Winter School in Marine Environmental Prediction

Code Automation
6 Mar 2017

Doug Latornell
dlatornell@eoas.ubc.ca

UBC

UQAR

MEOPAR

Links

- Slide Deck:

  https://bitbucket.org/douglatornell/uqar-winter-school/downloads/slides.pdf
- Repository:

  https://bitbucket.org/douglatornell/uqar-winter-school

Session Plan

- Definition
- Software Tools
- Organizing Your Modeling Research Life
- Exercise #1
- Writing Automation Tools
- Exercise #2
- NEMO command processor

Code Automation - Definition

Use software tools to automate tasks that are:

- tedious
- error-prone
- complicated
- repeated often

To document the story of your research

To make it more likely to be well organized, reproducible, traceable

Code Automation - Definition

Use software tools to automate tasks that are:

- tedious
- error-prone
- complicated
- repeated often

To document the story of your research

To make it more likely to be well organized, reproducible, traceable

Your most important collaborators are your past and future selves

Be kind to them!

Software Tools

- Version Control
- Backups
- Model Runs Management
- Results Visualization

Version Control

The most important automation tool you'll learn about this week!

Version Control All The Things! (Almost...)

- Code
- Manuscripts of proposals, papers
- Research notes, work log
- Course work assignments
- Pre- and Post-processing techniques and code
- ...

Version Control

If you are not using version control already, start! Today!

- Find a local expert and use the version control tool that they use if it is:
  - Git
  - Mercurial (hg)
  - Subversion (maybe)
- But probably not if they have a grey beard like mine (or even if they don't) and say "We don't use *traditional* version control, *per se*", or "I have this system of shell scripts that I wrote that does that..."

Assignment

- Create a repository for notes from this week
- Add and commit something about each session
- Bonus points for putting it on GitLab or another web hosting service
- More bonus points for making it public and sharing the URL

Version Control

- Commit early, commit often
- Version control can be used as a time machine, but you can’t revert back to something that you didn’t commit
- Use your commit messages to tell a story to your future self

What Not to Version Control

Ephemeral files that are products of compilation, LaTeX, etc.

- Fortran: .o, .mod, and executables
- Python: .pyc (in __pycache__ since Python 3)
- LaTeX: .aux, .log, .nav, .out, .snm, .toc

What Not to Version Control

Most large binary files:

- Run results
- Animation movies
- Maybe images, PDFs, etc.

Binary Files and Version Control

Issues:

- They require lots of memory to manage; on the order of 2x their size
- A full copy is stored on every commit, rather than a diff. So frequently changed binary files can make a repository grow rapidly. That affects performance, and hosting services have size limits.

Binary Files and Version Control

Less of an issue for "hard won" binary files that aren't changed often:

- bathymetry, coordinates, mesh masks, initialization fields, climatology
  - Consider using a separate repository as you are developing these, and then commit the "final" versions in your working model configuration repository
- Finished products (like the .odp and PDF of this slide deck) in a single purpose repository

Binary Files and Version Control

If they aren’t under version control, they need to be backed up, redundantly

Backups

- Hard drives and SSD drives fail
- Computers get lost and stolen
- Bad things happen
- Your files and data needs to be stored in multiple, geographically separate places
- If you don't feel that you can recover in less than a day to the point where you can do research again, you should be worried, and you should take action to get less worried

Backups

- Every clone of a Git or Mercurial repository is a full backup of the repository and its history
- Repositories that have been pushed to hosting services like Gitlab, Bitbucket, Github automatically have at least one backup

Backups For Your Laptop

- Get at least one portable USB drive
- Use a tool to back up to it regularly
- Keep the drive separate from your laptop as much as possible
- Even better to have 2 portable drives stored in different locations
- Better still, use a cloud backup service in addition to at least one portable drive

Backup Tools

Mac:

- Time Machine: www.support.apple.com/kb/ht1427
- SuperDuper: www.shirt-pocket.com/SuperDuper
- Silverkeeper: www.lacie.com/silverkeeper
- ArRsync: www.arrsync.sourceforge.net

Windows:

- Syncback: www.2brightsparks.com/syncback/syncback-hub.html
- Fbackup: www.fbackup.com
- AOMEI Backupper: www.backup-utility.com
- EaseUS Todo: www.todo-backup.com

Linux:

- Déjà-dup: https://launchpad.net/deja-dup

This is a list, not recommendations. Read the docs and choose a tool you like and understand.

