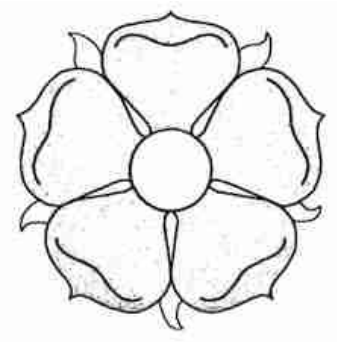
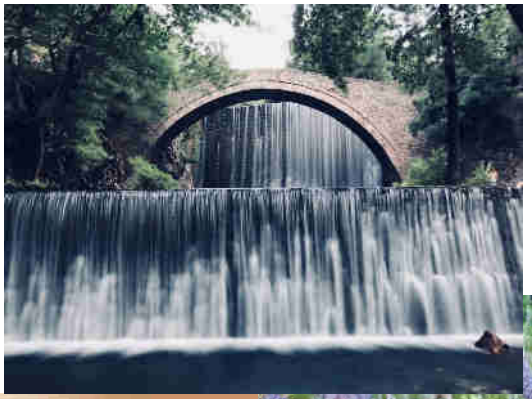
Best Practices in Handling Genomic Data

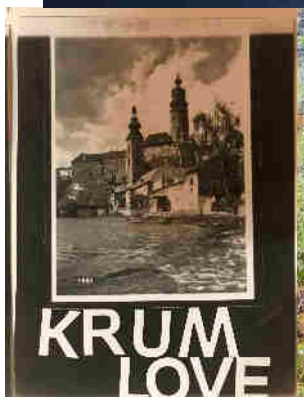
Dag Ahrén



Interests outside of work

Photography, Food & Family





My Background

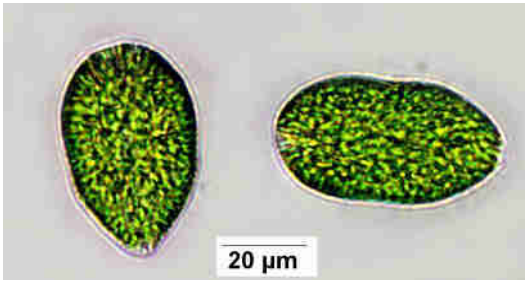
- Biologist that became a Bioinformatician
- Genomics research since before NGS

EMBL-EBI

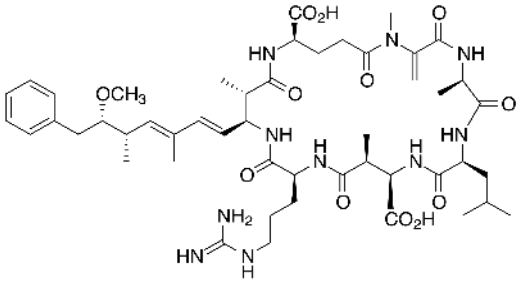
CERTH
CENTRE FOR
RESEARCH & TECHNOLOGY
HELLAS

NBIS

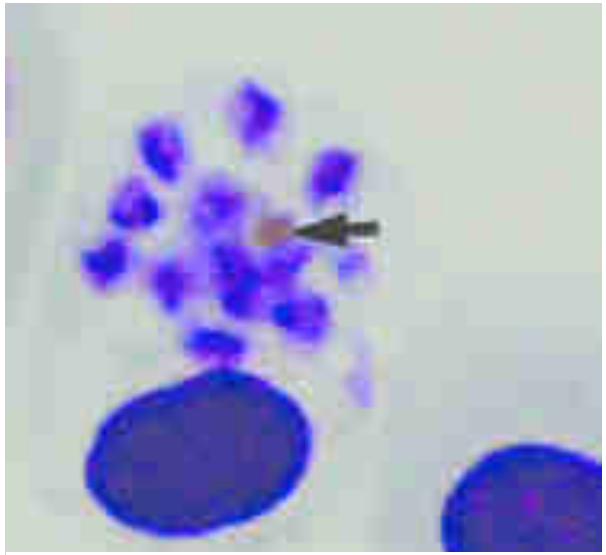
My Research Interests



Single cell population genomics
Gonyostomum semen



Toxin gene clusters in *Microcystis*



Avian malaria host parasite interactions

Adaptation to radiation in black yeasts

Methanogens and methanotrophs in SubArctic and Arctic Ecosystems


NBIS

~100 staff at six different sites across Sweden with expertise in many different omics-related areas

Let’s start cooking!

