.read_csv('penguins_lter.csv')

[5]:
<table>
<thead>
<tr>
<th>study name</th>
<th>sample number</th>
<th>species</th>
<th>culmen length (mm)</th>
<th>body mass (g)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAL0708</td>
<td>1</td>
<td>Adelie</td>
<td>39.1</td>
<td>3750.0</td>
</tr>
<tr>
<td>PAL0708</td>
<td>1</td>
<td>Gentoo</td>
<td>46.1</td>
<td>4500.0</td>
</tr>
<tr>
<td>PAL0708</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAL0910</td>
<td>151</td>
<td>Adelie</td>
<td>36.0</td>
<td>3700.0</td>
</tr>
<tr>
<td>PAL0910</td>
<td>152</td>
<td>Adelie</td>
<td>41.5</td>
<td>4000.0</td>
</tr>
</tbody>
</table>

[6]: df = df.dropna(subset=['culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm', 'body_mass_g'])

```

**KeyError: ['culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm', 'body_mass_g']**

```
[7]: df.columns

[7]: Index(['study name', 'sample number', 'species', 'region', ... , 'flipper_length_mm', 'culmen_length_mm', 'body_mass_g'],
        dtype='object')

[5]: df = df.rename(columns={'culmen_length_mm': 'culmen length (mm)', 'body_mass_g': 'body mass (g)'});

[5]:
<table>
<thead>
<tr>
<th>species</th>
<th>island</th>
<th>culmen length (mm)</th>
<th>body mass (g)</th>
<th>sex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chinstrap</td>
<td>Dream</td>
<td>50.9</td>
<td>3550.0</td>
<td>MALE</td>
</tr>
<tr>
<td>Gentoo</td>
<td>Biscoe</td>
<td>473</td>
<td>4725.0</td>
<td>NaN</td>
</tr>
<tr>
<td>Gentoo</td>
<td>Biscoe</td>
<td>499</td>
<td>5400.0</td>
<td>MALE</td>
</tr>
<tr>
<td>Chinstrap</td>
<td>Dream</td>
<td>47.0</td>
<td>3700.0</td>
<td>FEMALE</td>
</tr>
</tbody>
</table>

```

**Which df does this refer to?**

```
[8]: df['body mass (g)'].max()

[8]: 6300.0

[10]: df.groupby('study name')

[10]: study name
PAL0708 3900.0
PAL0809 4200.0
PAL0910 4000.0
Name: body mass (g), dtype: float64

```

Confusing Notebook

1: import pandas as pd

5: df = pd.read_csv('penguins_lter.csv')

5: | study name | sample number | species | ... | culmen length (mm) | body mass (g) |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PAL0708</td>
<td>1</td>
<td>Adelie</td>
<td></td>
<td>39.1</td>
<td>3750.0</td>
</tr>
<tr>
<td>PAL0708</td>
<td>1</td>
<td>Gentoo</td>
<td></td>
<td>46.1</td>
<td>4500.0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>PAL0910</td>
<td>151</td>
<td>Adelie</td>
<td></td>
<td>36.0</td>
<td>3700.0</td>
</tr>
<tr>
<td>PAL0910</td>
<td>152</td>
<td>Adelie</td>
<td></td>
<td>41.5</td>
<td>4000.0</td>
</tr>
</tbody>
</table>

6: df = df.dropna(subset=['culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm', 'body_mass_g'])

```
KeyError: ['culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm', 'body_mass_g']
```

```
Import In [6], in <cell line: 1>()
    1 df = df.dropna(subset=['culmen_length_mm', 'culmen_depth_mm',
    'flipper_length_mm', 'body_mass_g'])
    2

KeyError: ['culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm', 'body_mass_g']
```

7: df.columns

7: Index(["study name", 'sample number', 'species', 'region', ...

7: 'flipper length (mm)', 'culmen length (mm)', 'body mass (g)'],

dtype='object')

```
df = df.rename(columns={"culmen_length_mm": 'culmen length (mm)',
'body_mass_g': 'body mass (g)'})
```

```
species island culmen length (mm) ... body mass (g) sex
0 Chinstrap Dream 50.9 ... 35500.0 MALE
1 Gentoo Bisce 473 ... 4725.0 NaN
... ... ... ... ... ...
341 Gentoo Bisce 499 ... 5400.0 MALE
343 Chinstrap Dream 470 ... 37000.0 FEMALE
```

8: df['body mass (g)'].max()

8: 6300.0

10: df.groupby('study name')

