

Data Acquisition

Get started with a workflow,
read data from various sources

Yordan Darakchiev

Technical Trainer

iordan93@gmail.com





sli.do

#DataScience

Table of Contents

- The scientific method
- Setting up the environment
- Reading data from different sources
 - Text files
 - Excel
 - Web services
 - SQL databases
- Data consolidation principles

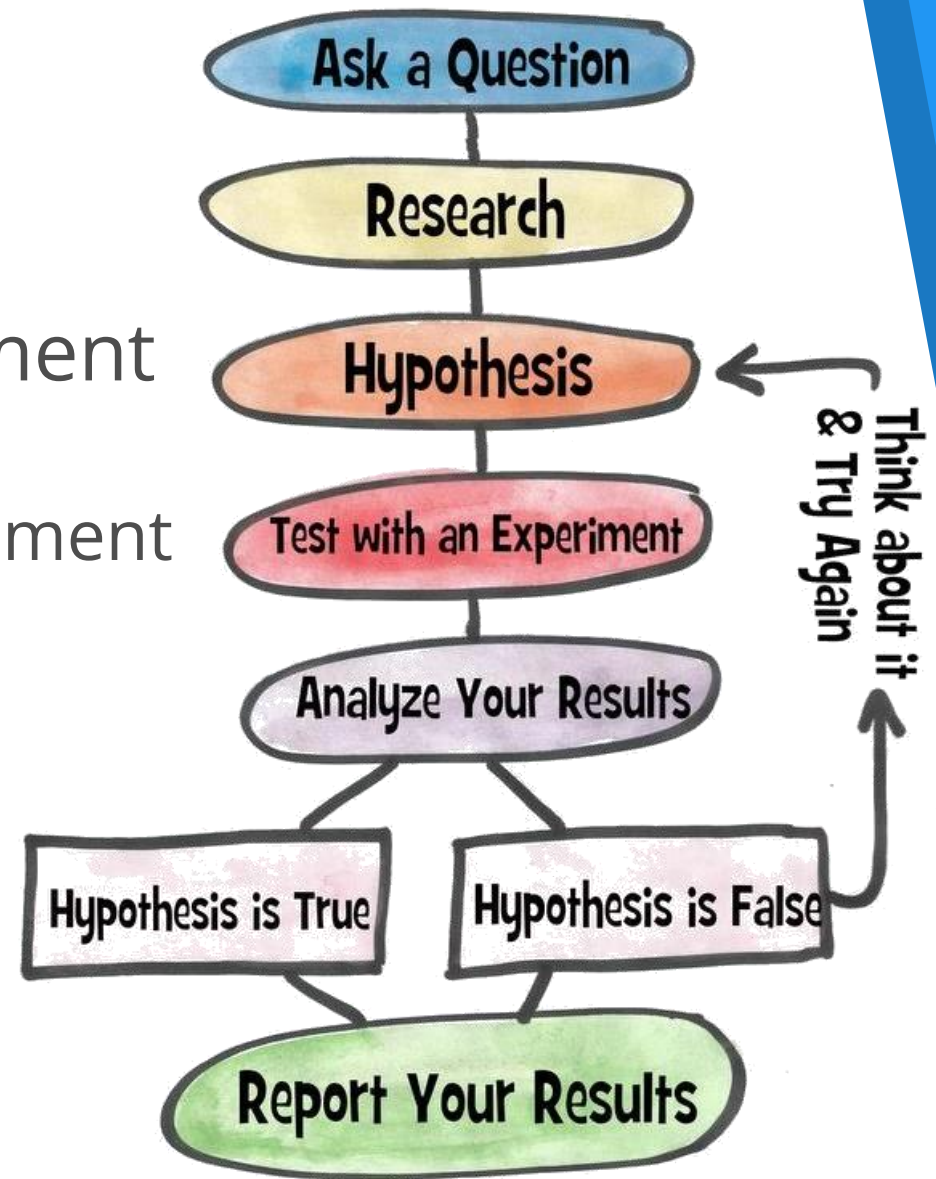


The Scientific Method

How not to get lost:
a quick reminder

The Scientific Method Steps

- Ask a question
- Do some research
- Form a hypothesis
- Test the hypothesis with an experiment
 - Experiment works \Rightarrow Analyze the data
 - Experiment doesn't work \Rightarrow Fix experiment
- Results align with hypothesis \Rightarrow OK
- Results don't align with hypothesis \Rightarrow new question, new hypothesis
- Communicate the results