Ingredients

- Reproducible research
- Tools for reproducibility
- Special requests
- Lab

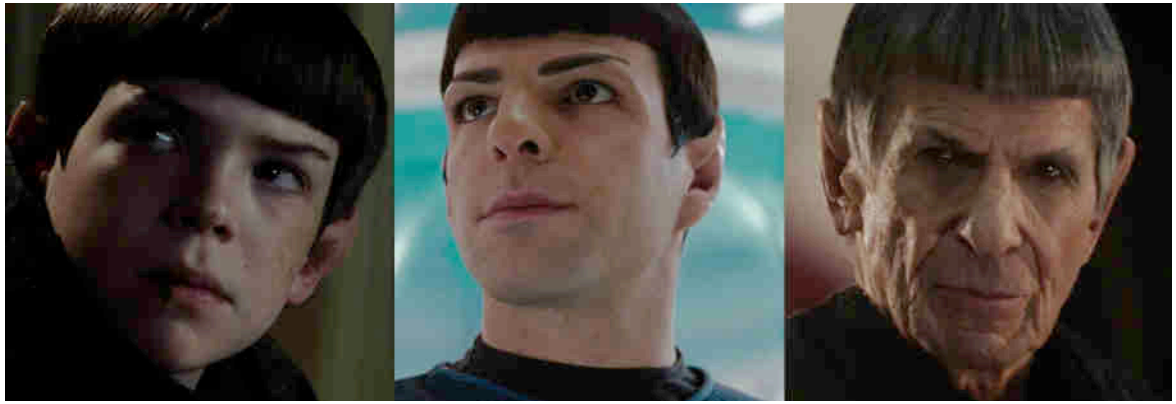
Reproducible research

Covered excellently by Chris Wheat

Why important?

- To be able to rerun analyses
- Assist when publishing
- Increase the usability of the data and results

Your future self will thank you!!!



My thoughts…

- Set realistic goals
- Share and help each other & give positive feedback (e.g. github repo)
- My goal today is to make all of this a little bit easier!


GitHub

Technical bits

Backup

- Get an off-site backup for your raw data as soon as it arrives
- Make sure metadata is backed up with the raw data
- Once initial QC is complete, submit raw data to a data repository (with embargo)
- Get frequent backups of scripts
- Backup intermediate results

> rsync -Pa

Let’s make life easier

…hopefully

TMUX

Terminal Multiplexer

- Split views in the same terminal window
- Reattach to a previous tmux session

TMUX

> tmux new -s genomeAssembly
> tmux ls
> tmux attach -t 0.

