Data Visualization in Notebooks

Dr. David Koop
Northern Illinois University
"The purpose of visualization is insight, not pictures"

– B. Schneiderman
Notebooks

```python
In [2]:
from lorenz import solve_lorenz

w Interactive(solve_lorenz, sigma=(0.0, 0.0), rho=(0.0, 0.0))

We explore the Lorenz system of differential equations:

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

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

In [2]:
def solve_lorenz(sigma=10.0, beta=8.0/3, rho=28.0):
    """Plot a solution to the lorenz differential equations."
    max_time = 4.0
    N = 20
    fig = plt.figure()
    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
    ax.axis('off')

    # prepare the axes limits
    ax.set_xlim((-25, 25))
    ax.set_ylim((-35, 35))
    ax.set_zlim((0, 55))

    def lorenz_derivative(x, y, z, t, sigma, beta, rho, rho):
        """Compute the time-derivative of a Lorenz system."
        x, y, z = x, y, z
        return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]

    # Choose random starting points, uniformly distributed from -15 to 15
    np.random.seed()
    x0 = (-15 + 30 * np.random.randn(N, 3))

    # Solve for the trajectories
    t = np.linspace(0, max_time, int(250 * max_time))
    x0_t = np.array([integrate.odeint(lorenz_derivative, x0[i], t) for x0 in x0])

    # Choose a different color for each trajectory
    colors = plt.cm.viridis(np.linspace(0, 1, N))

    for i in range(N):
        x, y, z = x0_t[i, :].T
        lines = ax.plot(x, y, z, '-', color=colors[i])
        plt.setp(lines, linewidth=2)
        angle = 184
        ax.view_init(30, angle)
```

[JupyterLab]
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.

```
[1]: %matplotlib inline
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.

```
[2]: 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
Two Kinds of Notebooks

• Different notebooks for different stages of work
  - Exploratory notebooks
  - Explanatory notebooks
• Parallels a general approach to research
  - Brainstorm questions and explore answers
  - Distill initial results into something understandable and verifiable
  - Publish polished text and (interactive) visualizations
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
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
```python
[291]:
  import pandas as pd
  from dbanalysis import stop_tools
  data = stop_tools.random_stop_data()

[292]:
  data['weekend']=data['day']>=4
  data['peak1']=data['hour'] >= 7 & (data['hour'] <= 18)
  data['peak2']=data['hour'] >= 16 & (data['hour'] < 19)
  data['hour'] = data['hour'].astype('category')
  data['day'] = data['day'].astype('category')
  data['month'] = data['month'].astype('category')
  data = data[data['traveltime'] >= 0]

[294]:
  data.columns

# Index(['index', 'dayofservice', 'tripid', 'plannedtime_arr_from',
# 'plannedtime_dep_from', 'actualtime_arr_from', 'actualtime_dep_from',
# 'plannedtime_arr_to', 'actualtime_arr_to', 'routeid', 'fromstop',
# 'tostop', 'traveltime', 'dwelltime', 'distance', 'speed', 'dt', 'date',
# 'year', 'dewpt', 'mst', 'rain', 'rhum', 'temp', 'vappn', 'webt',
# 'weekday', 'peak1', 'peak2', 'hour_5', 'hour_6', 'hour_7', 'hour_8',
# 'hour_9', 'hour_10', 'hour_11', 'hour_12', 'hour_13', 'hour_14',
# 'hour_15', 'hour_16', 'hour_17', 'hour_18', 'hour_19', 'hour_20',
# 'hour_21', 'hour_22', 'hour_23', 'day_0', 'day_1', 'day_2', 'day_3',
# 'day_4', 'day_5', 'day_6', 'month_1', 'month_2', 'month_3', 'month_4',
# 'month_5', 'month_6'],
# dtype='object')

[295]:
  data['weekend']=data['day']>=4

---

```

---

```

```

---

```
Explanatory Notebooks

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]:
```python
net.load_file('~/lung_cellline_3_1_16/lung_cl_all_ptm/precalc_processed/CCLE_exp.txt')
print('Expression data shape: ' + str(net.dat['mat'].shape))
```
Expression data shape: (1000, 37)

In [5]:
```python
net.set_cat_color('row', 1, 'Data-Type: Exp', 'yellow')
net.cluster(view=[])
net.widget()
```

[David Koop]

N. Fernandez et al.

ATPESC 2020
These nice explanations often take a lot of work!
Reproducibility of Notebooks

- Environment: have installed correct versions of libraries/packages
- Data: have access to all of the necessary data
- Execution: able to rerun the cells as intended
Exploration Potholes

- If you redefine or **mutate** a variable, another cell may break
- The **order** of cell execution matters
- If you forget to run an edited cell, the result may **not match** the code
In [a0a358]:
```python
raw_df = pd.read_csv("fifa17-top20-women.txt", sep="-", header=None)
```

```
<table>
<thead>
<tr>
<th></th>
<th>Name</th>
<th>Country</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Caroline Seger</td>
<td>Sweden</td>
<td>85</td>
</tr>
<tr>
<td>1</td>
<td>Wendie Renard</td>
<td>France</td>
<td>85</td>
</tr>
<tr>
<td>2</td>
<td>Steph Houghton</td>
<td>England</td>
<td>85</td>
</tr>
</tbody>
</table>
```

In [aaa3c6]:
```
column_names = {0: "Name", 1: "Country", 2: "Rating"}
column_names = {0: 'Name', 1: 'Country', 2: 'Rating'}
```

In [a249ea]:
```
named_df = raw_df.rename(columns=column_names)
```

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Country</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caroline Seger</td>
<td>Sweden</td>
<td>85</td>
</tr>
<tr>
<td>Wendie Renard</td>
<td>France</td>
<td>85</td>
</tr>
<tr>
<td>Steph Houghton</td>
<td>England</td>
<td>85</td>
</tr>
</tbody>
</table>
```

