DATA1002 Week 6 Tutorial

Monday 08/09/25

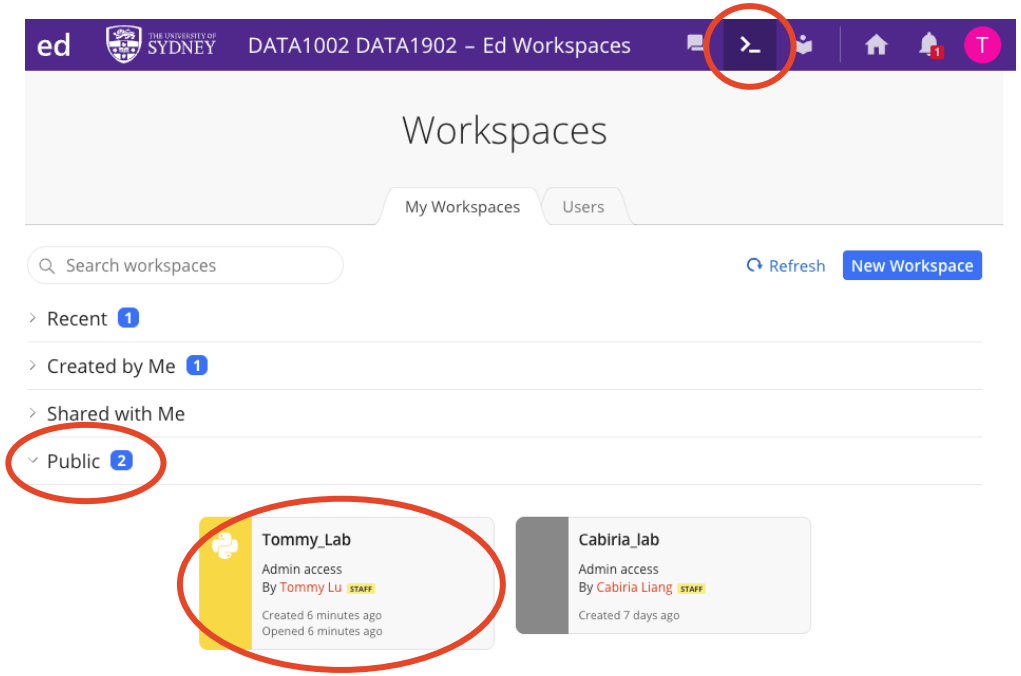
Tutorial Outline

- Content revision (Data Formats), Python Demo
- Content revision (Python Datasets), Python Demo
- Work on Assignment 1
  - Task G
  - Task T

THE UNIVERSITY OF SYDNEY

Tutor: *Tommy Lu*

Access the material for this tutorial through **Ed Workspaces**



Housekeeping

1st hour we'll be revising content.
2nd hour we'll be working on the Assignment.

Let me know if you're not in a group!

Group Project Stage 1

Due: 17:00 pm on Sunday at the end of week 8 (Sep 28th)

Content Revision

Logical Data Formats

Information Equivalence

- There can be several different schemas for storing the same knowledge of the world
  - Each has the same information content
  - But some might be:
    - stored in less space
    - easier for humans to read
    - quicker to process with a program
- One could write a program that automatically transforms files in one schema into one of the other schemas

Simple Variations

- Order of fields in a row
  - E.g., convert a file about chemicals with columns [official name, common name, hazard rating, quantity] to:
    - one with columns [official name, quantity, hazard rating, common name]
- Units for fields
  - E.g., convert a file about projects with equipment cost in $'000, to one with cost in $, by multiplying by a factor 0.001

Wide vs. Long

Wide

- One row per entity
- Several related columns, that vary along some aspect
- Column name indicates the variation

Stock	Price_2018	Price_2019	Price_2020
BHP	29.57	34.23	35.82
CSL	141.30	185.16	287.00

Long

- Multiple rows per entity
- Each row has one column containing the variant, and another containing the corresponding value

Stock	Year	Price
BHP	2018	29.57
BHP	2019	34.23
BHP	2020	35.82
CSL	2018	141.30
CSL	2019	185.16
CSL	2020	287.00

Limitations

- Wide format requires one to know the range of possible values for one varying aspect
  - E.g., to have a schema with columns Price_2018, Price_2019, Price_2020 (and no other Price_XXXX colums), you need to be sure that you will only be interested in data for years 2018, 2019, 2020
- In contrast, Long format can be used for datasets where there are any range of years

