Introduction

- I am a Doctoral Researcher in Educational Technology at the University of Jvyäskylä, Finland
- My recent work has involved quite a bit of pipeline-oriented batch processing
  - For natural language processing
    - Preparing corpora for word-sense disambiguation
    - Preparing corpora for collecting frequency like measures
  - Integrating computer vision techniques for gesture researchers
- I will first talk a bit about Singularity, HPC, SLURM and Snakemake
  - Mini Snakemake tutorial
  - Some (hard-earned) tips for effective usage of these tools
  - And then show an essentialised versions of a video processing pipelines
- Talk & demo is a bit loosely planned
  - Decided to go for a slightly more focussed scope than the abstract
  - Please interrupt to ask questions
  - Correspondence welcome! (Zulip, e-mail frankie@robertson.name, GitHub issues)

Reproducibility / Repeatability

- Here I'm distinguishing reproducibility and repeatability
- Repeatable is
  - Running the exact same code (or as close as possible)
  - On the exact same data (or as close as possible)
  - To get the exact results (or as close as possible)
- Reproducible is
  - Being able to apply the description in the publication
  - To get similar results as the publication
  - Given (potentially) unlimited effort
    - i.e. might have to rewrite everything from scratch
- In a limited sense repeatability helps with reproducibility too
  - Maybe there are bits missed out or misleading in the publication
  - At least you have working code to reverse engineer

Getting your software on to HPC

- Quite a familiar topic
- Two options in the typical old-school approach to running software
- Admin installs the software
  - Might need to be a popular package
  - Might need to ask nicely
  - Duplication of effort
  - Not repeatable
- You compile it yourself
  - Specify an installation prefix
  - Struggle with library/include paths
  - Better document the process for repeatability (version, flags)
- Now can use a new package manager to install many dependencies together
  - E.g. Conda which can install a lot of stuff
  - Likely to be able to reproducible
  - All dependencies have to be a Conda package
  - Start hacking at your environment and repeatability goes down

Singularity

- Container runtime for HPC environments
- Has nice features
  - Image format is a single file
    - Can be archived / distributed for repeatability
    - Can move it to fast disk space
  - Good default binds of home and working directory
    - Typically means are your data is 'just there' by default
    - Convenient blurred host/container distinction
- Helps with repeatability
  - Not necessarily a panacea
  - Pragmatically some things like data might have to be left out of image
  - Could be problems as type of available computing resources change
    - i.e. pragmatically portability could be part of reproducibility

Snakemake

- Make-like tool
- Written with the context of Bioinformatics
  - File-oriented approach
  - Easy to write scatter/gather or map/reduce
  - Not tied to Bioinformatics
- A bit hairy
  - Typically we are mixing three languages in a `Snakefile`
  - Python, Snakemake & Bash
- But also has a lot of pleasant features
  - Partial runs
  - Implicit coarse-grain parallelism
  - Combine multiple languages by using neutral file formats
  - Scales down to laptop usage
  - And up to HPC usage
- A good choice if lots of serialising/deserialising is okay

SLURM

- Batch job scheduler
- Most commonly used by writing a special SLURM script
  - With SLURM directives specifying the resources you want at the top
  - Followed by a bash script
  - Submitted with the `sbatch` command
- Higher level tools might support it by templating
  - Command usage
  - Script usage
- SLURM ain't broke
  - Has nice features like starting many similar jobs using job arrays

Singularity tips for building containers

1. Use `Dockerfiles` for building & distribute Docker/OCI images in development
   - Singularity has its own container recipe format "Singularity definition files"
   - However, this ties you to Singularity
   - If you use Dockerfile + OCI containers you can use free building and hosting from GitHub actions, GitLab CI, ...
   - Then convert to SIF for running on Singularity and archiving
2. Although you can run `singularity run docker://` – don't
   - Instead run `singularity pull` and then later `singularity run`
   - Reason: converting is slow
   - You can easily end up doing the conversion on multiple nodes