Basic commands: *ctrl-b %* Split into two vertical panes

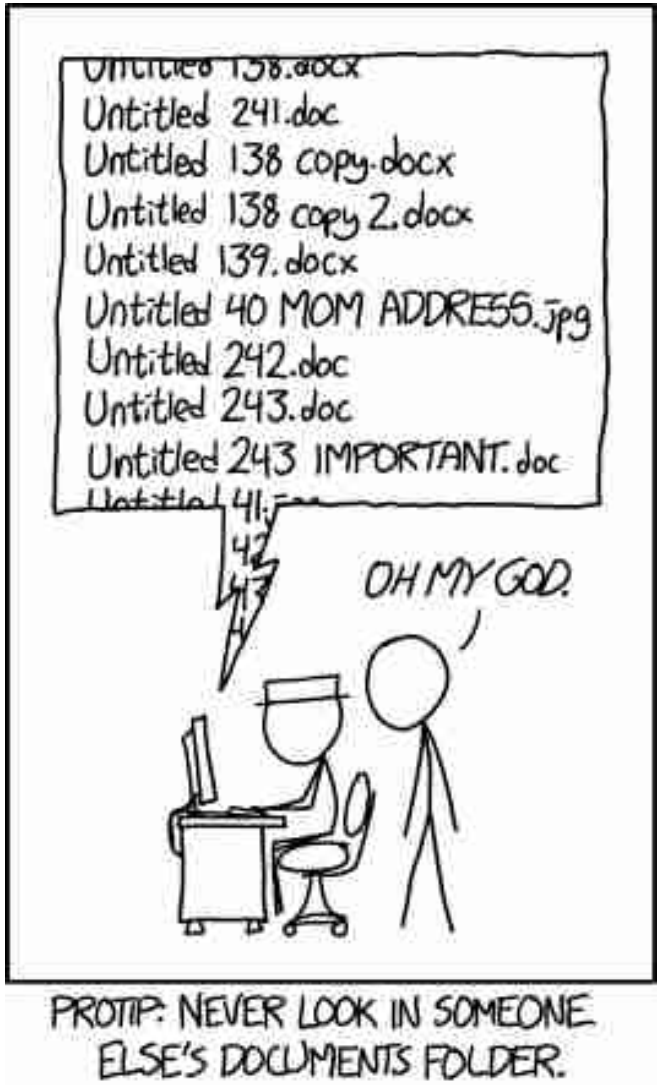
ctrl-b “ Split into horizontal panes

ctrl-b d Detach from tmux session

File names

- Use extentions to guide you (.txt .csv .fastq)
- Name files so that it is easy to understand and describe where it comes from (AT1_R1_trimmed.fq)
- Avoid any label that implies order relative to other files (Final1.txt UltraFinal.txt This_is_my_Final_Final_version2.txt)

File names


Untitled 241.doc
Untitled 138 copy.docx
Untitled 138 copy 2.docx
Untitled 139.docx
Untitled 40 MOM ADDRESS.jpg
Untitled 242.doc
Untitled 243.doc
Untitled 243 IMPORTANT.doc
OH MY GOD.
PROTIP: NEVER LOOK IN SOMEONE ELSE'S DOCUMENTS FOLDER.

My take on a strategy

(but with support from literature)

- Totally fine if you have another strategy…

… but remember that chaos does not count as a strategy!! . . .

Project

Data

Read-only, raw data and meta data
> chmod -R Data

This is an exact **COPY** of the data at the start of the project
Note: Keep a backup at a separate location
Submit raw data to public repository early, with embargo

Docs

Put documentation (e.g R markdown, Notes etc)

Scripts

Scripts, such as sbatch, bash, R scripts etc

Progs

Store software installed manually Keep a record of software & versions

Analysis

Make a separate folder for each of the steps in the analysis I like to number them to get a nice order 1.raw_data is a symbolic link: >ln -s Data/SRRZ123447_R1.fastq sampleA_R1.fastq

Work reproducibly

1. Track how results were produced (quarto, markdown, juypiter notebook)
2. Avoid manual data manipulation
3. Archive/document all external software used. Versions!! (conda, R yml files)
4. Version control custom scripts (conda, markdown git/github)
5. Make it all available! (github)

