Introduction to *Urban* Data Science

Data Grammar

(EPA1316A)

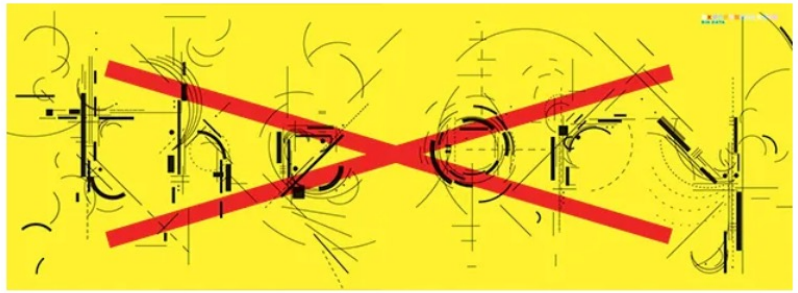
Lecture 3

Trivik Verma

Correlation supersedes causation, and science can advance even without coherent models, unified theories, or really any mechanistic explanation at all.

The End of Theory: The Data Deluge Makes the Scientific Method Obsolete

Illustration: Marian Bantjes "All models are wrong, but some are useful." So proclaimed statistician George Box 30 years ago, and he was right. But what choice did we have? Only models, from cosmological equations to theories of human behavior, seemed to be able to consistently, if imperfectly, explain the world around us. Until now. Today companies [...]

