

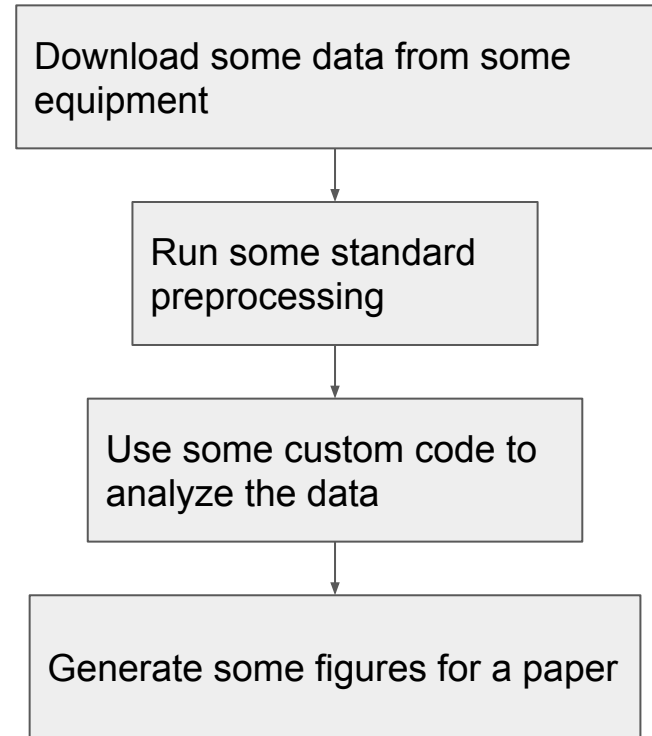
Automating your analysis code with workflows

Tristan Kuehn, Jason Kai, Ali Khan

Western University, London, Ontario

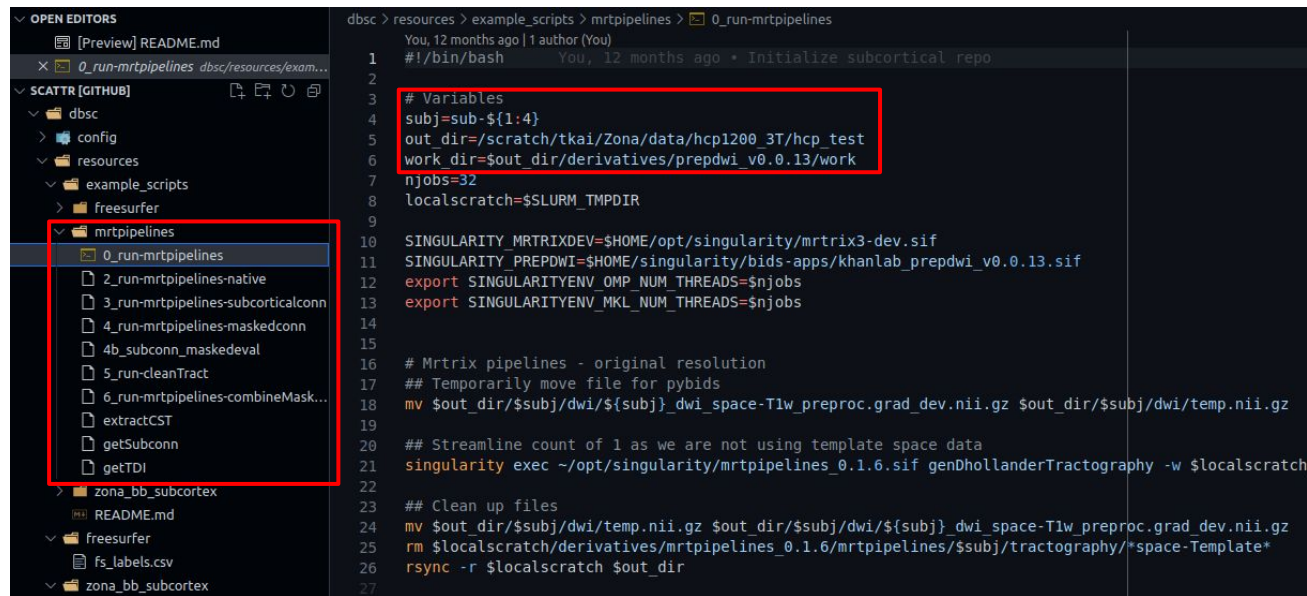
What is a workflow?

- Working with data always means using some kind of workflow
- Often the workflow isn't formalized
 - Just running some commands ad-hoc
- Workflow managers are tools for formalizing a workflow



Why workflow management?

