BOISE STATE UNIVERSITY

# CS 533
# INTRO TO DATA SCIENCE

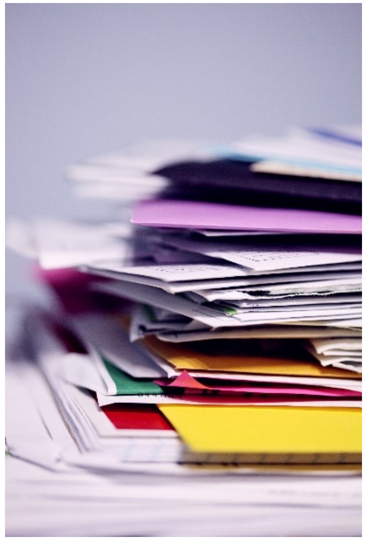
Instructor: Jun Zhuang

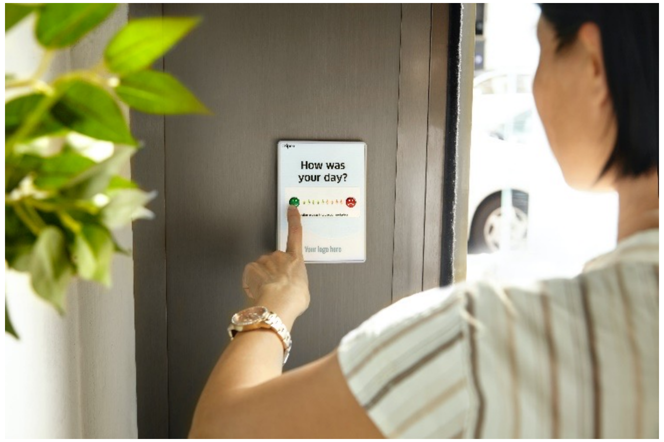
Source: https://cs533.ekstrandom.net/f22/

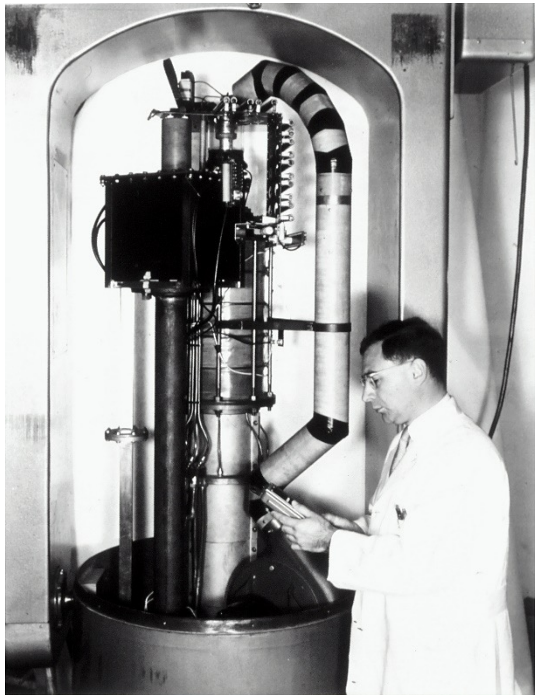
# Data

# Sources of Data

- Business records
- Administrative records
- Public service organizations
- Physical observations
- Surveys
- Experiments (physical or social)
- Online services / observations





# What Should We Do After Obtaining The Data?

- Get the raw data files
- Merge data if necessary
- Transform data into usable format
- Extract the data set needed for your task
- Preprocessing the data

# Locating Existing Data

- Lists of data sets
  - [UCI Machine Learning Repository]
  - Various lists on GitHub
- Governmental operations
  - US federal: data.gov
  - Individual government agencies
  - Government data portals
- Your organization
- Searching on the Web
- Purchasing
- Asking data owners
- Scraping from web sites
- Large repositories
  - Common Crawl
  - Semantic Scholar
- Seeing what other papers use

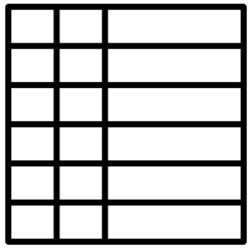
# In-class Exercise

- Down a covid dataset (csv file) from data.gov https://catalog.data.gov/dataset/mental-health-care-in-the-last-4-weeks

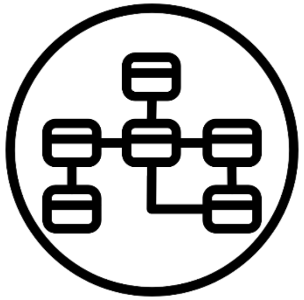
- Perform basic analysis on this dataset and answer the following questions:

1. List of average "Value" against each "Group" and "State", respectively.
   - What's the value of "Idaho"?
   - Which state has the highest value?
2. List of min "LowCI" and max "HighCI" against each "Time Period Label", respectively.
   - Merge the above two tables based on the "Time Period Label".
   - Which time period has a minimum gap between "HighCI" and "LowCI"?