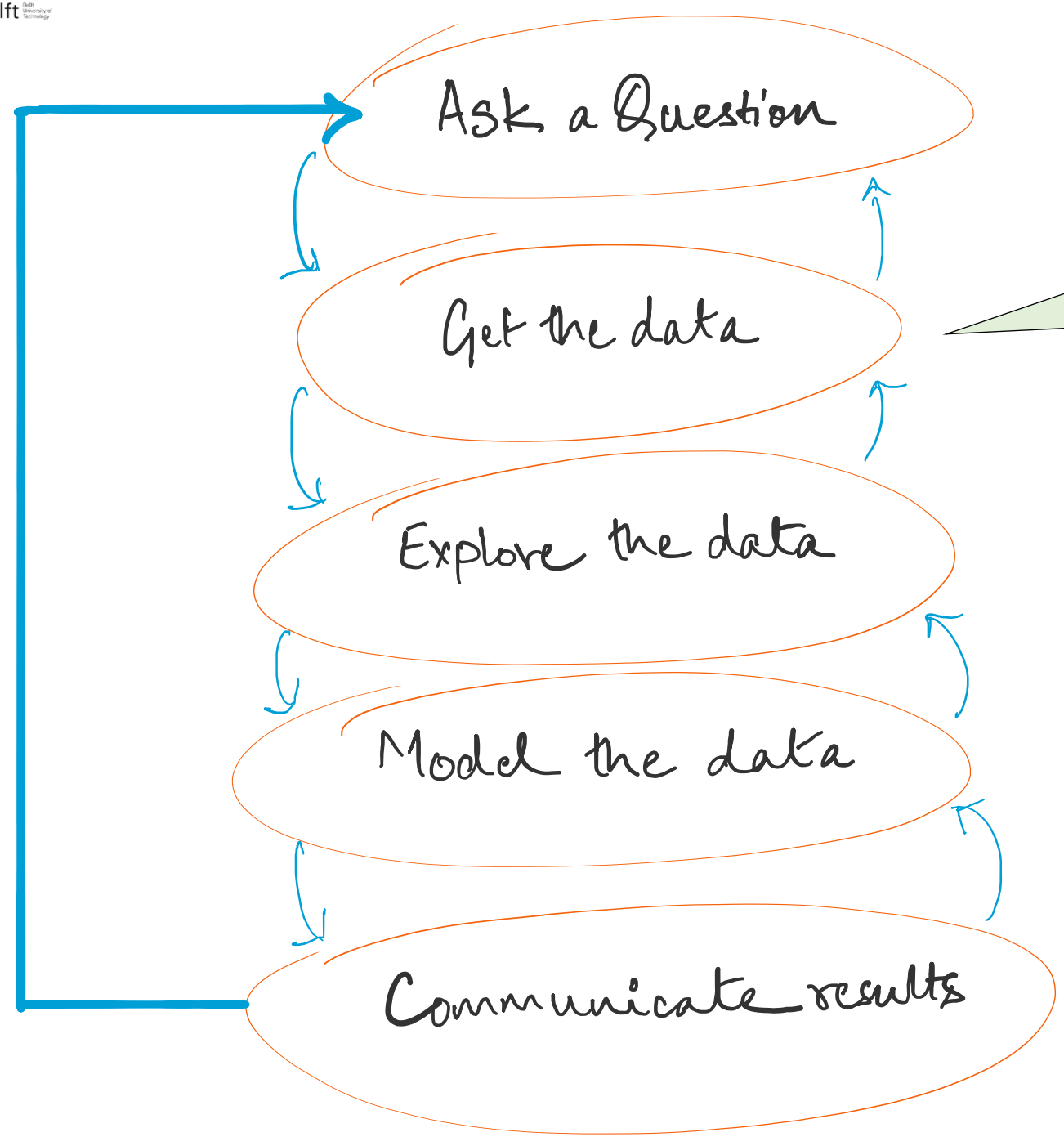


Last Time

- The Data Science Process
- Examine the **role** of evidence in policy
- Analyse **data** understanding and preparation **requirements**
- What are data?
- Types of (geo-)data
- Traditional and new sources of spatial data
- Opportunities and Challenges
- New ways for traditional approaches

Today

- Types of Data
- Grammar
- EDA without Pandas
- EDA with Pandas
- Data Concerns



How were the data **sampled**?
Which data are **relevant**?
Are there **privacy** issues?

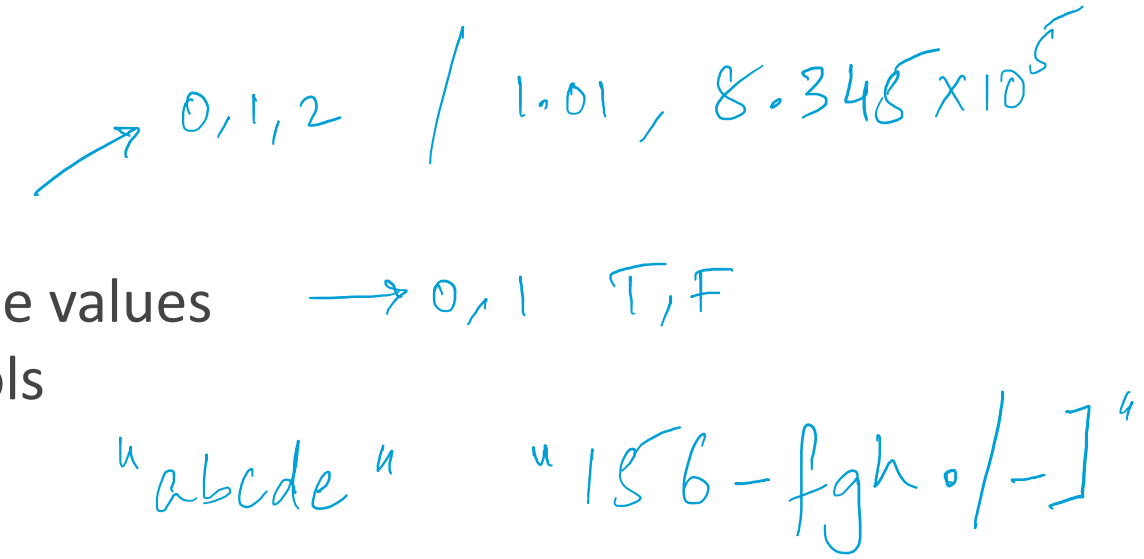
Types of Data

Types of Data

What kind of values are in your data (data types)?