- Modify or repeat processing / analysis
- Ease of changing parameters (e.g. file paths)
- Simplify managing complex workflows
- Automatically run workflows



```
dbsc > resources > example_scripts > mrtpipelines > 0_run-mrtpipelines
You, 12 months ago | 1 author (You)
1 #!/bin/bash
2
3 # Variables
4 subj=sub-${1:4}
5 out_dir=/scratch/tkai/Zona/data/hcp1200_3T/hcp_test
6 work_dir=$out_dir/derivatives/prepdwi_v0.0.13/work
7 njobs=32
8 localscratch=$SLURM_TMPDIR
9
10 SINGULARITY_MRTRIXDEV=$HOME/opt/singularity/mrtrix3-dev.sif
11 SINGULARITY_PREPDWI=$HOME/singularity/bids-apps/khanlab_prepdwi_v0.0.13.sif
12 export SINGULARITYENV_OMP_NUM_THREADS=$njobs
13 export SINGULARITYENV_MKL_NUM_THREADS=$njobs
14
15
16 # Mrtrix pipelines - original resolution
17 ## Temporarily move file for pybids
18 mv $out_dir/$subj/dwi/${subj}_dwi_space-T1w_preproc.grad_dev.nii.gz $out_dir/$subj/dwi/temp.nii.gz
19
20 ## Streamline count of 1 as we are not using template space data
21 singularity exec ~/opt/singularity/mrtpipelines_0.1.6.sif genDhollanderTractography -w $localscratch
22
23 ## Clean up files
24 mv $out_dir/$subj/dwi/temp.nii.gz $out_dir/$subj/dwi/${subj}_dwi_space-T1w_preproc.grad_dev.nii.gz
25 rm $localscratch/derivatives/mrtpipelines_0.1.6/mrtpipelines/$subj/tractography/*space-Template*
26 rsync -r $localscratch $out_dir
27
```


Things that are helpful (but not 100% required) to know

- How to install a Python package
 - Pip (<https://docs.python.org/3/installing/index.html>)
 - Conda (<https://docs.conda.io/en/latest/>)
- Some Python scripting
 - <https://docs.python.org/3/tutorial/index.html>
 - This isn't just for Python workflows but Snakemake runs on Python
- Some bash scripting
 - <https://learnxinyminutes.com/docs/bash/>
- Don't always need to do all this by scratch, don't feel intimidated by this list of things
 - Relatively easy to start with small chunks of what you're doing

CBS Server Locations to follow along

- We'll have live examples of the features we talk about in the slides
 - If you're interested in following along/seeing more context
 - Not mandatory, we'll have a live demo after
- For this tutorial:
 - Data: `/scratch/tkai/data`
 - Virtual Environment: `/scratch/tkai/snakemakeenv`
 - Workflows: `/scratch/tkai/example-snakemake`
 - Note: This stuff will be gone within 2 weeks (or a bit less)
- Will work faster if you copy data (and workflows) to localscratch:
 - `cp -r /scratch/tkai/data/t1w-cannabis /localscratch`
- In general, install snakemake with pip: `pip install snakemake`
- The Snakemake docs are a very good intro and reference:
<https://snakemake.readthedocs.io>

Snakemake intro

Example problem

- We've got this OpenNeuro dataset of baseline and 3-year follow-up scans of cannabis users: <https://openneuro.org/datasets/ds000174/versions/1.0.1>
- Question: Is there a difference in mean brain volume across the two sessions?
 - No claims about the scientific validity of this question... Just a toy problem.
- Need to calculate the brain volume for each scan, group them, and summarize
- Kind of a pain to do by hand

Interactively running tools

```
$ bet data/sub-314/ses-BL/anat/sub-314_ses-BL_T1w.nii.gz  
out/bet/sub-314/ses-BL/anat/sub-314_ses-BL_desc-brain_T1w.nii.gz
```

- For small one-off tasks, this is okay
- Later, it might be hard to remember how you generated the output file
- You may also forget which input file was used
- As soon as you're running multiple scripts sequentially this can become untenable and hard to reproduce

Simple bash script

```
#!/bin/bash
```

```
for session in BL FU; do
    bet data/sub-314/ses-${session}/anat/sub-314_ses-${session}_T1w.nii.gz
    out/bet/sub-314/ses-${session}/anat/sub-314_ses-${session}_desc-brain_T1w.nii.gz
done
```

```
$ ./myscript.sh
```

- Now the command, input file, and output file are all recorded somewhere.
- But: What if I want to run this code on another file?
 - Need to copy and paste the line and change the details maybe.
- What if I want to run this code on a dataset with 100 subjects/sessions?
- Then, what if I want to change a detail of the command?

