DAYZERO DIAGNOSTICS

Engineering Project

Project Goal: Build a data pipeline and structure for clinical bacterial isolate data

Estimated Time: 3-5 hours

Return: Code, formatted data

Introduction:

First, thank you for taking the time to work through this project. Your efforts here help us get a better sense of how you approach real problems that we work on at DZD, and we hope this project gives you a view into the type of work you might encounter in our company.

This project is centered on parsing and organizing data, something we do quite a bit of. DZD is developing MicrohmDB®, a database that contains the phenotypic and genotypic data of bacterial clinical isolates. Part of the challenge of building MicrohmDB is ingesting data from multiple sources, particularly from hospital EHR systems, and representing the data in a way that can later be easily used for downstream applications. For this project we will focus on organizing the phenotypic metadata of clinical isolates (without worrying about the genomic sequencing information). There are two csv files provided, and the goal is to write a pipeline that ingests these csv's, parses any necessary columns, and reorganizes them into tables that are more intuitive for downstream applications.

Hospital Provided Phenotype Data:

In a hospital, patient samples (e.g. blood suspected of harboring bacteria) are sent to the hospital's clinical microbiology lab. There the samples are *cultured*, i.e. the bacteria in these samples are grown, yielding *isolates* – single bacterial colonies isolated from a petri dish. Afterwards, each isolate is subjected to *phenotypic testing* to assess what the species of the bacteria is, and what antibiotics the isolate is resistant to. A bacteria can be *resistant*, *susceptible*, or *intermediate* (somewhere in between fully susceptible and fully resistant) to a given antibiotic. Hospitals store this metadata information about each isolate in their EHR system, often with their own custom formatting scheme.

One of the many challenges with using data from hospitals is that it can be messy and inconsistent. PhenotypeData.csv is a mock data dump from a hospital, with information about many isolates. Each row corresponds to a single test that was performed on an isolate. The columns in this dataset are as follows:

- *HID*: Hospital Identifier. *HID+Isolate* are both needed to uniquely identify a single isolate.
- *Isolate*: Isolate identifier. There can be multiple isolates for a given *HID*, so *HID+Isolate* uniquely identify an isolate. ISO1 indicates Isolate 1, ISO2 Isolate 2, etc.
- *Received*: Date received
- *Organism*: Species identified for the isolate

- *Source*: The clinical source of the sample, e.g. blood
- *Test*: The rest of the columns on this row refer to a particular test performed on the isolate. The most relevant tests are the "Susceptibility", "Pheno Susceptibility", and "E-Test Sensitivity" tests, which test if the isolate is resistant or susceptible (also known as sensitive) to a particular antibiotic.
- *Antibiotic*: The antibiotic tested
- *Value*: The measured value of the antibiotic resistance test
- *Antibiotic Interpretation*: Resistant, Susceptible, or Intermediate resistance call for that isolate and that antibiotic
- *Method*: Which machine / method was used to perform the test.

DZD Isolate Collection Data:

Thus far we have only touched on the hospital metadata regarding isolates, independent of the DZD collection process. But DZD must actually physically receive clinical isolates from the hospital in order to perform genomic sequencing on them, and then be able to know which rows of the phenotype metadata table correspond to which collected isolate. The process for doing that is as follows: in a hospital clinical microbiology lab, a technician collects a subset of all of the isolates in the lab specifically for DZD. In a spreadsheet, for each isolate that is collected, the technician notes the unique hospital identifier for the isolate, and also a newly generated DZD identifier for the isolate. Internally (in downstream applications), the DZD identifier is what gets used, while the hospital identifier is only used to join to hospital metadata. The isolates that are collected are shipped to DZD, and the collection spreadsheet is provided to DZD as well.

CollectionData.csv is the data taken from collecting:

- *sampid*: Unique DZD identifier, will be used in all downstream applications
- *HID*: Hospital Identifier, should match *HID* in PhenotypeData.csv
- *Isolate # (blank = "1")*: This is the isolate number, but for ease of recording the blanks indicate a "1". The isolate number is needed with the *HID* to uniquely identify the isolate; e.g. if the isolate number was 2, this should match "ISO2" in the *Isolate* column from PhenotypeData.csv
- *Date Collected:* Date of collection for DZD

Task:

The goal is to (1) write code in Python 3 to parse these files and (2) design a database structure to represent that data. We ask for Python 3 because that's the main programming language at Day Zero.

In a full-blown system, the resulting table structures would be designed for a SQL database. In this case something scaled down (e.g. sqlite, or just csv outputs) is fine for representing what the tables would look like. We are intentionally leaving this open ended, and the goal for you is to think through what the data needs might be and how to architect a pipeline that can be robust. Code can be accompanied with

representations of what the table structure would be or written explanation / discussion points. Some suggestions in thinking through this:

- The most important aspects for downstream applications are the use of the DZD ID, the species, and the resistance interpretations across the tested drugs
- There may be some parsing necessary – how to save raw / intermediate data?
- How can the process ensure robustness to different spellings and abbreviations, e.g. “Trimeth/Sulfa” and “Trimethoprim/Sulfa” are both abbreviations for the antibiotic “Trimethoprim/Sulfamethoxazole”.
- Note that not all collected isolates may be represented yet in the phenotypic data (data dumps are received from the hospital at regular intervals), and not all phenotypic data may correspond to collected isolates.

Bonus:

In PhenotypeData.csv, the column *Value* provides the raw measurement of the antibiotic resistance test. This *Value* measurement is written as being less than, equal to, or greater than a number - due to the imprecision of the actual test itself sometimes inequalities are all that can be measured. The column *Antibiotic Interpretation* translates that value into one of three interpretations: Resistant, Intermediate or Susceptible, depending on which range the value falls into. The interpretation in the provided PhenotypeData.csv is based on the hospital’s interpretive rules, but in some cases DZD might want to reinterpret the values (for example, if we think the hospital is using outdated rules, which unfortunately does happen!).

To reinterpret these values, for every combination of Organism/Antibiotic/Method, assume that DZD has determined a data rule for how to convert values to resistance interpretations. For example, for the Organism/Antibiotic/Method combination *Escherichia coli*/Ceftazidime/VITEK II, the translation of *Value* to *Interpretation* is provided as:

- If *Value* <=4: Susceptible
- If 4 < *Value* < 16: Intermediate
- If *Value* >=16: Resistant

Suppose you were given these set of rules. What structure would you store them in? How would you change the pipeline to reinterpret the *Value* column? (An actual code implementation here is great - though can be a little tricky!)