Simple or atomic:

Numeric: integers, floats
Boolean: binary or true false values
Strings: sequence of symbols



Data Types

What kind of values are in your data (data types)? Compound, composed of a bunch of atomic types:

Date and time: compound value with a specific structure

Lists: a list is a sequence of values

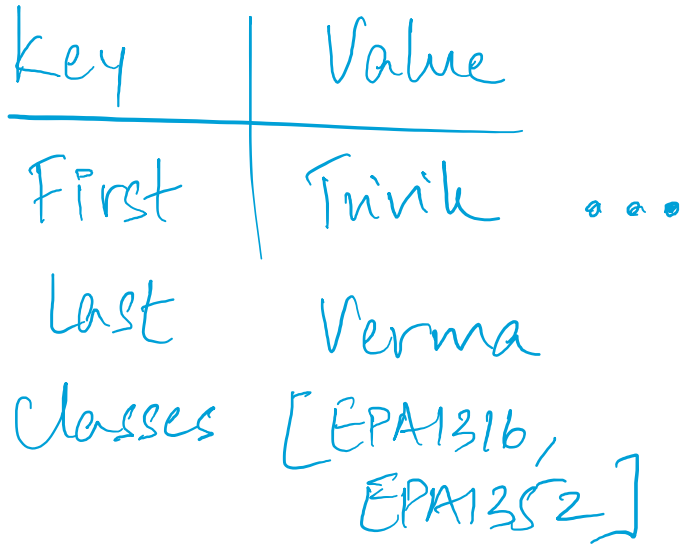
Dictionaries: A dictionary is a collection of key-value pairs, a pair of values *x* : *y* where *x* is usually a string called the key representing the "name" of the entry, and *y* is a value of any type.

Example: Student record: what are *x* and *y*?

First: Trivik

Last: Verma

Classes: [EPA1316, EPA1352]



Data Storage

How is your data represented and stored (data format)?

- **Tabular Data:** a dataset that is a two-dimensional table, where each row typically represents a single data record, and each column represents one type of measurement (csv, dat, xlsx, etc.).
- **Structured Data:** each data record is presented in a form of a [possibly complex and multi-tiered] dictionary (json, xml, etc.)
- **Semi structured Data:** not all records are represented by the same set of keys or some data records are not represented using the key-value pair structure.
- **Geographic Data:** (shp, dbx, shx…)

Tabular Data

In tabular data, we expect each record or observation to represent a set of measurements of a single object or event.

First Look At The Data

```
In [27]: hubway_data = pd.read_csv('hubway_trips.csv', low_memory=False)
         hubway_data.head()
```

Out[27]:

	seq_id	hubway_id	status	duration	start_date	strt_statn	end_date	end_statn	bike_nr	subsc_type	zip_code	birth_da
0	1	8	Closed	9	7/28/2011 10:12:00	23.0	7/28/2011 10:12:00	23.0	B00468	Registered	'97217	1976.0
1	2	9	Closed	220	7/28/2011 10:21:00	23.0	7/28/2011 10:25:00	23.0	B00554	Registered	'02215	1966.0
2	3	10	Closed	56	7/28/2011 10:33:00	23.0	7/28/2011 10:34:00	23.0	B00456	Registered	'02108	1943.0
3	4	11	Closed	64	7/28/2011 10:35:00	23.0	7/28/2011 10:36:00	23.0	B00554	Registered	'02116	1981.0
4	5	12	Closed	12	7/28/2011 10:37:00	23.0	7/28/2011 10:37:00	23.0	B00554	Registered	'97214	1983.0

This example and all material are provided by the Data Science course at Harvard University

Tabular Data

- Each type of measurement is called a **variable** or an **attribute** of the data (e.g. seq_id, status and duration are variables or attributes).
- The number of attributes is called the **dimension**. These are often called **features**.
- We expect each table to contain a set of **records** or **observations** of the same kind of object or event (e.g. our table above contains observations of rides/checkouts).