Model Runs Management

- Model configurations
- Files for:
  - Initialization
  - Boundary conditions
  - Forcing
- Executing model runs
- Run results files
- Analysis and visualization

Custom Model Code

- Most models have a mechanism to let you provide modified or extra code modules without changing the shipped model code:
  - MY_SRC directory in NEMO
  - ln3 tool in WaveWatch
- You should be able to keep your custom code in a version control repository that you control
  - Create a repository inside MY_SRC/
  - Symlink files into place from an external repository

Executing Model Runs

- Try to use tools that other people create and *maintain*
  - NEMO-Cmd
  - WaveWatch run_tests ??
  - Avoiding "not invented here" is hard
- If the tools are open-source on GitLab, Github, Bitbucket, etc. learn how to look at their repositories to judge their maturity, quality, and how well maintained they are

Model Runs Management Tools

- Sooner or later, though, you will have to write your own tools
- If you don't or can't script it, write notes/documentation about the exact steps
- Put those notes/docs under version control, of course!

Results Visualization

- Try to use tools that let you write code to create and adjust figures and animations rather than using a GUI tool to make adjustments:
  - Python and matplotlib, panda, xarray
  - Matlab scripts
  - R and ggplot2
- If you don't or can't script it, write notes/documentation about the exact steps
- Put those notes/docs under version control, of course!

Session Plan

- Definition
- Software Tools
- Organizing Your Modeling Research Life
- Exercise #1
- Writing Automation Tools
- Exercise #2
- NEMO command processor

Organizing Your Modeling Research Life

- Why?
- What?
- How?

Why Does Directory and File Organization Matter?

A well structured directory tree with meaningful, consistent directory and file names is:

- Easier for humans (future you!, your supervisor) to understand
- Easier to write automation scripts against
- A step towards reproducibility

What Needs to be Organized?

- Model source code and executables
- Model configurations
- Files for:
  - Initialization
  - Boundary conditions
  - Forcing
- Directories for executing model runs
- Run results directories and files
- Analysis and visualization code and products

Exercise #1a

- Create a list of the things that you need to organize for your model research

Idalia Machuca – Mackenzie Canyon Circulation & Upwelling

“My research is, in short, looking at the circulation around Mackenzie Canyon, in particular the upwelling dynamics. To that goal, I'm using the NEMO model to simulate the circulation and comparing the results between idealized and realistic cases. The idealized bathymetry is constantly being modified to represent the realistic bathymetry more accurately. What's great about keeping my files organized is that it helps me keep track of the constantly changing bathymetry files - which in turn sometimes require new coordinates and initial conditions files.”