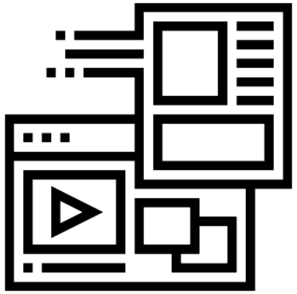
# Types of Data



**Tabular** is organized into columns and rows like a spreadsheet.
Each row has the same shape & attributes.

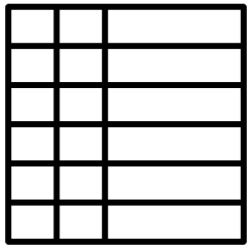


**Semi-Structured** data has structure, like labeled fields, but different objects can have different fields.



**Unstructured** data has no defined structure. Includes raw text and images.

# Types of Data



**Tabular** is organized into columns and rows like a spreadsheet.
Each row has the same shape & attributes.

**Delimited text**

- Comma-separated (CSV)

**Spreadsheet files**

- Excel (xlsx or xls)

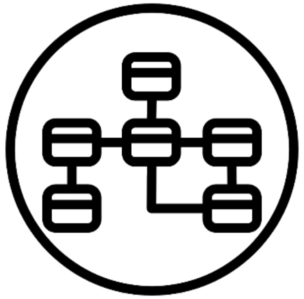
**Binary formats**

- HDF, NetCDF, Parquet

**Other**

- Matlab, STATA

# Types of Data



**Semi-Structured** data has structure, like labeled fields, but different objects can have different fields.

**JSON:** dictionaries, lists, strings, numbers, true/false, null

**XML:** trees of nodes with text and attributes

**YAML:** JSON ++

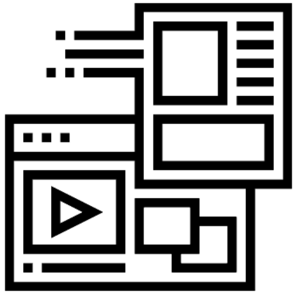
**MSGPACK**

**RDF**, **SQL**

Others…


```
{
 "fit": "fit",
 "user_id": "420272",
 "bust size": "34d",
 "item_id": "2260466",
 "weight": "137lbs",
 "rating": "10",
 "rented for": "vacation",
 "review_text": "An adorable romper! Belt and zipper were a
little hard to navigate in a full day of wear/bathroom use, but
that's to be expected. Wish it had pockets, but other than that--
absolutely perfect! I got a million compliments.",
 "body type": "hourglass",
 "review_summary": "So many compliments!",
 "category": "romper",
 "height": "5' 8\"",
 "size": 14,
 "age": "28",
 "review_date": "April 20, 2016"
}
```

# Types of Data



**Unstructured** data has no defined structure. Includes raw text and images.

**Raw text** — (usually) human-written text

**Images** — we can try to get data from images

Can appear as a field in tabular or semi-structured data.

# Look at Data

Mac/Linux: less

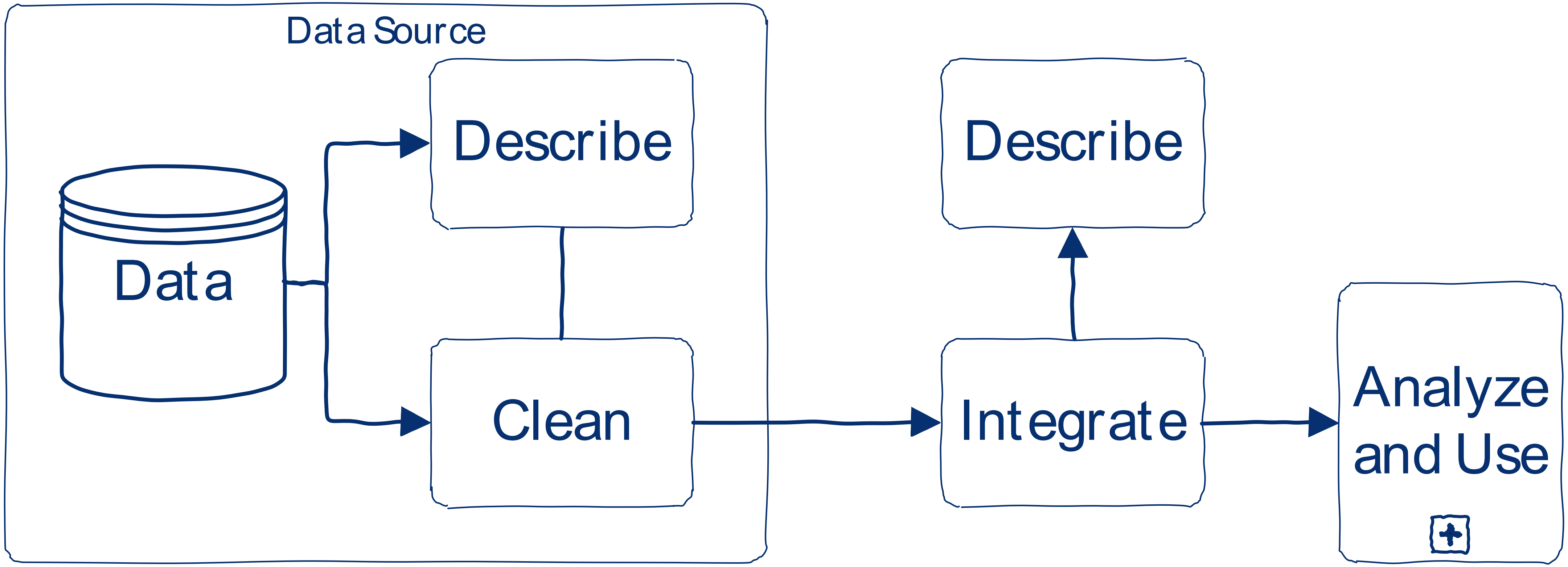
Windows: Notepad++ (or other text editor)