First Look At The Data

```
In [27]: hubway_data = pd.read_csv('hubway_trips.csv', low_memory=False)
         hubway_data.head()
```

Out[27]:

	seq_id	hubway_id	status	duration	start_date	strt_statn	end_date	end_statn	bike_nr	subsc_type	zip_code	birth_da
0	1	8	Closed	9	7/28/2011 10:12:00	23.0	7/28/2011 10:12:00	23.0	B00468	Registered	'97217	1976.0
1	2	9	Closed	220	7/28/2011 10:21:00	23.0	7/28/2011 10:25:00	23.0	B00554	Registered	'02215	1966.0
2	3	10	Closed	56	7/28/2011 10:33:00	23.0	7/28/2011 10:34:00	23.0	B00456	Registered	'02108	1943.0
3	4	11	Closed	64	7/28/2011 10:35:00	23.0	7/28/2011 10:36:00	23.0	B00554	Registered	'02116	1981.0
4	5	12	Closed	12	7/28/2011 10:37:00	23.0	7/28/2011 10:37:00	23.0	B00554	Registered	'97214	1983.0

Types of Data

Important to distinguish between classes of variables or attributes based on the type of values they can take on.

- **Quantitative variable:** is numerical and can be either:
  - **discrete -** a finite number of values are possible in any bounded interval.
    - For example: “Number of siblings” is a discrete variable
  - **continuous -** an infinite number of values are possible in any bounded interval.
    - For example: “Height” is a continuous variable

- **Categorical variable:** no inherent order among the values
  - For example: “What kind of pet you have” is a categorical variable

Common Issues

Common issues with data:

- Missing values: how do we fill in?
- Wrong values: how can we detect and correct?
- Messy format
- Not usable: the data cannot answer the question posed

Messy Data

- The following is a table accounting for the number of produce deliveries over a weekend.
- What are the variables in this dataset? What object or event are we measuring?

	Friday	Saturday	Sunday
Morning	15	158	10
Afternoon	2	90	20
Evening	55	12	45

What's the issue? How do we fix it?

Messy Data

We're measuring individual deliveries; the variables are Time, Day, Number of Produce.

	Friday	Saturday	Sunday
Morning	15	158	10
Afternoon	2	90	20
Evening	55	12	45

Problem: each column header represents a single value rather than a variable. Row headers are "hiding" the Day variable. The values of the variable, "Number of Produce", is not recorded in a single column.

Fixing Messy Data

We need to reorganize the information to make explicit the event we're observing, and the variables associated to this event.

ID	Time	Day	Number
1	Morning	Friday	15
2	Morning	Saturday	158
3	Morning	Sunday	10
4	Afternoon	Friday	2
5	Afternoon	Saturday	9
6	Afternoon	Sunday	20
7	Evening	Friday	55
8	Evening	Saturday	12
9	Evening	Sunday	45

Tabular = Happy Me ☺

Common causes of messiness are:

- Column headers are values, not variable names
- Variables are stored in both rows and columns
- Multiple variables are stored in one column/entry
- Multiple types of experimental units stored in same table

In general, we want each file to correspond to a dataset, each column to represent a single variable and each row to represent a single observation.

We want to **tabularize** the data. This makes Python happy.

Grammar

Exploratory Data Analysis (EDA)

Why?

- EDA encompasses the "*explore* data" part of the data science process
- EDA is crucial but often overlooked:
  - If your data is bad, your results will be bad
  - Conversely, understanding your data well can help you create smart, appropriate models

Exploratory Data Analysis (EDA)

What?

1. Store data in data structure(s) that will be convenient for exploring/processing (Memory is fast. Storage is slow)
2. Clean/format the data so that:
    - Each row represents a single object/observation/entry
    - Each column represents an attribute/property/feature of that entry
    - Values are numeric whenever possible
    - Columns contain atomic properties that cannot be further decomposed*

* Unlike food waste, which can be composted. Please consider composting food scraps.

Exploratory Data Analysis (EDA)

What? (continued)

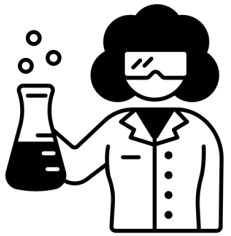
3. Explore ***global*** properties: use histograms, scatter plots, and aggregation functions to summarize the data
4. Explore ***group*** properties: group like-items together to compare subsets of the data (are the comparison results reasonable/expected?)

This process transforms your data into a format which is easier to work with, gives you a basic overview of the data's properties, and likely generates several questions for you to follow-up in subsequent analysis.