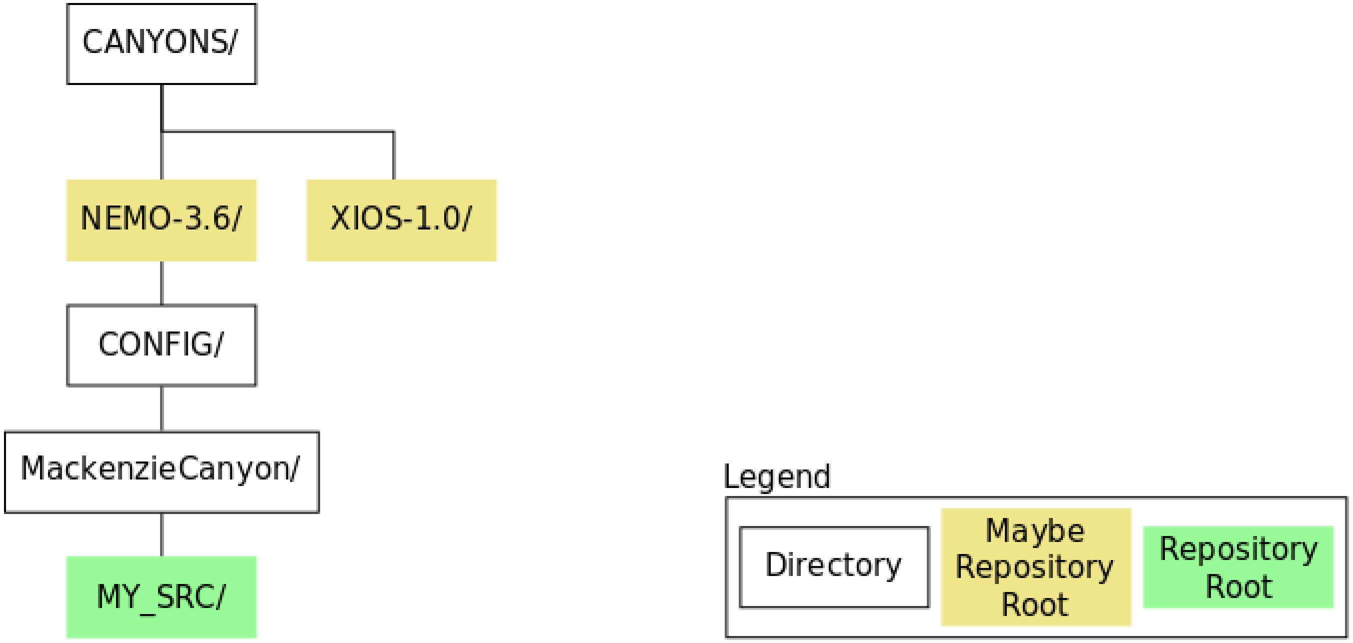
Link to figures showing Idalia’s idealized and realistic bathymetries

Directories and Repositories

- Choose a top level project name; e.g. CANYONS
- Be consistent across platforms when you set up your file space:
  - /home/doug/Documents/CANYONS/ on my laptop
  - /ocean/dlatorne/CANYONS/ on our dev compute server
  - /home/dlatorne/CANYONS/ on HPC

Model Code and Executables

- One or more directories:
- Cloned or checked out from upstream repositories if possible
  - NEMO-3.6/, XIOS-2.0/ by svn checkout
  - wwatch3-5.16/ by unpacking downloaded tarball


CANYONS/
NEMO-3.6/
XIOS-1.0/
CONFIG/
MackenzieCanyon/
MY_SRC/
Legend
Directory
Maybe Repository Root
Repository Root

Naming Things

Don't use spaces, parentheses, or other special characters in file or directory names:

- They won’t tab-complete well
- They can be hard to handle in automation scripts

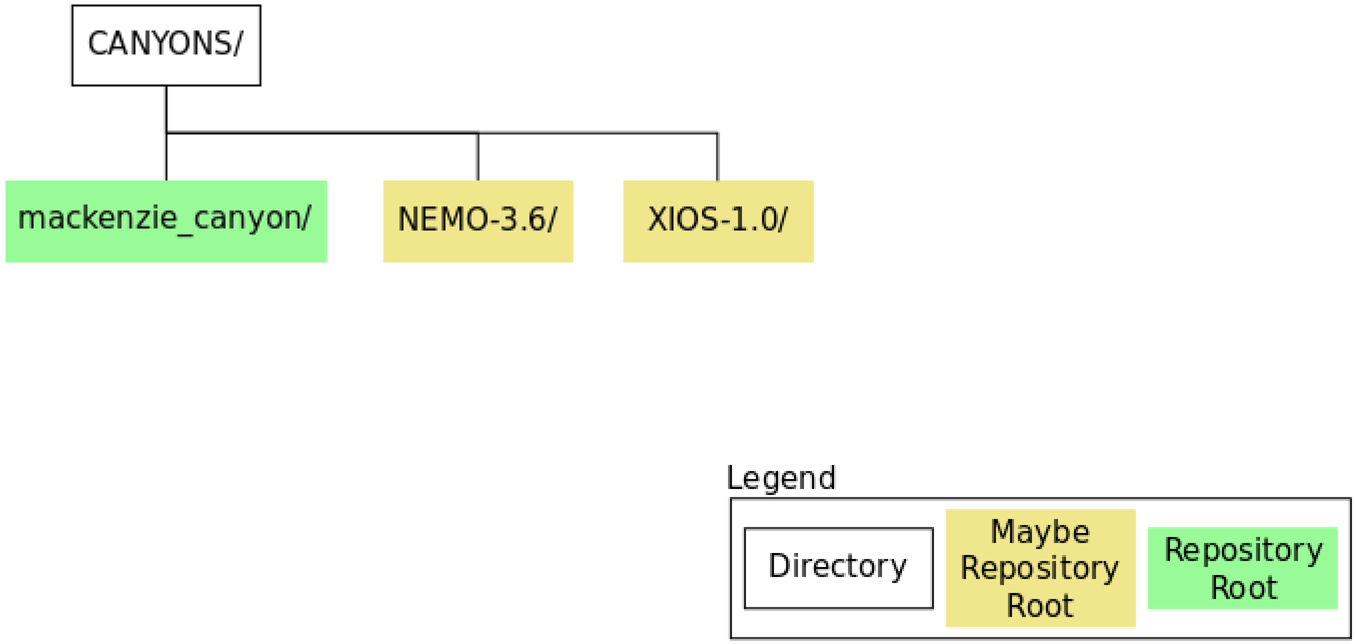
Instead use:

- CamelCase
- snake_case
- separate-words-with-hyphens

Directories and Repositories

Model configuration repository:

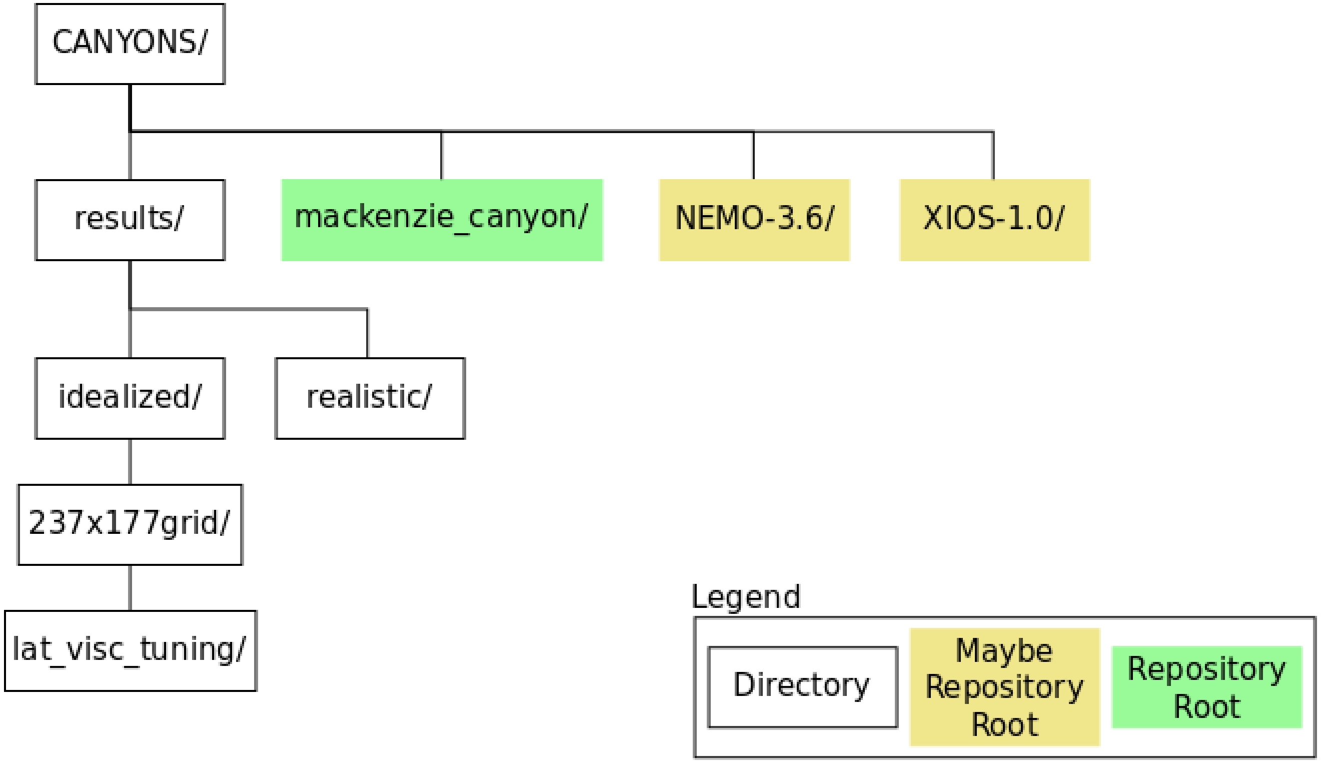
- Coordinates, bathymetry, namelists, output file definitions, etc.
- Automation scripts might live here, or in a separate repository
- Absolutely a version control repository
- Example path:
  - mackenzie_canyon/