Identifiers vs. File Keys

Identifiers vs. File Keys

- Entities in the domain are almost always given unique identifiers
  - Values which belong to only one entity
  - These are often not intrinsically meaningful
  - They allow references to the entity in datasets

VS

- In a schema, some column (or some combination of columns) will be distinct among all the rows
  - E.g., in weather data (site, date, maxtemp, rainfall, etc) there are several rows for any particular site, and several for any particular date, but only one row for any (site, date) combination
- We call a set of columns a key, if their values are distinct among rows

Python Exercise

Pivoting Data

Dealing With Lists

- You are provided with a csv called *student_scores.csv*.
- This data is provided to you in wide format
- By looking up the Pandas function '**melt**', convert the data into long format

- **EXTENSION**: figure out how to '**unmelt**' your data frame after pivoting it.

```
Original DataFrame:
      Name  Math  English  Science
0    Ayman    85       92       78
1   Biling    70       88       90
2  Charlie    95       85       85
```

```
Long-format DataFrame:
      Name  Subject  Score
0    Ayman     Math     85
1   Biling     Math     70
2  Charlie     Math     95
3    Ayman  English     92
4   Biling  English     88
5  Charlie  English     85
6    Ayman  Science     78
7   Biling  Science     90
8  Charlie  Science     85
```

Definition and Usage

The `melt()` method reshapes the DataFrame into a long table with one row for each each column.

Syntax

```
dataframe.melt(id_vars, value_vars, var_name, value_name, col_level, ignore_index)
```

Parameters

The `id_vars`, `value_vars`, `var_name`, `value_name`, `col_level`, `ignore_index` parameters are keyword arguments.

Parameter	Value	Description
id_vars	*Tuple* *List* *Array*	Optional, specifies the column, or columns, to use as identifiers
value_vars	*Tuple* *List* *Array*	Optional, specifies columns to unpivot.
var_name	String	Optional, specifies the label of the 'variable' column, default 'variable'
col_level	*Number* *String*	Optional, for MultiIndex DataFrames, specifies the level to melt
ignore_index	`True` `False`	Optional, default True. Specifies whether to ignore the original index or not

Content Revision

Communication & Datasets in Python

Note on Communication

- **Clarify goals**: what you want the audience to know, feel, or do.
- Attend to **context**: tailor explanations to the audience's knowledge, expectations, and needs.
- Be **clear**, **concise**, and **respectful** of their time.
- Maintain **professionalism** (formal language, tone).

Design Decisions

- The data structure used by the program does not need to correspond closely with the structure of the file the data came from
- But it's often easiest to bring the data into a structure with structure that is closely related to the file arrangement
  - And then re-arrange the data if needed, into other (informationally equivalent) structures

Example

- Suppose we have a file about shares, with schema stock, year, price, dividend

```
stock,year,price,dividend
BHP,2018,29.57,0.58
BHP,2019,34.23,0.60
BHP,2020,35.82,0.60
CSL,2018,141.30,0.23
CSL,2019,185.16,0.23
CSL,2020,287.00,0.25
```

BRAINSTORM: After reading in the csv (Pandas!), how could we store this dataset in Python? Why?

List of Lists

- The dataset is stored as a list
  - One entry per line of the file
- Each entry of the list is itself a list, with one entry for each field of the line
  - **E.g.,** `data = [["BHP", 2018, 29.57, 0.58], ["BHP", 2019, 34.23, 0.60], ...]`

One Dictionary

- The dataset is stored as a dictionary
  - One entry per line of the file
- Key of the dictionary is a tuple formed from the columns of the file key
- Value of the dictionary is a tuple formed from the other columns
  - **E.g.,** `data =`

```
{("BHP",2018):(29.57,0.58),("BHP",2019):(34.23,0.60),
...}
```

List of Dictionaries

- Each entry of the list is a dictionary, with key being attribute name
- Example:
  - `data = [{"stock":"BHP", "year":2018, "price":29.57, "dividend":0.58}, {"stock":"BHP", "year":2019, price":34.23, "dividend":0.60}, ...]`

Several Dictionaries

- The information about each attribute is stored as a **dictionary**
  - Key being the file key value from that row
- Example:

```
price_data = {("BHP", 2018):29.57, {("BHP", 2019):34.23, ...}

dividend_data = {("BHP", 2018):0.58, {("BHP", 2019):0.60, ...}
```

JSON (JavaScript Object Notation)