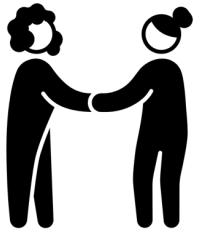
Break

CHILL

WALK



COFFEE OR TEA



MAKE FRIENDS

Icons by Iconathon on The Noun Project

EDA: without Pandas

Say we have a small dataset of the top 50 most-streamed Spotify songs, globally, for 2019.

EDA: without Pandas

Say we have a small dataset of the top 50 most-streamed Spotify songs, globally, for 2019.

NOTE: The following music data are used purely for illustrative, educational purposes. The data, including song titles, may include explicit language. TU Delft, including myself and the rest of the EPA1316 staff, does not endorse any of the entailed contents or the songs themselves, and we apologise if it is offensive to anyone in anyway.

EDA: without Pandas

top50.csv

Each row represents a distinct song. The columns are:

- **ID:** a unique ID (i.e., 1-50)
- **TrackName:** Name of the Track
- **ArtistName:** Name of the Artist
- **Genre:** the genre of the track
- **BeatsPerMinute:** The tempo of the song.
- **Energy:** The energy of a song - the higher the value, the more energetic.
- **Danceability**: The higher the value, the easier it is to dance to this song.
- **Loudness**: The higher the value, the louder the song.
- **Liveness**: The higher the value, the more likely the song is a live recording.
- **Valence**: The higher the value, the more positive mood for the song.
- **Length**: The duration of the song (in seconds).
- **Acousticness**: The higher the value, the more acoustic the song is.
- **Speechiness**: The higher the value, the more spoken words the song contains.
- **Popularity**: The higher the value, the more popular the song is.

EDA: without Pandas

	TrackName	ArtistName	Genre	BeatsPer Minute	Energy	Danceability	Loudness	Live ness	Valence	Length	Acousticness	Speechi ness	Popularity
1	Senorita	Shawn Menc	canadian pop	117	55	76	-6	8	75	191	4	3	79
2	China	Anuel AA	reggaeton flow	105	81	79	-4	8	61	302	8	9	92
3	boyfriend (w	Ariana Granc	dance pop	190	80	40	-4	16	70	186	12	46	85
4	Beautiful Pec	Ed Sheeran	pop	93	65	64	-8	8	55	198	12	19	86

Q1: What are some ways we can store this file into data structure(s) using regular Python (not the Pandas library).

EDA: without Pandas

top50.csv

	TrackName	ArtistName	Genre	BeatsPer Minute	Energy	Danceability	Loudness	Live ness	Valence	Length	Acousticness	Speechi ness	Popularity
1	Senorita	Shawn Menc	canadian pop	117	55	76	-6	8	75	191	4	3	79
2	China	Anuel AA	reggaeton flow	105	81	79	-4	8	61	302	8	9	92
3	boyfriend (w	Ariana Granc	dance pop	190	80	40	-4	16	70	186	12	46	85
4	Beautiful Pec	Ed Sheeran	pop	93	65	64	-8	8	55	198	12	19	86

Possible Solution #1: A 2D array (i.e., matrix)

Weaknesses:

- What are the row and column names? Need separate lists for them – clumsy.
- Lists are O(N). We'd need 2 dictionaries just for column names

data = [ ][ ]
col_name → index
index → col_name

EDA: without Pandas

top50.csv

	TrackName	ArtistName	Genre	BeatsPer Minute	Energy	Danceability	Loudness	Live ness	Valence	Length	Acousticness	Speechi ness	Popularity
1	Senorita	Shawn Menc	canadian pop	117	55	76	-6	8	75	191	4	3	79
2	China	Anuel AA	reggaeton flow	105	81	79	-4	8	61	302	8	9	92
3	boyfriend (w	Ariana Granc	dance pop	190	80	40	-4	16	70	186	12	46	85
4	Beautiful Pec	Ed Sheeran	pop	93	65	64	-8	8	55	198	12	19	86

Possible Solution #2: A list of dictionaries

list

Item 1 = {“Track.Name”: “Senorita”, “Artist.Name”: “Shawn Mendes”, “Genre”: “Canadian pop”, …}