CANYONS/
mackenzie_canyon/
NEMO-3.6/
XIOS-1.0/
Legend
Directory
Maybe Repository Root
Repository Root

Directories and Repositories

results/ directory:

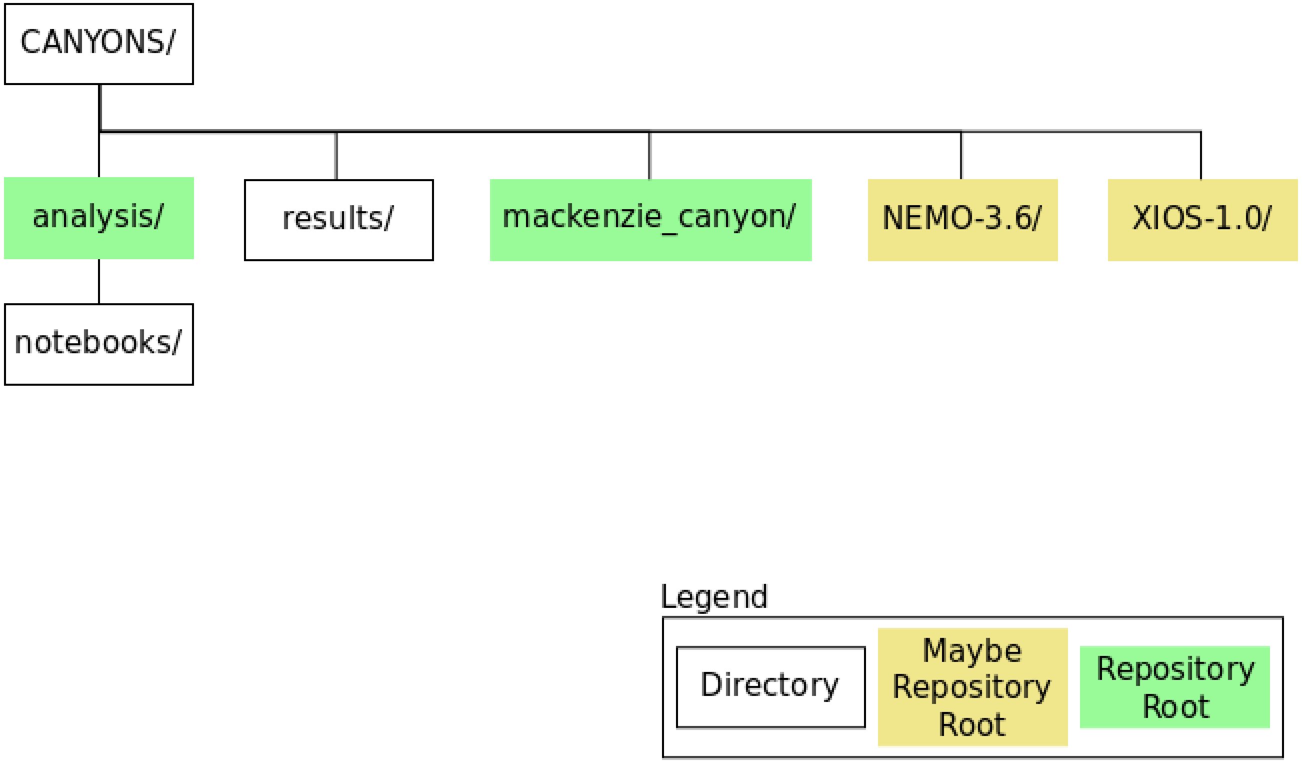
- Tree of systematically named directories that hold run results
- *Not* a version control repository
- Example path:
  - results/idealized/237x177grid/lat_visc_sensitivity/10m2ps/