Data may be compressed, usually indicated in extension

- .gz
- .xz
- .7z
- .zip

Unix: zless, etc.

# Data Workflow



# Wrapping Up

Data comes from a variety of sources and comes in many formats.

Sometimes it is tabular, semi-structured, or unstructured.

*Photo by [Wesley Tingey](#) on [Unsplash](#)*

# Data Cleaning

# Types of Cleaning

- Convert data types
- Standardize data codes
- Remove or clean corrupt data
- Fill missing data (with care)

# Basic Data Type Conversion

`.astype()`

- Converts data from one type to another
- Parses strings w/ simple rules

```
df['column'].astype('i4')
```

Common NumPy data types:

- Integer: i1, i2, i4, i8
- Unsigned: u1, u2, u4, u8
- Float: f4, f8

Also have bit-based sizes:

- i4 = int32
- f8 = float64 (double-precision)

# Standardizing Data

- Normalize missing data
  - String encoding like 'NA'?
  - Numeric sentinel values like -999?
  - Reassign to NA
- Unify case (upper, lower, title, casefold) [`series.str.upper`]
- Replace substrings [`series.str.replace`]
- Trim whitespace [`strip/rstrip/lstrip`]
- Rename codes [`cat.rename_categories`]
- Merge codes [reassign, then `cat.remove_unused_categories`]

# Cleaning Data

Strings are often corrupt – excess characters, etc.

- Drop leading/trailing whitespace [`strip` and friends]
- Match with regular expressions
  - Expression to match expected data & keep
  - Expression to match invalid data & delete
  - `series.str.replace(regex, replace)`
- Extract specific columns [`series.str.slice`]

# Cleaning Data

Sometimes values are unrecoverably corrupt

- Delete value (replace with NA or INVALID code)
  - May separate UNKNOWN from INVALID
  - Or just use one UNKNOWN code
  - Depends on question – I often separate early, combine later
- Delete record (if unusable)
- Don’t delete from underlying files – in memory, or in new files

# Wrapping Up

Data is messy.

Pandas gives us a number of tools for working with individual values or columns.

Photo by [Ashim D'Silva](#) on [Unsplash](#)

# Data Integration

# Types of Integration

**Linking records** – matching records in one set with another

Best case: we have a *linking identifier* shared between data sets.

**Pooling records** — taking records of the same kind from different sources

Convert each into common format, and stack!

# Example: Linking US Geopolitical Data

- State name (unique, fine)
- Postal code (2-character state abbreviation, unique, also fine)
- FIPS code (Federal Information Processing ~~Standard~~ Series)
  - States and counties!
  - Withdrawn but still in use
  - Great when you have them!
- ZIP codes
- Legislative districts
- Census tracts
- Geographic position (lat, long) (ugggh)

# Linking Challenges

- Corrupt identifiers
  - Clean and correct them
- Duplicate identifiers
  - Measure frequency of occurrence, try to measure impact
- Missing identifiers
  - Find alternate linking strategies

# Alternate linking strategies

- Names?
  - Often not unique
  - Often take different forms
- Locations?
  - Require complex geographic matching
  - Or address matching / normalization

Linking takes creativity and care

# Technical Pieces (for linking with Pandas)

Cleaning up individual columns

- Series operations (esp. string ops!)

Merge data frames

- `pd.merge` or `pd.DataFrame.join` (for linking records)

# Pooling Records

1. Convert into common structure
2. Stack on top of each other
3. Sometimes: de-duplicate

Usually good to keep a field identifying record source.

`pd.concat` is your friend

# Wrapping Up

We often need to combine data from multiple sources; sometimes linking, sometimes pooling.

Linking identifiers make this easy (sometimes).

We don't always have them.

*Photo by Joshua Hoehne on Unsplash*