Item 2 = {“Track.Name”: “China”, “Artist.Name”: “Anuel AA”, “Genre”: “reggaetón flow”, … }

Item 3 = {“Track.Name”: “Ariana Grande”, “Artist.Name”: “boyfriend”, “Genre”: “dance pop”, … }

EDA: list of dictionaries

Possible Solution #2: A list of dictionaries

```
f = open("../data/top50.csv", encoding = "ISO-8859-1")
column_names = f.readline().strip().split(",")[1:] # puts names in a list
cleaned_column_names = [name[1:-1] for name in column_names]
cleaned_column_names.insert(0, "ID")

dataset = []

# iterates through each line of the .csv file
for line in f:
    attributes = line.strip().split(",")

    # constructs a new dictionary for each line
    dataset.append(dict(zip(cleaned_column_names, attributes)))
```

EDA: list of dictionaries

Possible Solution #2: A list of dictionaries

Q2: Write code to print all songs (Artist and Track name) that are longer than 4 minutes (240 seconds):

```
for song in dataset:
    if int(song["Length."]) > 240:
        print(song["Artist.Name"], "-", song["Track.Name"], "is", song["Length."],"seconds long")
```

EDA: list of dictionaries

Possible Solution #2: A list of dictionaries

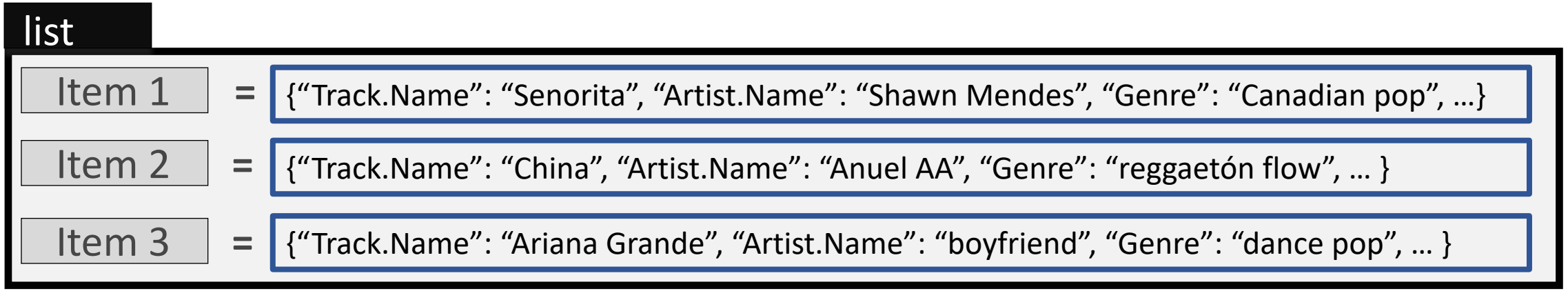
Q3: Write code to print the most popular song (artist and track) – if ties, show all ties.

```
max_score = -1
most_populars = set()
for song in dataset:
    if int(song["Popularity"]) > max_score:
        most_populars = set([str(song["Artist.Name"] + "-" + song["Track.Name"])])
        max_score = int(song["Popularity"])
    elif int(song["Popularity"]) == max_score:
        most_populars.add(str(song["Artist.Name"] + "-" + song["Track.Name"]))
print(most_populars)
```

EDA: list of dictionaries

Possible Solution #2: A list of dictionaries

Q4: Write code to print the songs (and their attributes), if we sorted by their popularity (highest scoring ones first).