So you have a file structure

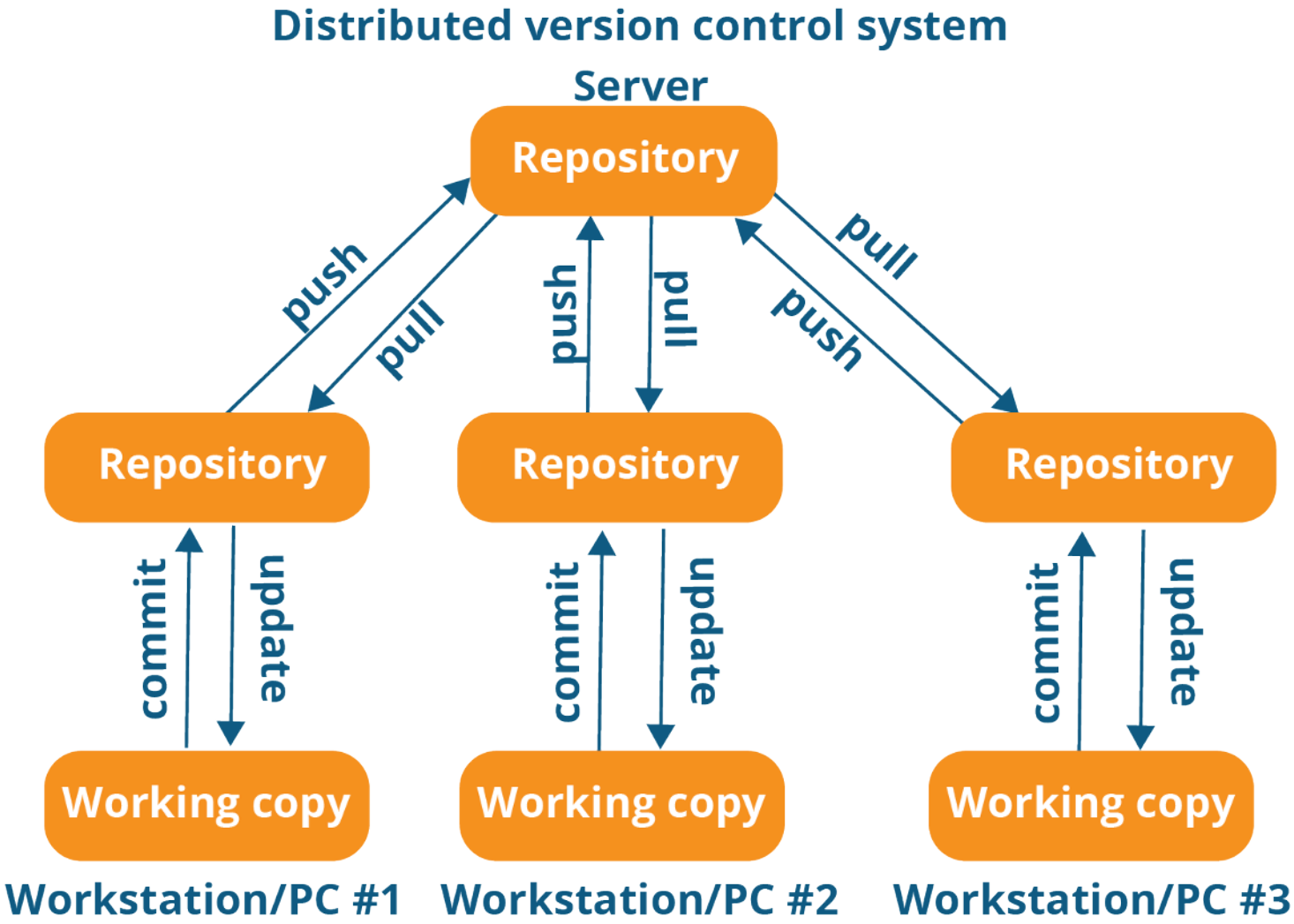
Version control

Git & Github

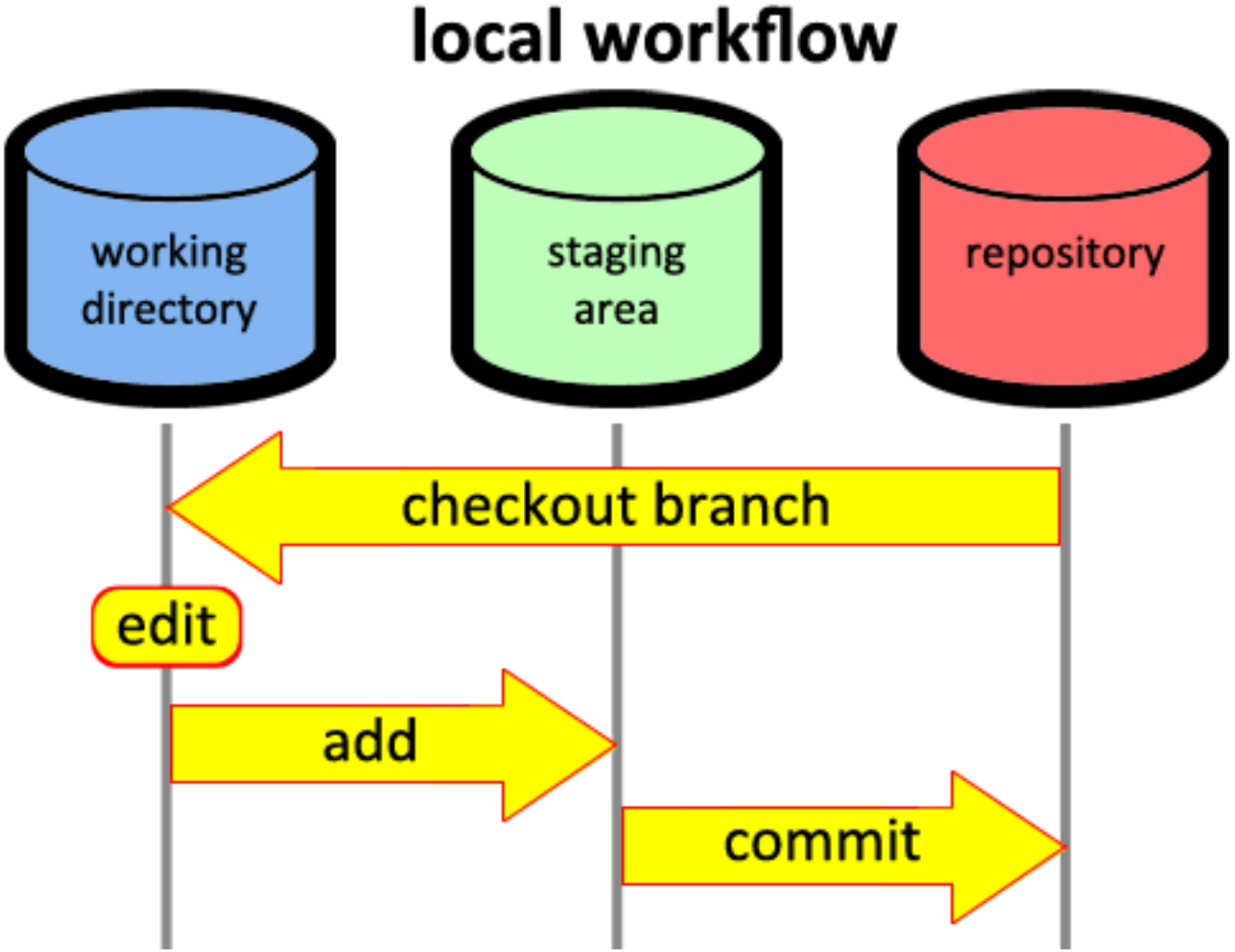
What is Git?
Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

What is GitHub?
GitHub is a web page were git repositories can be shared. It is a essentially social platform for code. Good for most things that fit with Git.

Git is distributed



Basic git workflow



Shortlist of the most useful terms in git

status
stage (add)
commit
push
pull
clone
branch

Recommendations when committing to the repository

- Commit on a regular basis, ideally when one set of work has been performed and tested.
- Write short descriptive comments to each commit

Conda

Package and environment manager

- Install software with dependencies
- Avoid dependency issues
- Save the software versions and dependencies in a file

Conda commands

```
conda create -n project_A
conda env list
conda activate project_A
conda info –envs
conda install -c bioconda sra-tools
```