Snakemake

- This kind of scenario is where a workflow manager becomes very helpful
- Key idea: define a workflow in terms of rules for producing files
- Then you can ask Snakemake to produce a file, and it will look through all the rules you've defined to figure out how to do it (and fail if it can't).

```
rule bet:
  input:
    t1w="data/sub-314/ses-BL/anat/sub-314_ses-BL_T1w.nii.gz"
  output:
    brain="out/bet/sub-314/ses-BL/anat/sub-314_ses-BL_desc-brain_T1w.nii.gz"
  shell:
    "bet {input.t1w} {output.brain}"
```

```
$ snakemake out/bet/sub-314/ses-BL/anat/sub-314_ses-BL_desc-brain_T1w.nii.gz -c1
```

Adding wildcards

```
rule bet:
    input:
        t1w="data/sub-{subject}/ses-{session}/anat/sub-{subject}_ses-{session}_T1w.nii.gz"
    output:
        brain=(
            "out/bet/sub-{subject}/ses-{session}/anat/sub-{subject}_ses-{session}_desc-brain_T1w
            .nii.gz"
        )
    shell:
        "bet {input.t1w} {output.brain}"
```

```
$ snakemake out/bet/sub-314/ses-BL/anat/sub-314_ses-BL_desc-brain_T1w.nii.gz -c1
```

- Snakemake finds the needed wildcard(s) from the output you request
- The file we ask for matches `output` if `subject` and `session` are `101` and `BL` respectively - those values propagate to the shell command Snakemake runs.

Generating params with a function

```
rule bet:
    input:
        t1w="data/sub- $\{subject\}$ /ses- $\{session\}$ /anat/sub- $\{subject\}$ _ses- $\{session\}$ _T1w.nii.gz"
    output:
        brain=(
            "out/bet/sub- $\{subject\}$ /ses- $\{session\}$ /anat/sub- $\{subject\}$ _ses- $\{session\}$ _desc-brainthre
            shold $\{threshold\}$ _T1w.nii.gz"
        )
    params:
        threshold=lambda wildcards: f"0. $\{wildcards.threshold\}$ "
    shell:
        "bet  $\{input.t1w\}$   $\{output.brain\}$  -f  $\{params.threshold\}$ "
```

```
$ snakemake out/bet/sub-314/ses-BL/anat/sub-314_ses-BL_desc-brainthreshold75_T1w.nii.gz -c1
```

Chaining rules

- If the rule that produces the file you ask for doesn't have the input it needs, Snakemake will check if any other rules can produce that input file.
- This is a really powerful feature to grasp if you want to put together a more complex workflow.

```
rule bias_field_correction:
    input:
        brain=rules.bet.output.brain,
    output:
        corrected=(
"out/biasfieldcorrection/sub- $\{subject\}$ /ses- $\{session\}$ /anat/sub- $\{subject\}$ _ses- $\{session\}$ _desc-correctedbr
ainthreshold $\{threshold\}$ _T1w.nii.gz"
        )
    shell:
        "N4BiasFieldCorrection -d 3 -i {input.brain} -o {output.corrected}"
```

- If the inputs don't exist yet, Snakemake will figure out that it needs to run "bet" (once with each property) first.

Target rule

```
rule all:
    input:
        expand(
            rules.bias_field_correction.output,
            subject=[314, 316],
            session=["BL", "FU"],
            threshold=[5],
        )
    default_target: True
```

```
$ snakemake -c1
```

- “Expand” will look for every combination of the given wildcards (by default)
- It will apply the combinations to the first argument
 - Here, output of the `bias_field_correction` rule

Config file

```
subjects:
  - 314
  - 316

sessions:
  - BL
  - FU

threshold: 50
```

```
configfile: "config/config.yaml"

rule all:
  input:
    expand(
      rules.bias_field_corection.output,
      subject=config["subjects"],
      session=config["sessions"],
      threshold=config["threshold"]
    ),
```