CANYONS/
results/
mackenzie_canyon/
NEMO-3.6/
XIOS-1.0/
idealized/
realistic/
237x177grid/
lat_visc_tuning/
Legend
Directory
Maybe Repository Root
Repository Root

Directories and Repositories

analysis/ directory:

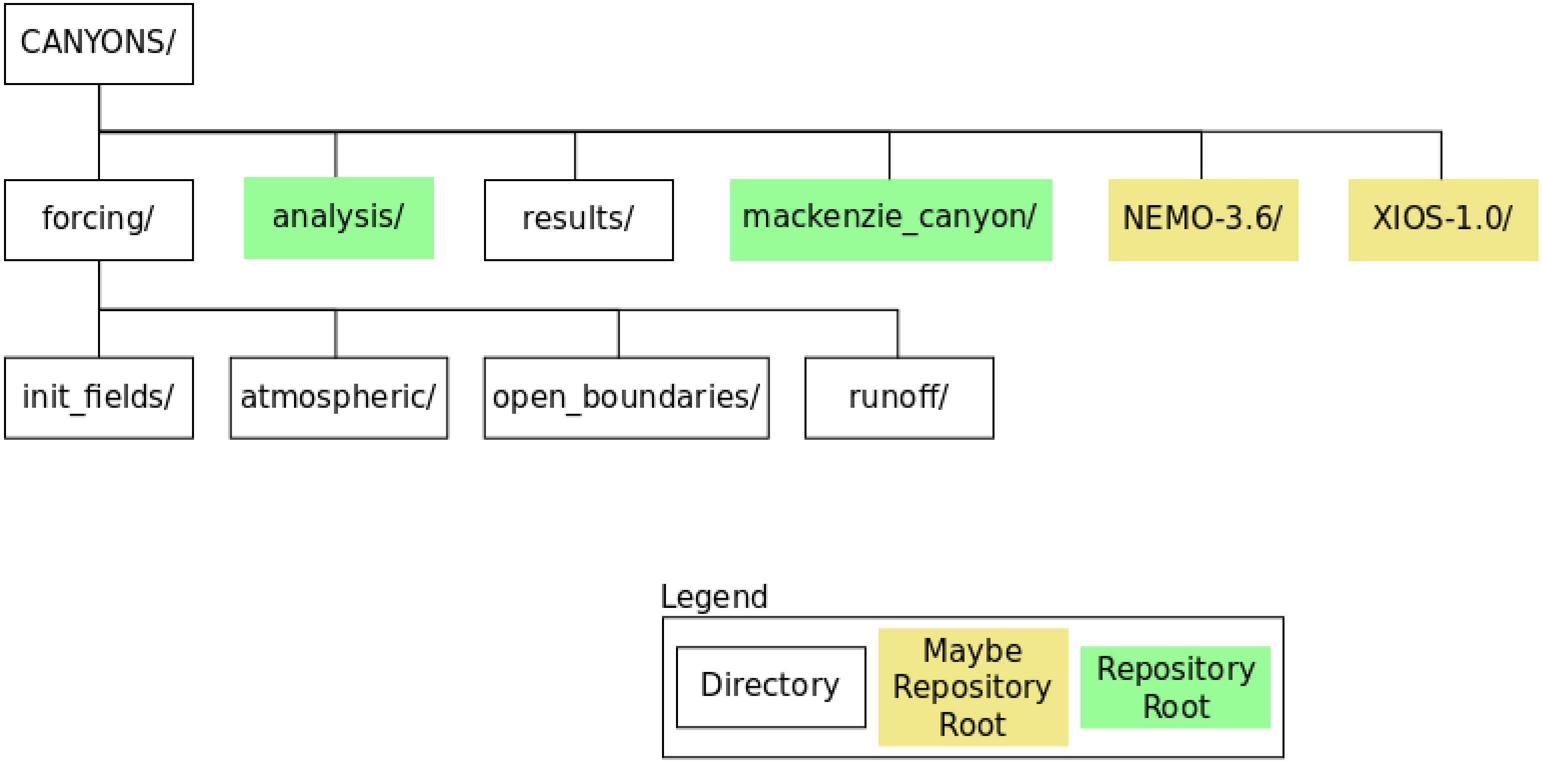
- Code and Jupyter Notebooks for analysis and visualization of results
- Absolutely a version control repository