Save the environment software and dependencies to a file

>conda env export > project_A_condaenv.yml

Other tools for reproducible science

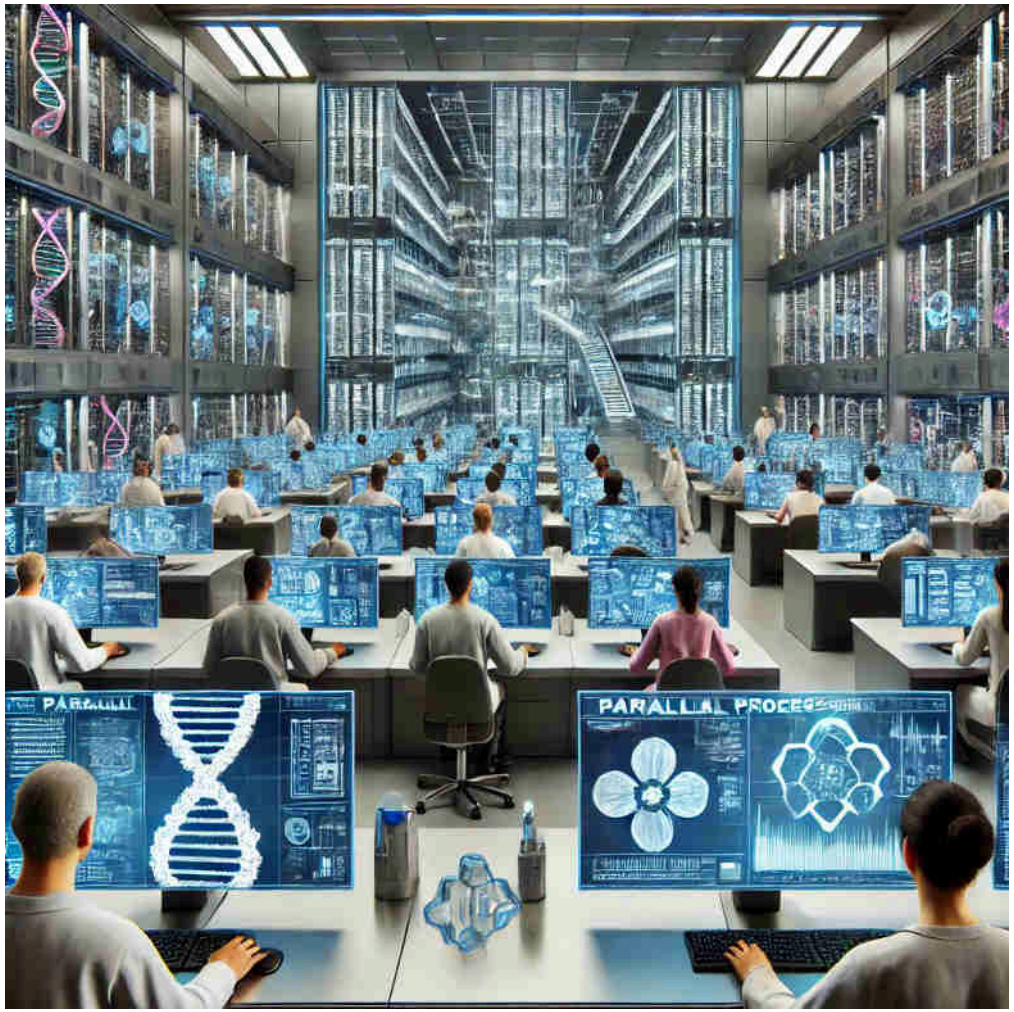
- Workflows such as Snakemake & Nextflow
- Containers Docker & Apptainer

Take home messages

Do not try to do all at once.

Start with file structure and backup.
then consider more advanced steps such at git and conda Set goals that are realistic

Parallellization



Why is Parallelization Important?

- **Data Volume**: The sheer size of bioinformatics datasets, such as genomic sequences, requires robust computational approaches.
- **Complexity**: Many bioinformatics algorithms involve complex calculations that can benefit from parallel execution.
- **Time**: In time-sensitive research, reducing computational time can accelerate discovery and the application of findings.

Approaches to Parallelization

Multithreading: Utilizing multiple threads within a single processor to execute multiple tasks concurrently.

Distributed Computing: Spreading tasks across multiple compute nodes in a cluster or cloud environment.

GPU Acceleration: Using Graphics Processing Units (GPUs) for their parallel processing capabilities with large numbers of cores suited for certain types of calculations.

Not all software can be efficently parallelized

E.g Genome assembly Check if multithreading is an option

Tools & Libraries

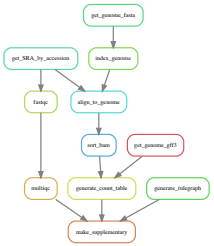
- GNU parallel
- MPI (Message Passing Interface)
- OpenMP (Open Multi-Processing)
- Bioconductor packages (e.g., BiocParallel)

Pick your poison

Putting it all together

1. Create a new git repository for the project (e,g, GitHub)
2. Add a README file which should contain the required information on how to run the project
3. Create a Conda environment.yml file with the required dependencies 4 Create a R Markdown or Jupyter notebook to run your code
4. Alternatively, create a Snakefile to run your code as a workflow and use a config.yml file to add settings to the workflow
5. Use git to continuously commit changes to the repository
6. Possibly make a Docker or Singularity image for your project

Best Practices Lab



Lab on Git and Conda

NBIS Data management & Reproducibility courses

Setup

git clone https://github.com/NBISweden/workshop-reproducible-research.git

> Avoid creating a repo inside another repo

Thanks

I look forward to talk to you about:

- Reproducible research
- Tools that was not mentioned
- Work-Life balance
- Life in Sweden/UK/Greece

… and Food!