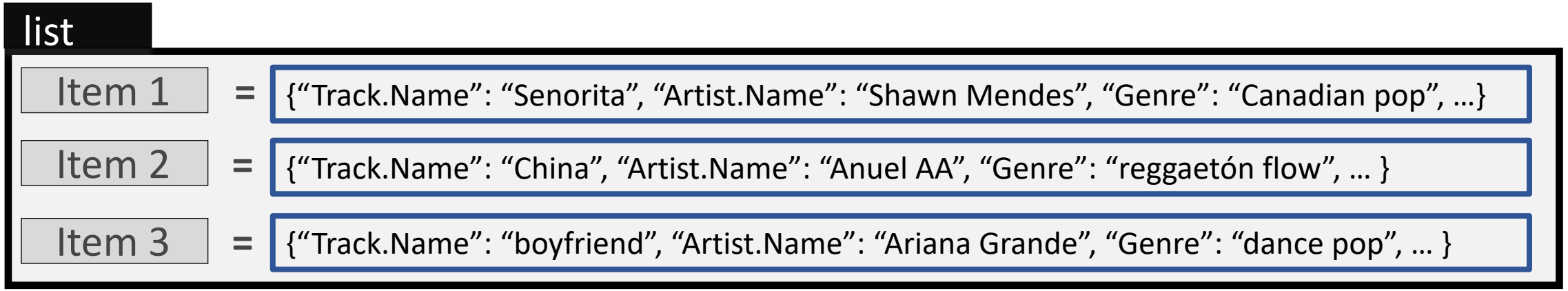
Cumbersome to move dictionaries around in a list.

Problematic even if we don’t move the dictionaries.

EDA: list of dictionaries

Possible Solution #2: A list of dictionaries

Q5: How could you check for null/empty entries? This is only 50 entries. Imagine if we had 500,000.



EDA: list of dictionaries

Possible Solution #2: A list of dictionaries

Q6: Imagine we had another table* below (i.e., .csv file). How could we combine its data with our already-existing *dataset*?

spotify_aux.csv

	TrackName	ArtistName	ExplicitLanguage
1	Senorita	Shawn Menc	TRUE
2	China	Anuel AA	FALSE
3	boyfriend (w	Ariana Granc	TRUE
4	Beautiful Pec	Ed Sheeran	FALSE

* 3rd column is made-up by me. Random values. Pretend they're accurate.

EDA: with Pandas!

EDA: with Pandas

What / Why?

- Pandas is an *open-source* Python library (anyone can contribute)
- Allows for high-performance, easy-to-use data structures and data analysis
- Unlike NumPy library which provides multi-dimensional arrays, Pandas provides 2D table object called **DataFrame**
  (akin to a spreadsheet with column names and row labels).
- Used by *a lot* of people

EDA: with Pandas

How

- import **pandas** library (convenient to rename it)
- Use **read_csv()** function

```
import pandas as pd
dataframe = pd.read_csv("yourfile.csv")
```

EDA: with Pandas

Common Panda functions

High-level viewing:

- head() – first N observations
- tail() – last N observations
- columns() – names of the columns
- describe() – statistics of the quantitative data
- dtypes() – the data types of the columns

EDA: with Pandas

Common Panda functions

Accessing/processing:

- df[“column_name”]
- df.column_name
- .max(), .min(), .idxmax(), .idxmin()
- <dataframe> <conditional statement>
- .loc[] – label-based accessing
- .iloc[] – index-based accessing
- .sort_values()
- .isnull(), .notnull()

EDA: with Pandas

Common Panda functions

Grouping/Splitting/Aggregating:

- groupby(), .get_groups()
- .merge()
- .concat()
- .aggegate()
- .append()

Data Concerns

When determining if a dataset is sound to use, it can be useful to think about these four questions:

- Did it come from a trustworthy, authoritative source?
- Is the data a complete sample?
- Does the data seem correct?
- **(optional)** Is the data stored efficiently or does it have redundancies?

Data Concerns: the format

- Often, there may not exist a single dataset that contains all the information we are interested in.
- May need to merge existing datasets
- Important to do so in a **sound** and **efficient** format

Data Concerns: the format

Dataset 1

Top 200 most-frequent streams per day (for June 2019)

	SpotifySongID,	# of Streams,	Date
200	2789179,	42003,	06-01
	•		
	•		
	3819390,	89103,	06-01
200	4492014,	52923,	06-02
	•		
	•		
	8593013,	189145,	06-02

6,000 x 3

Dataset 2

Top 50 most streamed in 2019, so far

	SpotifySongID,	Artist,	Track, [10 acoustic features]
50	2789179,	Billie Eilish,	bad guy, 3.2, 5.9, …
	•		
	•		
	3901829,	Outkast,	Elevators, 9.3, 5,1, …

50 x 13

This example and all material are provided by the Data Science course at Harvard University

Data Concerns: the format

We are interested in determining if songs with high danceability are more popular during the weekends of June than weekdays in June. What should our merged table look like? Concerns?