Singularity tips for running containers

1. Find somewhere big for your `/.singularity` directory
   - Everything gets cached here – can get big
   - Many HPC environments have relatively small quotas for home directories
   - Find the biggest shared directory you can
   - Then move your `/.singularity` directory there and symlink to it
2. Cut down on rebuilds using binds
   - Building Singularity images and converting them is slow
   - You can also use tools like `lsyncd` to

Example: Heterogeneous video processing pipeline

- This is an example based on work I have done with gesture researchers (Red Hen lab, IMCC, University of Oxford)
- Some steps need GPU
  - Like identifying keypoints
  - And embedding faces
- Some steps can require lots of memory
  - Like clustering many face embeddings

Mini-Snakemake tutorial: top level

```
from os.path import join as pjoin
WORK = config.get("WORK", "work")

VIDEO_IDS = [
    "9U4Ha9HQvMo",
    "xTXz_4u-4mc",
    "CJkWS4t4l0k",
    "Q_OIXfkXEj0",
    "qXD9HnrNrvk",
    "XUT8ec24anM",
    "BOyebcrVWb4",
    "uFpK_r-jEXg",
]
```

- Normal Python code can go at the top level
- Some global variables available from Snakemake

Mini-Snakemake tutorial: rules

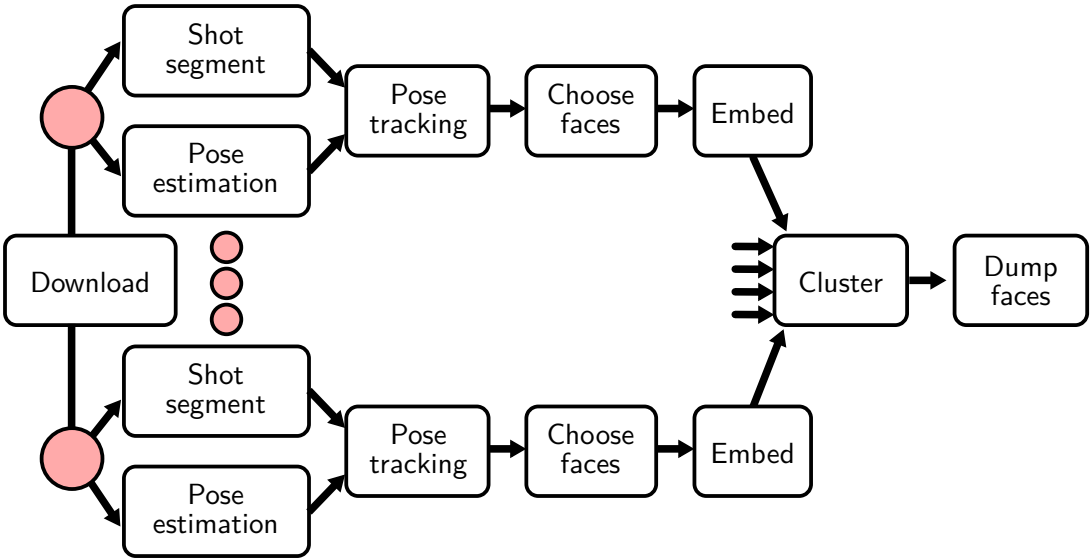
```
rule download_videos:
    output:
        [pjoin(WORK, video_id + ".mp4") for video_id in VIDEO_IDS]
    shell:
        " youtube-dl " +
        " -o '%(id)s.%(ext)s' " +
        " ".join((
            f"'https://www.youtube.com/watch?v={video_id}'"
            for video_id in VIDEO_IDS
        ))
```

- A bit like a Makefile
- But rules can have multiple outputs, comments
- Mix shell with Python with string interpolation

More Snakemake resources

- Official docs: `https://snakemake.readthedocs.io/en/stable/`
- Software carpentry course
  `https://carpentries-incubator.github.io/workflows-snakemake/`
- Some NLP/corpus processing Snakefiles found in the wild:
  - `https://github.com/LuminosoInsight/exquisite-corpus/blob/master/Snakefile`
  - `https://github.com/rspeer/spacious_corpus/blob/main/spacious_corpus/config/Snakefile`

Overview of video corpus processing pipeline



Running a Snakemake pipeline on a SLURM cluster

- Snakemake supports being passed a queue submission commmand
- Also options for cancellation command / status command
- This combination of options can be packaged together with a system called profiles
- SLURM-profile: `https://github.com/Snakemake-Profiles/slurm`
- Not a plugin system — not a Python library

How to containerise Snakemake workflows?