Getting Data

Reading data from various sources

The pandas Library

- Provides a way to read and work with data
 - **Table** (DataFrame)
 - May have many dimensions
 - We usually call this a "dataset"
 - **List** (Series)
 - One-dimensional
 - Usually represents a column of a table
- Usage

```
import pandas as pd
```
- General requirements
 - Rows and columns are indexed, columns may have names
 - **Each column has a fixed data type**
 - Python will try to infer the best type according to the data

Data Sources

- In order to work with the data, we need to represent it in tabular form
 - Sometimes our data is tabular – we just need to read it
 - In other cases, we need to create our tables
 - **Unstructured data:** data that doesn't have a **model**
 - There is some structure, it's just not very clear
 - Examples: Images, plain text, audio, web pages
- Most common sources
 - Tables in a text format such as .csv
 - Spreadsheets (such as Excel or Google Sheets)
 - Web services
 - Databases

Reading a Local File

- Let's read the file `accidents.csv`
 - Copy the file to a data folder
 - Not required, just makes working with many data files easier
 - Inspect the file (use a text editor or Excel) just to see what it contains

```
accidents_data = pd.read_csv("data/accidents.csv")
```

- `read_csv()` [docs](#)
- You'll see that all `read_*`() functions have a lot of optional arguments
 - They make working with different formats easy, e.g.
 - Instead of `True` and `False`, the table contains "Yes" and "No"
 - The actual table starts at line 30 of the file
 - There are blank / comment lines which should be skipped
 - There are no column names in the file

Exploring the Dataset

- In Python, we can print the variable

```
print(accidents_data)
```

- Even better, in Jupyter, a cell outputs its last returned value

- This will create a nicer output

```
accidents_data
```

- We can see that

- Rows have numerical indices starting at 0 by default
- Columns have names taken from the first line in the .csv file

- Column names: `accidents_data.columns`

- Index values: `accidents_data.index`

- Dimensions: `accidents_data.shape`

- Format: (rows, columns)

Reading Data from Other Files

- The process is very similar
- Other text-based formats
 - `pd.read_table()` is the most general function
 - All others (`read_csv()`, `read_fwf()`, etc.) just apply some settings
 - If we come across a file, we can apply our own settings
 - The point is to match the format in the best possible way
 - Example: [AutoMPG dataset](#)
- Excel
 - Read the `green_tripdata_2015-09.xls` file using `pd.read_excel()`
 - Explore the file dimensions

Reading Data from Web Services

- Web services work over the HTTP protocol and provide data in several formats
 - Most commonly used: JSON and XML
 - [Some APIs to try](#)
- Example: [OpenLibrary API](#)
 - We want information about books with ISBNs
 - Example: [these 4 books](#)
 - We can put the URL directly, pandas will perform a GET request
 - Function: `pd.read_json()`
 - We can provide the parameter `orient = "index"` to arrange the dataset better
 - Books should be placed by rows, their properties – by columns
 - More details on this – next time
 - More complex queries require more pre-processing

Reading Data from SQL

- Relational databases store data in tables
 - Very similar to the datasets we use
- First, install a library to connect to databases
 - From the command line: `conda install SQLAlchemy`
- Then, import the library and connect to the database
 - **Note:** This is going to vary depending on your server settings

