



# AI Enterprise Workflow Study Group

Course 2, Week 1

3/7/2020

# Agenda

- Check in
- C2W1 key concepts
- Discussion
- Next steps

# Course & Study Group Schedule

AI Enterprise Workflow Study Group		
Session	Topic	Date
Overview Webinar	Webinar with instructor, Ray Lopez	15-Feb
Course 1 Week 1	Course intro	22-Feb
Course 1 Week 2	Data ingestion, cleaning, parsing, assembly	29-Feb
Course 2 Week 1	Exploratory data analysis & visualization	7-Mar
Course 2 Week 2	Estimation and NHT	14-Mar
Course 3 Week 1	Data transformation and feature engineering	21-Mar
Course 3 Week 2	Pattern recognition and data mining best practices	28-Mar
Course 4 Week 1	Model evaluation and performance metrics	4-Apr
Course 4 Week 2	Building machine learning and deep learning models	11-Apr
Course 5 Week 1	Deploying models	18-Apr
Course 5 Week 2	Deploying models using Spark	25-Apr
Course 6 Week 1	Feedback loops and monitoring	2-May
Course 6 Week 2	Hands on with OpenScale and Kubernetes	9-May
Course 6 Week 3	Captstone project week 1	16-May
Course 6 Week 4	Captstone project week 2	23-May

# Course 2 Week 1 learning objectives

1. Explain the principal steps in exploratory data analysis
2. Explain the use case for Python tools (pandas, matplotlib, and Jupyter) in EDA
3. Describe strategies for dealing with missing data
4. Explain the role of communication in EDA

# Key concepts

- Exploratory Data Analysis
  - EDA Process
  - Tools
  - Summarizing Data
- Missing Data
  - Types
  - Ways to Deal With
  - Imputation Strategies

# Exploratory Data Analysis

- Exploratory vs Confirmatory
- Sometimes thoughtful EDA & visualization can reveal solution to problem such that models aren't required.
- Challenge: Many options in how data can be visualized and results communicated

# EDA Process



# EDA Process Example

- Load data into pandas, NumPy or another similar tool and summarize the data
- Use tables, text and visualizations to tell the story that relates the business opportunity to the data
- Identify a strategy to deal with missing values
- Investigate the data and underlying business scenario with visualizations and hypothesis testing
- Communicate your findings



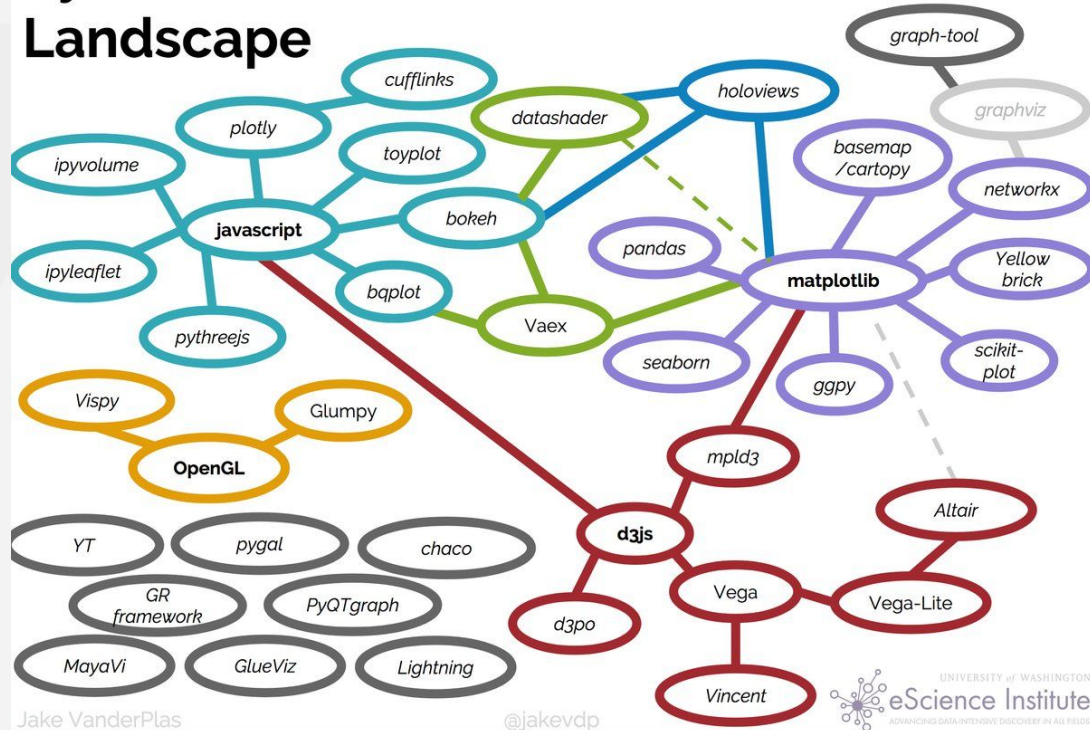
# Communication

Not just results, also:

- What you have done
- What you are doing
- What you plan to do

# Tools

## Python's Visualization Landscape



<https://www.anaconda.com/python-data-visualization-2018-why-so-many-libraries/>

## twiml

# Python's Visualization Landscape

The diagram illustrates the Python visualization landscape, showing connections between various libraries. The central hub is **matplotlib**, which is highlighted by a red box. It is connected to **pandas**, **seaborn**, **ggpy**, **basemap/cartopy**, **networkx**, **Yellow brick**, **scikit-plot**, and **Altair**. **Altair** is further connected to **Vega** and **Vega-Lite**, which are connected to **Vincent**. **matplotlib** is also connected to **mpld3**, which is connected to **d3js**. **d3js** is connected to **d3po** and **Vega**. **matplotlib** is connected to **nodeviews**, which is connected to **graph-tool** and **graphviz**. **matplotlib** is connected to **datareader**, which is connected to **bokeh** and **Vaex**. **matplotlib** is connected to **pythreejs**, which is connected to **javascript**. **matplotlib** is connected to **toyplot**, which is connected to **bokeh**. **matplotlib** is connected to **cufflinks**, which is connected to **bokeh**. **matplotlib** is connected to **ipyvolume**, which is connected to **javascript**. **matplotlib** is connected to **ipyleaflet**, which is connected to **javascript**. **matplotlib** is connected to **Vispy**, which is connected to **OpenGL**. **matplotlib** is connected to **Glumpy**. **matplotlib** is connected to **YT**, **pygal**, **chaco**, **GR framework**, **PyQTgraph**, **MayaVi**, **Glueviz**, and **Lightning**. **matplotlib** is connected to **OpenGL**. **matplotlib** is connected to **Vispy**. **matplotlib** is connected to **Glumpy**. **matplotlib** is connected to **YT**. **matplotlib** is connected to **pygal**. **matplotlib** is connected to **chaco**. **matplotlib** is connected to **GR framework**. **matplotlib** is connected to **PyQTgraph**. **matplotlib** is connected to **MayaVi**. **matplotlib** is connected to **Glueviz**. **matplotlib** is connected to **Lightning**.

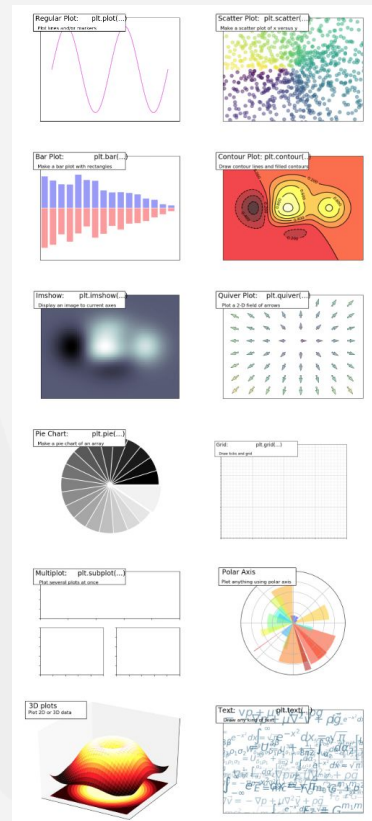
Jake VanderPlas  
@jakevdp

UNIVERSITY of WASHINGTON  
eScience Institute  
ADVANCING DATA-INTENSIVE DISCOVERY IN ALL FIELDS

<https://www.anaconda.com/python-data-visualization-2018-why-so-many-libraries/>

# Tips

- Keep plots quick and dirty
- Emphasis is identifying and communicating insights
- Not meant to be long term dashboards
- Keep codebase separate from your visualization notebook. Code should exist as separate files, in part to ensure version control and collaboration.
- Ok to get through projects via copy/paste, but very helpful to dig into the docs to see what's possible and understand how all the pieces fit together.



# Summarizing Data

- Use `groupby` and `pivot_table` to summarize data
- Discretization for continuous data: `qcut()` & `cut()`

# Types of Missing Data

- Missing Completely at Random (MCAR)
- Missing at Random (MAR)
  - Some dependence on missing value e.g gender bias
  - Models can be used to impute missing values
- Missing Not at Random (MNAR)
  - Missing data depend on unmeasured or unknown variables

# Dealing with Missing Data

- Ignore
- Complete Case Analysis
  - Impact depends on category of missingness
  - Rows: `df.dropna()`
  - Columns: `df.dropna(axis='columns')` // feature engineering
- Imputation
- Empathize - Try to get at the root of what's causing missing values

# Imputation

Imputation: Replace missing data with substituted values

- Bayesian Imputation
- Multiple imputation

Choose strategy based on type of data:

- Categorical Data: Treat missing data as a category in categorical data.
- Numerical Data: Better to use imputation technique. Can track w new column.

Imputation strategies decided by evaluating models on hold-out set

“reconsider your imputation methods once you get to the back-and-forth between the feature engineering and modeling phases of the project”



The logo for Twiml, featuring the word "twiml" in a white, lowercase, sans-serif font. A small blue horizontal bar is positioned above the "i".

twiml