- Now we have Snakemake as a dependency of our project
- So let's make sure to include it in our container
- Problem: How can we use SLURM from our container
- My solution: `singslurm2` & `singreqrun`
- Alternative solution: Bind `sbatch` into container
  - But now we also need to bind various libraries
  - And then we find out there are different versions of libc
  - Bad

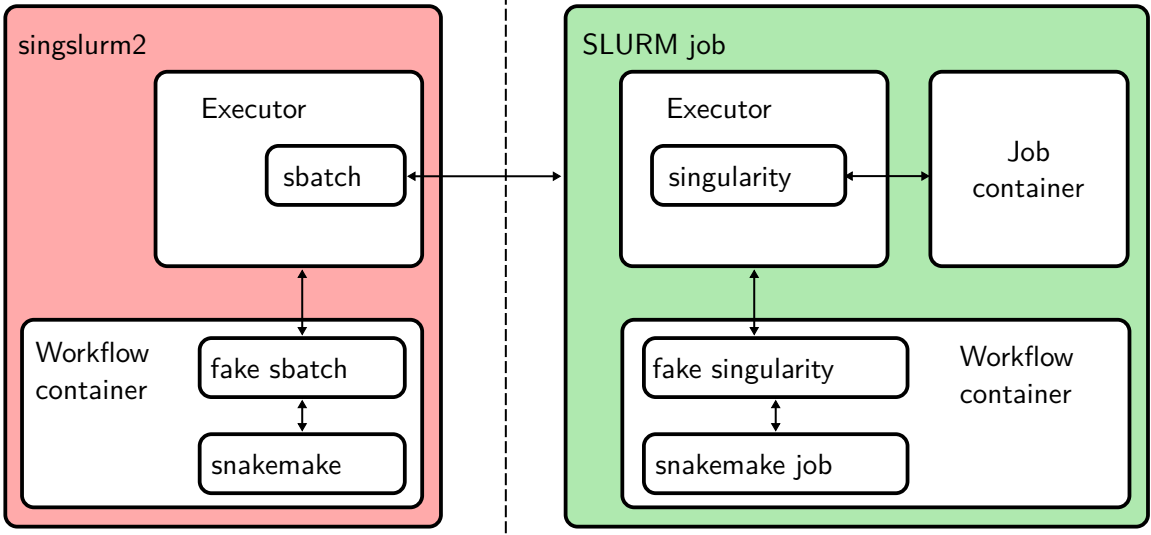
singslurm2/singreqrun

- `singreqrun`: Consists of a client and a server
  - Allows code run inside Singularity container to request a command is run on host
  - Client is statically built C program
  - It can be bound into the container as `/usr/bin/sbatch`
  - Server/executor is a batch script run on the host
- `singslurm2`: Convenience script
  - Arranges all the correct binds
  - Arranges for SLURM-profile
  - Arranges Singularity containers to be started

Singularity-in-Singularity

- So far we have been thinking in terms of monolithic containers
  - Which I would recommend by default
  - More convenient
  - Less SIF files to archive
- But sometimes we need different versions of libraries for different pipeline steps
- Snakemake has its own Singularity support
  - However we have the same problem again
  - Singularity is on the host, not in the container
  - But we can reuse singreqrun!

singslurm2 with Singularity-in-Singularity



End

- Open to questions
- Also very interested to hear about alternative approaches
  - Tradeoffs w.r.t. complexity
  - Perspectives on repeatability / reproducibility