Data Concerns: the format

This is wasteful, as it has 10 acoustic features, artist, and track repeated many times for each unique song.

Datasets Merged (poorly)

SpotifySongID, # of Streams, Date, Artist, Track, [10 acoustic features]

200

2789179, 42003, 06-01 Billie Eilish, bad guy, 3.2, 5.9, …

•
•

3819390, 89103, 06-01 Outkast, Elevators, 9.3, 5,1, …

200

4492014, 52923, 06-02

•
•

8593013, 189145, 06-02

6,000 x 15 ➔ 90,000 cells

Data Concerns: the format

Some rows may have null values for # of Streams (if the song wasn't popular in June)

Datasets Merged (better)

SpotifySongID, Artist, Track, [10 acoustic features], 06-01 Streams, 06-02 Streams

50

2789179, Billie Eilish, bad guy, 3.2, 5.9, …, 42003, 42831, 43919

⋮

3901829, Outkast, Elevators, 9.3, 5,1, … 29109, 27193, 25982

50 x 43 ➔ 2,150 cells

Data Concerns: the format

- Is the data correctly constructed (or are values wrong)?
- Is there redundant data in our merged table?
- Missing values?

The Parable of Google Flu: Traps in Big Data Analysis

Large errors in flu prediction were largely avoidable, which offers lessons for the use of big data.

David Lazer,[1,2*] Ryan Kennedy,[1,3,4] Gary King,[3] Alessandro Vespignani[5,6,3]

In February 2013, Google Flu Trends (GFT) made headlines but not for a reason that Google executives or the creators of the flu tracking system would have hoped. *Nature* reported that GFT was predicting more than double the proportion of doctor visits for influenza-like illness (ILI) than the Centers for Disease Control and Prevention (CDC), which bases its estimates on surveillance reports from laboratories across the United States (*1*, *2*). This happened despite the fact that GFT was built to predict CDC reports. Given that GFT is often held up as an exemplary use of big data (*3*, *4*), what lessons can we draw from this error?

The problems we identify are not limited to GFT. Research on whether search or social media can predict *x* has become commonplace (*5*–*7*) and is often put in sharp contrast with traditional methods and hypotheses. Although these studies have shown the value of these data, we are far from a place where they can supplant more traditional methods or theories (*8*). We explore two issues that contributed to GFT's mistakes—big data hubris and algorithm dynamics—and offer lessons for moving forward in the big data age.

ability and dependencies among data (*12*). The core challenge is that most big data that have received popular attention are not the output of instruments designed to produce valid and reliable data amenable for scientific analysis.

The initial version of GFT was a particularly problematic marriage of big and small data. Essentially, the methodology was to find the best matches among 50 million search terms to fit 1152 data points

run ever since, with a few changes announced in October 2013 (*10*, *15*).

Although not widely reported until 2013, the new GFT has been persistently overestimating flu prevalence for a much longer time. GFT also missed by a very large margin in the 2011–2012 flu season and has missed high for 100 out of 108 weeks starting with August 2011 (see the graph). These errors are not randomly distributed. For example, last week's errors predict this week's errors (temporal autocorrelation), and the direction and magnitude of error varies with the time of year (seasonality). These patterns mean that GFT overlooks considerable information that could be extracted by traditional statistical methods.

Even after GFT was updated in 2009, the comparative value of the algorithm as a stand-alone flu monitor is questionable. A study in 2010 demonstrated that GFT accuracy was not much better than a fairly simple projection forward using already available (typically on a 2-week lag) CDC data (*4*). The comparison has become even worse since that time, with lagged models significantly outperforming GFT (see the graph). Even 3-week-old CDC data do a better job of projecting current flu prev-

For next class..

Finish Labs to practice programming

Complete Homework and review your peers' work

Check Assignment contents and due date

See "To do before class" for next lecture (~ 1 hour of self study)

Read paper for **Discussion** session before next week (~ 1 hour)

Post questions on the **Discussion** forum on Brightspace