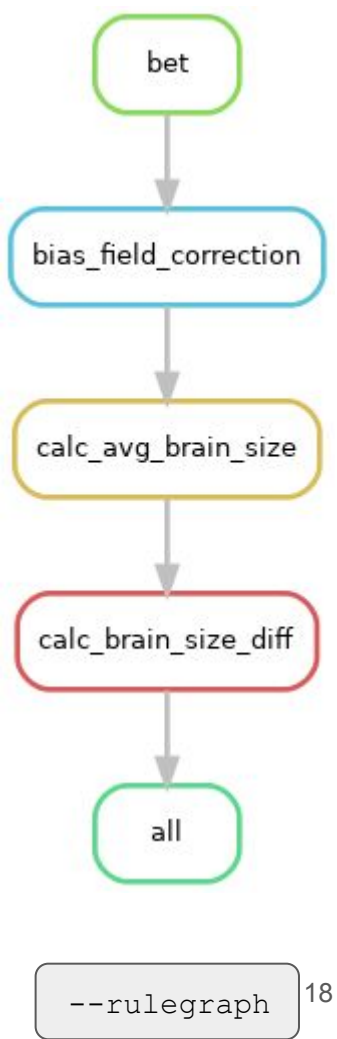
```
$ snakemake -c1
```

- Config variables in “config/config.yaml” are accessible from the workflow.

Visualize the DAG

- You can look at a graph of your workflow using Snakemake
- This can help make sense of a large, confusing workflow

```
$ snakemake --rulegraph | dot -Tpdf > rule_dag.pdf  
$ snakemake --dag | dot -Tpdf > dag.pdf
```



Nice Snakemake features

- Modularization (can share rules between workflows)
- Can define a docker (or singularity container) to run per rule.
 - Can also define a conda environment if that's your preference.
- Self-contained HTML reports
- Project template

Snakebids teaser

```
From snakebids import bids

rule bet:
    input:
        t1w=bids(
            root="data",
            subject="{subject}",
            session="{session}",
            datatype="anat",
            suffix="T1w.nii.gz"
        )
    output:
        brain=bids(
            root="out/bet",
            subject="{subject}",
            session="{session}",
            datatype="anat",
            desc="brain",
            suffix="T1w.nii.gz"
        )
    shell:
        "bet {input} {output}"
```

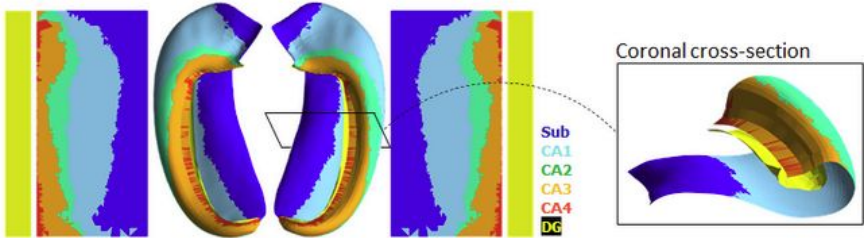
Snakebids in the wild

[/ Hippunfold](#) [Edit on GitHub](#)

docs **passing** docker pulls 1.4k version v1.2.1

Hippunfold

This tool aims to automatically model the topological folding structure of the human hippocampus, and computationally unfold the hippocampus to segment subfields and generate hippocampal and dentate gyrus surfaces.



The image displays the Hippunfold tool's output. On the left, there are two vertical heatmaps showing the segmentation of the hippocampus. In the center, a 3D model of the hippocampus is shown, with its subfields color-coded. On the right, a coronal cross-section of the hippocampus is shown, with a legend indicating the subfields: Sub (blue), CA1 (green), CA2 (yellow), CA3 (orange), CA4 (red), and DG (black).

Structural Connectivity Applied To Targeted Regions (SCATTR)

docs **passing** version v0.1.2 python 3.8 | 3.9 | 3.10 [Lint and test workflow](#) **passing**

docker pulls 3 DOI 10.5281/zenodo.7636506

[» funcmasker-flex](#)

funcmasker-flex

Brain masking app using Unet for fetal bold mri

[» snakedwi](#)

[Edit on GitHub](#)

snakedwi

BIDS app and snakemake workflow for dwi pre-processing

Thank you

- Help contribute to the development!
<https://github.com/akhanf/snakebids>
- What features would you like to see?
<https://github.com/akhanf/snakebids/issues>



Tristan Kuehn, Peter Van Dyken, Jason Kai, Ali Khan, github-actions bot, and you?

Demo/questions?

CBS Server Locations to follow along

- We'll have live examples of the features we talk about in the slides
 - If you're interested in following along/seeing more context
 - Not mandatory, we'll have a live demo after
- For this tutorial:
 - Data: `/scratch/tkai/data`
 - Virtual Environment: `/scratch/tkai/snakemakeenv`
 - Workflows: `/scratch/tkai/example-snakemake`
- In general, install snakemake with pip: `pip install snakemake`
- Will work faster if you copy data (and workflows) to localscratch:
 - `cp -r /scratch/tkai/data/t1w-cannabis /localscratch`
- In general, install snakemake with pip: `pip install snakemake`
- The Snakemake docs are a very good intro and reference:
<https://snakemake.readthedocs.io>