The trouble with scripts

Shell scripts are useful for automation
But when not to use them?

Will Furnass
Research Platforms Engineer
IT Services

Scripts are great, right?

Useful in drive for more automation and reproducible research

Esp. with HPC, cloud

What meant by defining research workflows using shell scripting?

- Define the bulk of what you want to do programatically
- not type interactively
- Run one or more programs
- With particular parameters
- In a particular environment (access to storage, network, environment variables)
- To generate output

Example workflow: HPC job submission script

```
#!/bin/bash
#$ -l rmem=4G

module load R/4.0
module load Python/3.9

cp -ra ~/$PROJECT/data /fastdata/te1st/
python ~/$PROJECT/train.py -i /fastdata/te1st/data -o ~/$PROJECT/trained/
python ~/$PROJECT/test.py ~/proj1/trained/
R ~/$PROJECT/plots.py ~/proj1/trained/
```

Issue 1: how can that fail?

- Modules don't exist
- Not asked for enough memory
- Input dir doesn't exist
- A command might error but subsequent commands might still run!
- $PROJECT not defined
- $PROJECT has spaces in!
- …

In general: concise and convenient but many more silent failure modes than e.g. MATLAB

Command might error but subsequent commands might still run

```
ls ~/.bashrc
ls ~/idontexist
ls ~/.condarc  # will still run!
```

Tips:

- Well-behaved command should return a (hidden) 'exit code' which is 0 if all okay
- `commanda && commandb` -> only run `commandb` if `commanda` succeeds
- `commanda || commandb` -> only run `commandb` if `commanda` succeeds
- `set -e` -> exit the script if a command returns a non-zero exit code

Undefined variables

Dereferencing shell variables results in the empty string if previously not explicitly assigned to:

```
$ echo ~/.conda/$NOTDEFINED/envs
/home/will/.conda//envs
```

Tips:

- `set -u` -> exit the script if reach an undefined variable
- `${SOMEVAR-42}` -> default to '42' if SOMEVAR is undefined

Spaces in names / variables

With `~/$PROJECT/data`

- If `PROJECT` is defined as e.g. `my project`
- Then e.g. `ls $PROJECT` will be interpreted as `ls my project!`

Dealing with spaces in variables and in file/path names is hard

Tips:

- Avoid putting spaces in file/directory names!
- Safest way to reference variables: `somecommand "some text ${VARIABLE_NAME} more text"`
- Use single and double quotes liberally

Issue 2: Fault tolerance (idempotence?)

- What if a script/job fails part way through?
  - Can re-run without having to edit script logic?
  - Or without deleting some output files?
- What if want script to be able to 'resume' if re-run?
  - Only generate output files not already generated?
  - Or if existing output files are older than input files and/or script?
- Adding such logic to shell scripts can be tricky

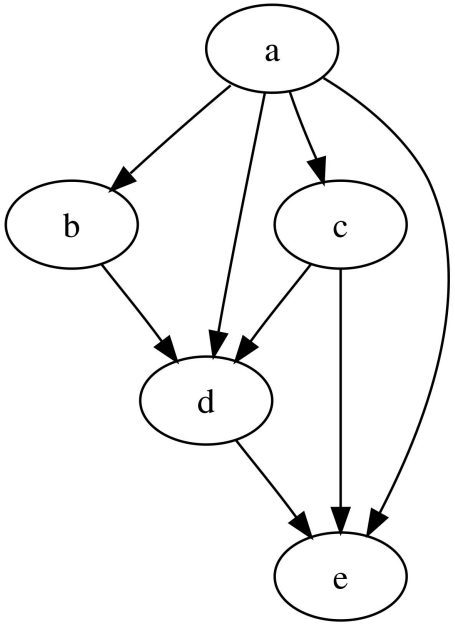
Issue 3: Portability

- Given an existing shell script what aspects are specific to:
  - You?
  - Your group/dept/institution?
  - The system you're running it on?
  - Dataset(s)?
  - Date/time?
  - Parameters?
- Making scripts (inc shell scripts) and configs portable and reusable between researchers/systems/projects is tricky!

Issue 4: Multiple inputs and/or job steps

On HPC / cloud:

- What if you have lots of input files/parameter sets?
- And/or lots of job steps?
- One job script with one or more loops?
  - Not efficient if unnecessarily serial
- One job script with internal parallelism?
  - Not efficient if different inputs / steps need different resources
  - Error handling might be tricky
- Job scripts that submit other jobs?
  - Can efficiently tune resource requirements per 'sub'-job and
  - could execute complex network of tasks but
  - robust error handling is even harder!



Vector version of Image:Tred-G.png, Public domain, via Wikimedia Commons

Summary

- Shell scripts useful for defining simple workflows
- But writing and testing robust complex shell scripts will always be hard
- Even if using a text editor (or shellcheck) that can advice on bad practices!
- Knowing where shell scripting is appropriate is important
- Better tools/processes exist for defining and running more complex workflows…