- Very common format for data exchange between websites
  - represents a fact as a pair with value and its meaning/key
- These sets of pairs can be nested, and arranged in collections (either ordered or unordered)
- This is semi-structured; it's more flexible than structured data


```
{
    "orders": [
        {
            "orderno": "748745375",
            "date": "June 30, 2088 1:54:23 AM",
            "trackingno": "TN0039291",
            "custid": "11045",
            "customer": [
                {
                    "custid": "11045",
                    "fname": "Sue",
                    "lname": "Hatfield",
                    "address": "1409 Silver Street",
                    "city": "Ashland",
                    "state": "NE",
                    "zip": "68003"
                }
            ]
        }
    ]
}
```

Python Exercises

Storing data in Python

Storing Data in Python

Four different data storage approaches were discussed. Try implementing at least one for provided *stocks.csv*.
EXTENSION: Try all four, then explain why certain approaches are more suitable in certain use cases.

List of Lists

[["BHP", 2018, 29.57, 0.58], ["BHP", 2019, 34.23, 0.60], ...]

One Dictionary

{("BHP",2018):(29.57,0.58),("BHP",2019):(34.23,0.60), ...}

List of Dictionaries

[{"stock":"BHP","year":2018,"price":29.57,"dividend":0.58},{"stock":"BHP","year":2019,price":34.23, "dividend":0.60}, ...]

Several Dictionaries

price_data = {("BHP", 2018):29.57, {("BHP", 2019):34.23, ...}
dividend_data = {("BHP", 2018):0.58, {("BHP", 2019):0.60, ...}

Let’s Take a Short Break!

Lab Activities

Working on Assignment 1

Where should you be at this week?

- Make sure you hold at least one group meeting during the week
  - Plan the process for writing the report; decide on the tool to use - e.g., GoogleDocs, MS Words, LaTex…
- Each member individually produce a clean high-quality version of their dataset

☑ Cleaned individual dataset

☑ Started to put together summary statistics

☑ Thought about how you might want to integrate the datasets and present it in the report

Activity G (Group)

***Goal**: plan the data integration work.*

- The starting point should be the schema of each members dataset, and the outcome should be an agreed schema for integrated data
- To Pass this part of the project, you need to pick up attributes from at least two of the members' datasets
- In choosing a schema for integrated data, make sure that you discuss how you will transform data from the original sets to produce the actual data that goes with the integrated schema you have chosen.
- We suggest starting with just a few attributes, and then gradually considering if others can be added too

Activity T (group & Tutor)

- Present your method of data cleaning to the tutor
- Each member take turn to present their Python code to the tutor
- The tutor will provide feedback on what you show them.
  - If some member of the group is absent, the rest of the group can show their work from the shared storage and take note of the feedback to report back to the student concerned!

Exam-Style Questions

Question 1:

How does the choice of data format impact the efficiency and effectiveness of data analysis in a data science project? Provide examples to support your answer.

Exam-Style Questions

The choice of data format significantly impacts the efficiency and effectiveness of data analysis. For instance, **CSV (Comma-Separated Values)** files are easy to **read and write**, making them ideal for simple data storage and quick analysis.

However, they lack the ability to store hierarchical data structures, which `JSON (JavaScript Object Notation)` can handle efficiently.

JSON is more suitable for **web applications and APIs,** where data often has nested structures. In a data science project, using a CSV file for tabular data and JSON for complex, nested data can **optimise the data processing pipeline**. Choosing the appropriate format ensures that data is easily accessible and manageable, reducing the time and effort required for data cleaning and preparation, ultimately leading to more efficient and accurate analysis.

Exam-Style Questions

Question 2 [DATA1002]:

Discuss the ethical considerations in communicating data science results, including the risk of overclaiming and the need for transparency.

Exam-Style Questions

Ethical considerations in communicating data science results are paramount to maintaining trust and integrity.

Overclaiming results, such as su**ggesting causal relationships where only correlations exist, can mislead stakeholders and lead to poor decision-making**. Transparency is essential in conveying the limitations of the analysis, such as **data quality issues, assumptions made, and the scope of applicability**.

For example, in a predictive model for customer churn, it is crucial to disclose the model's **accuracy and potential biases** due to the training data.

Ethical communication also involves **declaring any conflicts of interest and the sources of funding to avoid perceived biases**. By adhering to ethical standards, data scientists ensure that their communication is honest, clear, and trustworthy, fostering confidence in their work and the decisions based on it.

That’s it folks!

Remaining Ed Lessons, Questions, Assignment etc.