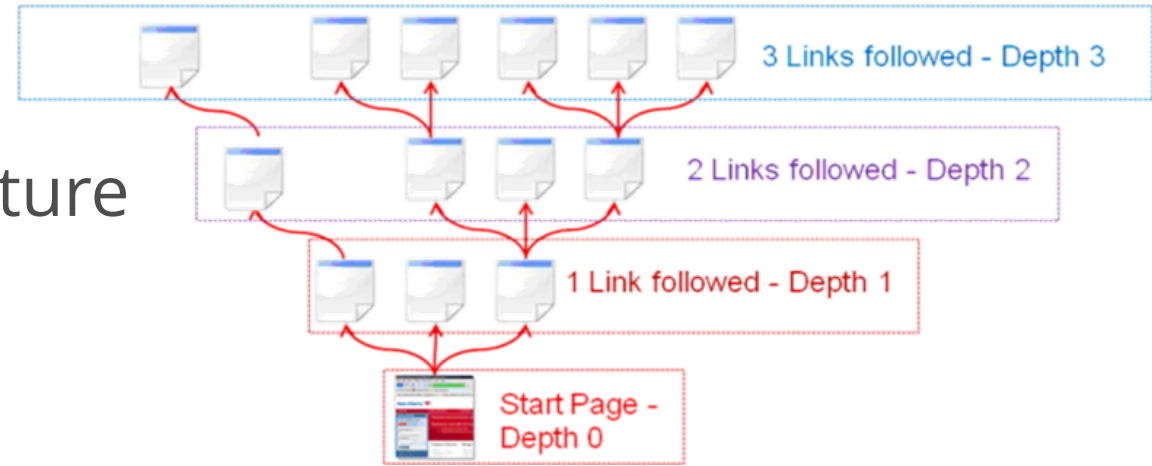
```
import sqlalchemy  
engine = sqlalchemy.create_engine("...")
```

- Perform a query

```
customer_info = pd.read_sql(  
    "select * from Sales.Customer",  
    engine)
```

Web Scrapping

- Another method for getting data
- Sometimes combined with **crawling**
 - Traversing a Web page structure recursively
- Basic procedure
 - Read a Web page as HTML
 - Use the HTML to obtain the data
 - A webpage is unstructured
 - We need to create and maintain the structure
 - We usually need more libraries to do that
- Examples
 - Get all job listings from a website
 - Get user contact details from a Web page





Using Multiple Sources

Constraints and Validity

Data Guidelines

- Some queries will not be simple
 - E.g., scraping, dealing with "freeform" text, audio data, networks
 - We need to create a tabular structure from the raw data
 - How? We'll discuss this later in the course
- After we read the data, we have to ensure it's been read without errors
 - A very simple first check: check the dimensions (`dataframe.shape`) and show the first few rows (`dataframe.head()`)
 - We may need to rename columns
 - We may need to perform different manipulations to ensure the data is in a proper state
 - We'll do this in the next lectures

Merging Many Data Sources

- **Automate the process** as much as possible
 - From reading the raw data to getting the processed dataset
 - If the dataset changes or updates, you'll just re-run your code
- **Document the process**
- Create as few datasets as possible
 - I.e., merge many sources into one table if you can
 - We'll talk more about combining relations next time
- Ensure the different sources are compatible and consistent
 - If they aren't, process the raw data
 - Most common example: Mismatched IDs
- Make sure all column types are correct
 - Check: `dataframe.dtypes`
 - Example: `str` type for a numeric column

Summary

- Methods
 - Divide and conquer
 - Scientific method
- Setting up the environment
- Reading data from different sources
 - Text files
 - Excel
 - Web services
 - SQL databases
- Data consolidation principles

The image features a white background with two blue decorative bars. The top bar is a solid blue band at the very top. Below it is a thinner, darker blue band that curves slightly downwards from left to right. The bottom of the image is framed by a similar pattern of a thin dark blue band above a solid blue band, both curving upwards from left to right.

Questions?