In [aab079]:
```
named_df.groupby("Country").size().sort_values(ascending=False)
```

```
Country
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>USA 6</td>
</tr>
<tr>
<td>Canada 3</td>
</tr>
<tr>
<td>Brazil 3</td>
</tr>
</tbody>
</table>
```

[Dataflow Notebooks](https://dataflownb.github.io)
In [a0a358]:

```python
raw_df = pd.read_csv("fifa17-top20-women.txt", sep="-", header=None)
```

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Caroline Seger</td>
<td>Sweden</td>
<td>85</td>
</tr>
<tr>
<td>1</td>
<td>Wendie Renard</td>
<td>France</td>
<td>85</td>
</tr>
<tr>
<td>2</td>
<td>Steph Houghton</td>
<td>England</td>
<td>85</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

In [aaa3c6]:

```python
column_names = {0: "Name", 1: "Country", 2: "Rating"}
```

column_names: {0: 'Name', 1: 'Country', 2: 'Rating'}

In [a249ea]:

```python
named_df = raw_df.rename(columns=column_names)
```

<table>
<thead>
<tr>
<th></th>
<th>Name</th>
<th>Country</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Caroline Seger</td>
<td>Sweden</td>
<td>85</td>
</tr>
<tr>
<td>1</td>
<td>Wendie Renard</td>
<td>France</td>
<td>85</td>
</tr>
<tr>
<td>2</td>
<td>Steph Houghton</td>
<td>England</td>
<td>85</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

In [aab079]:

```python
named_df.groupby("Country").size().sort_values(ascending=False)
```

Out[aab079]:

<table>
<thead>
<tr>
<th></th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>USA</td>
</tr>
<tr>
<td>3</td>
<td>Canada</td>
</tr>
<tr>
<td>3</td>
<td>Brazil</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
### Dataflow Notebooks

- **Persistent Identifiers**
- **Named Outputs**

```python
In [a0a358]:
    raw_df = pd.read_csv("fifa17-top20-women.txt", sep="-", header=None)

raw_df:

<table>
<thead>
<tr>
<th></th>
<th>Name</th>
<th>Country</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Caroline Seger</td>
<td>Sweden</td>
<td>85</td>
</tr>
<tr>
<td>1</td>
<td>Wendie Renard</td>
<td>France</td>
<td>85</td>
</tr>
<tr>
<td>2</td>
<td>Steph Houghton</td>
<td>England</td>
<td>85</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

In [aaa3c6]:
    column_names = {0: "Name", 1: "Country", 2: "Rating"}

column_names:

{0: 'Name', 1: 'Country', 2: 'Rating'}

In [a249ea]:
    named_df = raw_df.rename(columns=column_names)

named_df:

<table>
<thead>
<tr>
<th>Name</th>
<th>Country</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caroline Seger</td>
<td>Sweden</td>
<td>85</td>
</tr>
<tr>
<td>Wendie Renard</td>
<td>France</td>
<td>85</td>
</tr>
<tr>
<td>Steph Houghton</td>
<td>England</td>
<td>85</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

In [aab079]:
    named_df.groupby("Country").size().sort_values(ascending=False)

Out[aab079]:

<table>
<thead>
<tr>
<th>Country</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>6</td>
</tr>
<tr>
<td>Canada</td>
<td>3</td>
</tr>
<tr>
<td>Brazil</td>
<td>3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
```
Dataflow Notebooks

- Persistent Identifiers
- Named Outputs
- Unnamed Outputs

```python
In [a0a358]:
raw_df = pd.read_csv("fifa17-top20-women.txt", sep="-", header=None)

In [aaa3c6]:
column_names = {0: "Name", 1: "Country", 2: "Rating"}

column_names: {0: 'Name', 1: 'Country', 2: 'Rating'}

In [a249ea]:
named_df = raw_df.rename(columns=column_names)

In [aab079]:
named_df.groupby("Country").size().sort_values(ascending=False)

Out[aab079]:
Country
USA       6
Canada    3
Brazil    3
...      ...  
```

Dataflow Notebooks
• Persistent Identifiers
• Named Outputs
• Unnamed Outputs
Dataflow Notebooks

- Persistent Identifiers
- Named Outputs
- Unnamed Outputs
- Connection by Variable Reference

```
In [a0a358]: raw_df = pd.read_csv("fifa17-top20-women.txt", sep="-", header=None)

raw_df:
     0      1      2
0  Caroline Seger  Sweden   85
1    Wendie Renard  France   85
2   Steph Houghton  England   85
...        ...        ...

In [aaa3c6]: column_names = {0: "Name", 1: "Country", 2: "Rating"}
column_names: {0: 'Name', 1: 'Country', 2: 'Rating'}

In [a249ea]: named_df = raw_df.rename(columns=column_names)
named_df:
         Name    Country   Rating
0  Caroline Seger    Sweden    85
1    Wendie Renard    France    85
2   Steph Houghton    England    85
...        ...        ...

In [aab079]: named_df.groupby("Country").size().sort_values(ascending=False)

Out[aab079]:
Country
   USA     6
  Canada   3
  Brazil   3
...  ...
The Python Visualization Landscape

[C. VanderPlas, adapted by N. Rougier]
The Python Visualization Landscape

[Image of a network diagram showing various visualization libraries connected to each other, such as `matplotlib`, `networkx`, `pythreejs`, and `plotly`.]
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
Examples

• Examine airfoil data on Cooley
• Login to:
  - jupyter.alcf.anl.gov
• Click on "Login Cooley"
• 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