```
study name     3900.0
PAL0708         4200.0
PAL0809         4000.0
Name: body mass (g), dtype: float64
```

Which df does this refer to?
Notebook Potholes

- If you redefine or \textit{mutate} a variable, another cell may break
- The \textit{order} of cell execution matters
- If you forget to run an edited cell, the result may \textbf{not match} the code
Improving Notebook Output and Links Between Cells

- Improve Reuse
  - Remove ambiguities
  - Enhance recall
- Improve Display
Remove Ambiguities and Preserve Recall

In [5]:
```python
import pandas as pd
df = pd.read_csv('guardian-top100-female-2019.csv')
```
Out[5]:
<table>
<thead>
<tr>
<th>Name</th>
<th>Rank</th>
<th>Position</th>
<th>Age on 1 Dec 2019</th>
<th>Nationality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam Kerr</td>
<td>1</td>
<td>Forward</td>
<td>26</td>
<td>Australia</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Ludmila</td>
<td>100</td>
<td>Forward</td>
<td>25</td>
<td>Brazil</td>
</tr>
</tbody>
</table>

100 rows x 5 columns

In [6]:
```python
df = df.rename(columns={'Age on 1 Dec 2019': 'Age'})
```
Out[6]:
<table>
<thead>
<tr>
<th>Name</th>
<th>Rank</th>
<th>Position</th>
<th>Age</th>
<th>Nationality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam Kerr</td>
<td>1</td>
<td>Forward</td>
<td>26</td>
<td>Australia</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Ludmila</td>
<td>100</td>
<td>Forward</td>
<td>25</td>
<td>Brazil</td>
</tr>
</tbody>
</table>

100 rows x 5 columns

In [3]:
```python
df = df[df.Age >= 31]
```
Out[3]:
<table>
<thead>
<tr>
<th>Name</th>
<th>Rank</th>
<th>Position</th>
<th>Age</th>
<th>Nationality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Megan Rapinoe</td>
<td>3</td>
<td>Midfielder</td>
<td>34</td>
<td>USA</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Cláudia Neto</td>
<td>97</td>
<td>Midfielder</td>
<td>31</td>
<td>Portugal</td>
</tr>
</tbody>
</table>

19 rows x 5 columns

In [7]:
```python
df = df[df.Age <= 24]
```
Out[7]:
<table>
<thead>
<tr>
<th>Name</th>
<th>Rank</th>
<th>Position</th>
<th>Age</th>
<th>Nationality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ada Hegerberg</td>
<td>4</td>
<td>Forward</td>
<td>24</td>
<td>Norway</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Lena Oberdorf</td>
<td>99</td>
<td>Midfielder</td>
<td>17</td>
<td>Germany</td>
</tr>
</tbody>
</table>

25 rows x 5 columns
Remove Ambiguities and Preserve Recall

In [d51f8eab]:
```python
import pandas as pd
df = pd.read_csv('guardian-top100-female-2019.csv')
```

```
Name  Rank  Position    Age on 1 Dec 2019  Nationality
0  Sam Kerr      1  Forward            26  Australia
... ...         ...                  ...  ...
99  Ludmila      100  Forward           25  Brazil
```

100 rows x 5 columns

In [full]:
```python
df = df.rename(columns={'Age on 1 Dec 2019': 'Age'})
```

```
Name  Rank  Position  Age  Nationality
0  Sam Kerr      1  Forward      26  Australia
... ...         ...         ...  ...
99  Ludmila      100  Forward     25  Brazil
```

100 rows x 5 columns

In [over30]:
```python
df = df[df.Age >= 31]
```

```
Name  Rank  Position  Age  Nationality
2  Megan Rapinoe  3  Midfielder    34  USA
96  Cláudia Neto  97  Midfielder   31  Portugal
```

19 rows x 5 columns

In [under25]:
```python
df = df[df.Age <= 24]
```

```
Name  Rank  Position  Age  Nationality
3  Ada Hegerberg  4  Forward      24  Norway
98  Lena Oberdorf 99  Midfielder   17  Germany
```

25 rows x 5 columns

[dataflownb.github.io]
Improve Output Representations

