

Road to JUMP Hackathon II

Alán F. Muñoz

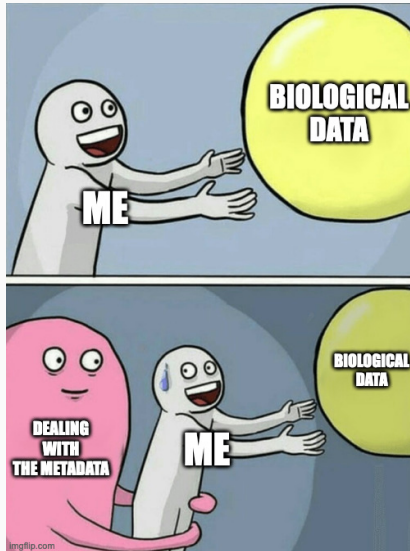
2024/07/18

Introduction: The challenges of accessing JUMP data

The problem

- Loads of data and metadata.
- Limited access methods, most of which are documented in notebooks across repositories.

The problem



Before we start

We will split into two section later.

WSL/MacOS

```
python -m venv .venv  
source .venv/bin/activate  
pip install jump_portrait
```

Nix

```
mkdir demo  
cd demo  
nix flake init -t \  
github:broadinstitute/monorepo#portrait
```

Test installation

```
python -c "import jump_portrait"
```

What we will learn

- A new alternative for reproducible isolated environments
- Some commands that facilitate working on remote servers
- How to access JUMP data within Python

Available data

- Images for ~140k perturbations
- Metadata
- Morphological profiles and their corrected versions
- Non-written knowledge (e.g., which JCP IDS are controls in CRISPR dataset)

The basic needs of biologists are covered

JUMP_RR (Round-Robin) Provides simple interfaces to pre-processed data:

- Perturbation-to-perturbation match (broad.io/crispr)
- Statistically significant features (broad.io/crispr_feature)
- Image exploration (broad.io/crispr_gallery)

For [<crispr>](https://broad.io/crispr), [<orf>](https://broad.io/crispr_feature) and (gallery-only) [<compounds>](https://broad.io/crispr_gallery).

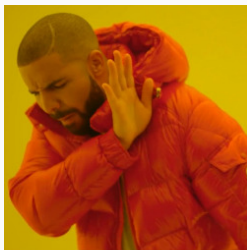
How did we use to access each data/metadata

- Images: AWS S3
- Metadata: Github repository
- Profiles: AWS S3, but before this P2P

Notebooks served as examples

- External data sources use "standard ids" (i.e., NCBI), but many internal ones use Broad or JUMP ids
- How do we link metadata to images/profiles?

So instead of notebooks, I build modules/libraries to make my life more pleasant



Notebooks



Modules

Today's focus: Developers and Data
Scientists

Main goal: Concentrate the **essential knowledge** necessary to process JUMP perturbations.

Essential use-cases:

- Is this JUMP ID the same as this Entrez (NCBI) ID?
- Is X a treatment or control? If the latter, which type? (yes, there are types of positive controls)

Additional use-cases:

- Can I get a mapper from JUMP IDS to gene symbols or InChiKeys?
- Can I export the entire table for CRISPR, ORF and Compounds mapping perturbations to control type?

Main goal: Fetch JUMP images.

Essential use-cases:

- What is the [Source, Batch, Plate, Well] of my perturbation with standard name X?
- Give me the available images for [Source, Batch, Plate, Well] X.
- Give me the control images for [Source, Batch, Plate] X.

Additional use-cases:

- Download images and controls straight to disk

Choose your own adventure

You can choose what to do

- Follow the main demo
- Test the limits of `jump_portrait` and/or `broad_babel` and see if they breaks

The breaking game



The breaking game

- Input gene names must be present in JUMP (you can check broad.io/babel)
- Inputs must respect documentation and typing
- Likely bug locations:
 - Threaded components
 - Metadata with missing images
 - Edge cases when
- Useful things to look out for:
 - Portrait: Lazy+Anonymous S3 access via Polars
 - Babel: Pooch to download datasets only once and keep them in disk

Some ideas of things to pay attention to

- These are some edge cases
 - Brightfield fetching available but barely supported, but works for some images. Is there a way to reliably identify bright field channels?

Should we?

- How do we deal with invalid entries ("null", instead of "negcon" or "trt")?
- There is still data with low accessibility. -For instance, someone needed to get an **Images.csv**. For reasons (?).
- Should we aim to make everything fully transparent and accessible? Or only the sections that we consider "public-ready".

Walkthrough




















- Load the corrected CRISPR profiles
- Select the features that are also present on the ORF dataset
- Calculate the most anticorrelated profiles
- Pick a feature at random (seed=42), sort it and get five JCP IDs that range from min to max.
- Find the gene name
- Fetch images for this gene in both CRISPR and ORF
- find the other available names for these genes

Open call for suggestions

- What is the best way to clean test artifacts?
- Threading is a pain. Are there alternatives?
 - joblib
 - Pathos
 - Multiprocessing

One final remark

HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE?
(ACROSS FIVE YEARS)

		HOW OFTEN YOU DO THE TASK					
		50/DAY	5/DAY	DAILY	WEEKLY	MONTHLY	YEARLY
HOW MUCH TIME YOU SHAVE OFF	1 SECOND	 DAY	2 HOURS	30 MINUTES	4 MINUTES	1 MINUTE	5 SECONDS
	5 SECONDS	 DAYS	12 HOURS	2 HOURS	21 MINUTES	5 MINUTES	25 SECONDS
	30 SECONDS	 4 WEEKS	 3 DAYS	12 HOURS	2 HOURS	30 MINUTES	2 MINUTES
	1 MINUTE	 8 WEEKS	 6 DAYS	 1 DAY	4 HOURS	1 HOUR	5 MINUTES
	5 MINUTES	9 MONTHS	 4 WEEKS	 6 DAYS	21 HOURS	5 HOURS	25 MINUTES
	30 MINUTES		6 MONTHS	 5 WEEKS	 5 DAYS	 1 DAY	2 HOURS
	1 HOUR		10 MONTHS	2 MONTHS	 10 DAYS	 2 DAYS	5 HOURS
	6 HOURS				2 MONTHS	 2 WEEKS	 1 DAY
	 1 DAY					 8 WEEKS	 5 DAYS

- Slides: github.com/afermg/2024_07_JUMP_devtools
- Previous slides:
github.com/afermg/2024_04_hackathon_brainstorm