```python
{'setosa': (    sepal_length  sepal_width  petal_length  petal_width
13           4.3          3.0           1.1          0.1,
<IPython.core.display.Image object>),
'versicolor': (    sepal_length  sepal_width  petal_length  petal_width
99           5.7          2.8           4.1          1.3
67           5.8          2.7           4.1          1.0,
<IPython.core.display.Image object>),
'virginica': (     sepal_length  sepal_width  petal_length  petal_width
104           6.5          3.0           5.8          2.2
121           5.6          2.8           4.9          2.0
116           6.5          3.0           5.5          1.8,
<IPython.core.display.Image object>))
```
## Improve Output Representations

\[
\begin{array}{|c|c|c|c|c|}
\hline
& sepal_length & sepal_width & petal_length & petal_width \\
\hline
104 & 6.5 & 3.0 & 5.8 & 2.2 \\
121 & 5.6 & 2.8 & 4.9 & 2.0 \\
116 & 6.5 & 3.0 & 5.5 & 1.8 \\
\hline
\end{array}
\]

[link to dataflownb.github.io]
Computational notebooks facilitate efficient exploration because users can quickly inspect and reuse intermediate outputs.

It is important to show interactive output that summarizes while providing the ability to dig into details.

It is important to be able to recall and reuse past outputs during and after analysis without re-running or re-writing code.

JupyterLab and IPython extensions to improve notebooks:
- Recall and reuse of past outputs (dfnotebook-extension + dfkernel)
- Output display (ipycollections)
Visualization Landscape

- **Apps:** ParaView, VisIt
- **Domain-Specific Apps:** VMD, VAPOR
- **APIs:** VTK, ITK
- **Also… Data Analysis Tools/Libraries**
  - JavaScript: D3, Observable Plot
  - R: ggplot
  - **Python:** matplotlib, altair, bokeh, …
  - Matlab
  - GNUPlot
The Python Visualization Landscape

[Image of a diagram showing various Python visualization libraries, including `matplotlib`, `pandas`, `seaborn`, `networkx`, `bokeh`, `vispy`, `Mayavi`, and more. The diagram is color-coded and uses arrows to connect different libraries, indicating relationships or dependencies.]
The Python Visualization Landscape

[Diagram showing various visualization libraries and tools connected to each other.]

[Text: The Python Visualization Landscape by J. VanderPlas, adapted by N. Rougier]
Examples

- Examine airfoil data on Cooley
- Login to:
  - jupyter.alcf.anl.gov
- Click on "Login Cooley"
- Copy .ipynb files from Track 4 Examples dir to your $HOME: /grand/projects/ATPESC2023/EXAMPLES/track-4-visualization
- ...or use upload button to upload the two airfoil notebooks
  - airfoil-flow.ipynb
  - airfoil-line-plots.ipynb
- Once uploaded, click on the flow notebook to start
**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

```python
plt.hist(...)
```
Anatomy of a Figure

- **Figure**
  - **Axes**
    - **Title**
  - **Y label**: 1.0, 0.5, 0.0, -0.5, -1.0
  - **X label**: 0, 1, 2, 3, 4, 5, 6, 7

Anatomy of a figure:
- **Major tick**
- **Minor tick**
- **Title**
- **Line** (line plot)
- **Grid**
- **Markers** (scatter plot)
- **Spines**
- **Legend**

Figure included in the slide. Made with https://matplotlib.org
Altair

- Declarative Visualization
  - Specify **what** instead of how
  - Separate specification from execution
- Based on VegaLite which is browser-based
- Strengths:
  - Declarative visualization
  - Web technologies
- Weaknesses:
  - Scaling (but see VegaFusion)
  - Specifications + translate to JavaScript
Data Items Become 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

[T. Munzner, E. Maguire (ill.)]
Encode Attributes via Visual Channels

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

- **Color**

- **Shape**

- **Tilt**

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

[T. Munzner, E. Maguire (ill.)]
Examples

• Examine airfoil data on Cooley
• Login to:
  - jupyter.alcf.anl.gov
• Click on "Login Cooley"
• Copy .ipynb files from Track 4 Examples dir to your $HOME: /grand/projects/ATPESC2023/EXAMPLES/track-4-visualization
• …or use upload button to upload the two airfoil notebooks
  - airfoil-flow.ipynb
  - airfoil-line-plots.ipynb
• Once uploaded, click on the flow notebook to start
Questions?

```python
plt.imshow(twod_data_array, cmap=cmap, extent=[-2.5,2.5,-2.5,2.5])
plt.clim(-1.0, 0.5); # for p
plt.title('MFEM Simulation of Potential Flow Around an Airfoil');
plt.ylabel('Height of domain');
plt.xlabel('Length of domain');
cb = plt.colorbar(extend='both');
cb.set_label('Pressure', rotation=270, labelpad=24)

chart = alt.Chart(df).encode(x='step')
area = chart.mark_area(opacity=0.4).encode(y='max', y2='min')
line = chart.mark_line(color='blue').encode(y='max')
line2 = chart.mark_line(color='orange').encode(y='mean')
line3 = chart.mark_line(color='red').encode(y='min')
area + line + line2 + line3
```