CANYONS/
analysis/
results/
mackenzie_canyon/
NEMO-3.6/
XIOS-1.0/
notebooks/
Legend
Directory
Maybe Repository Root
Repository Root

Directories and Repositories

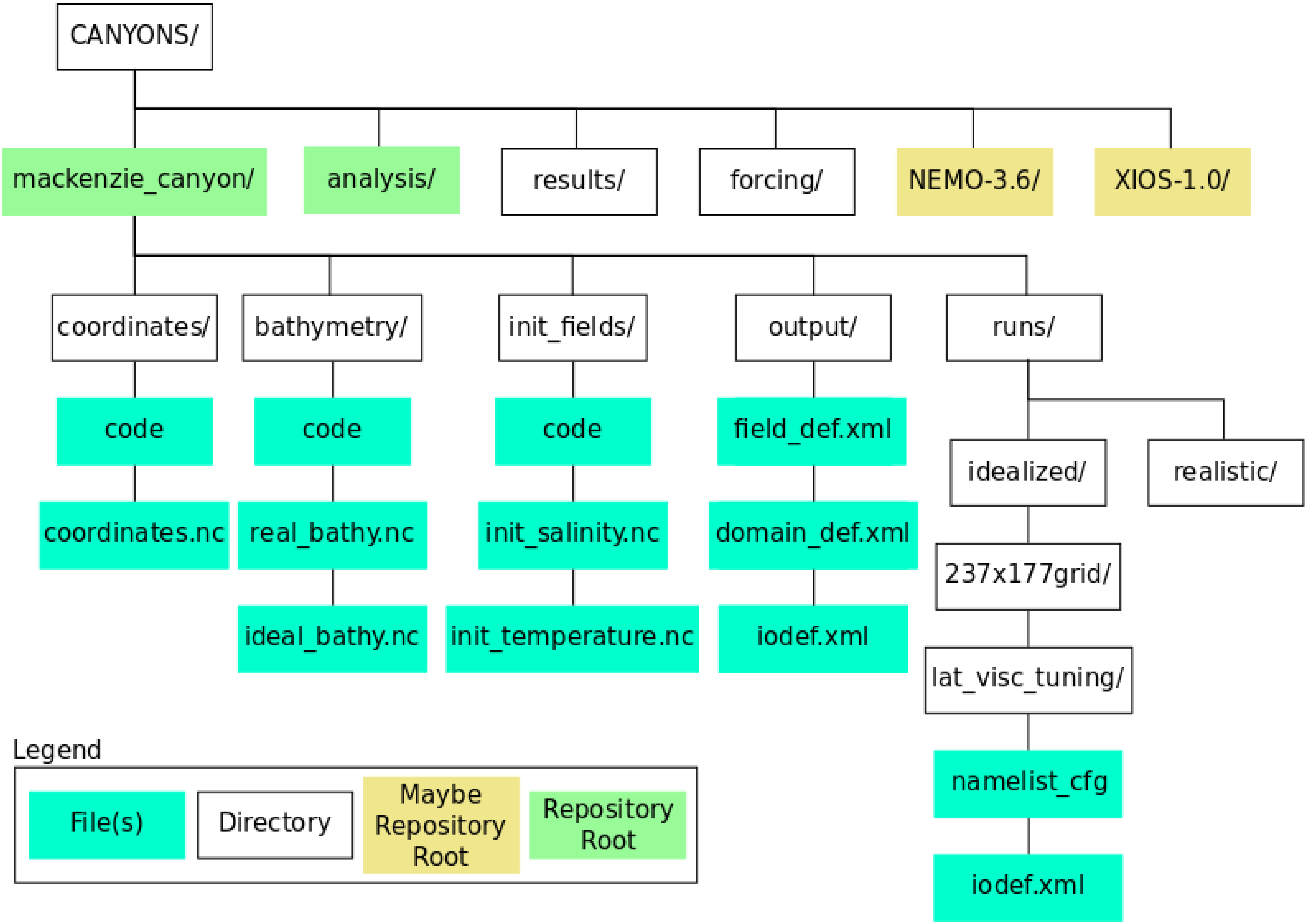
forcing/ directory:

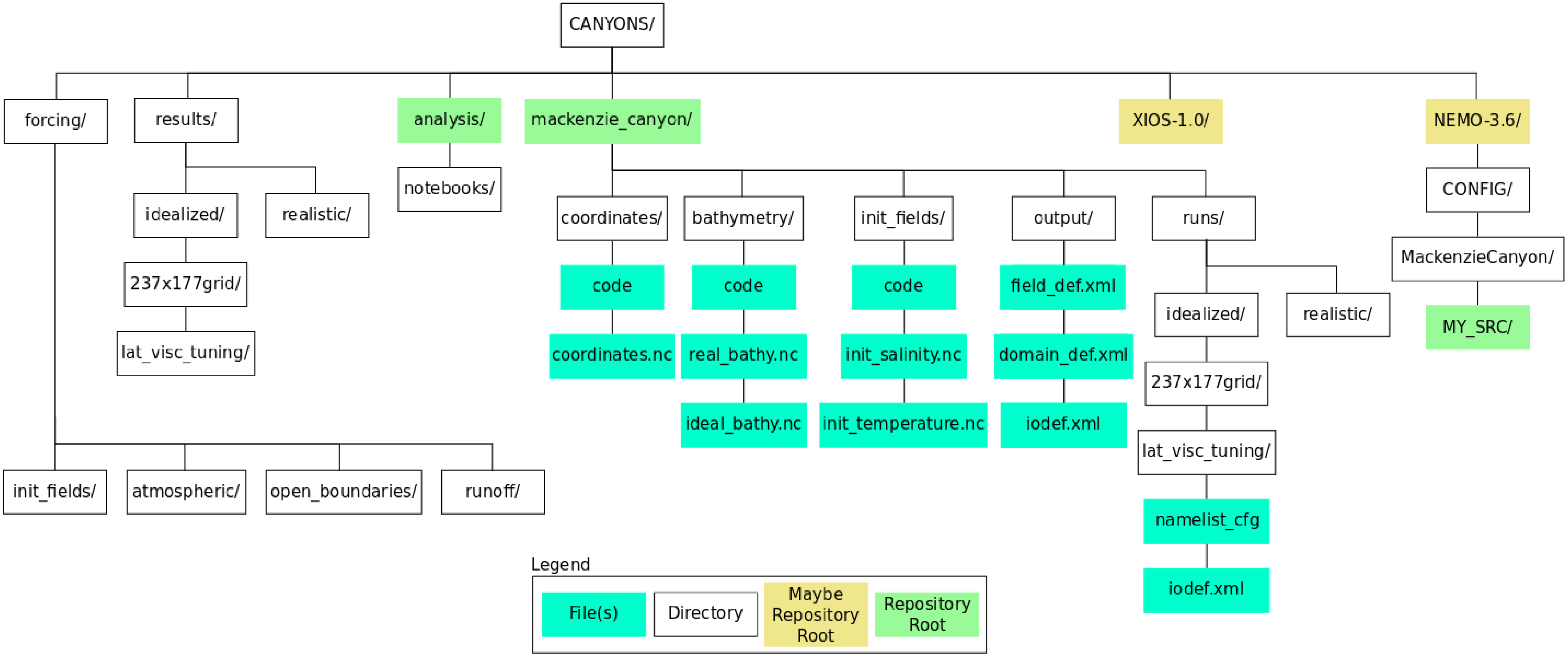
- A tree of systematically named directories that hold initialization and/or forcing files; e.g.
  - atmospheric/
  - init_fields/
  - open_boundaries/
  - runoff/
- You might not need all of these, any of them, or you might need different forcing files
- *Not* a version control repository, but the code that produces these files should be under version control in the model configuration repository, or a separate tools repository


CANYONS/
forcing/
analysis/
results/
mackenzie_canyon/
NEMO-3.6/
XIOS-1.0/
init_fields/
atmospheric/
open_boundaries/
runoff/
Legend
Directory
Maybe Repository Root
Repository Root

Model Configuration Repository

- Coordinates, bathymetry, namelists, output file definitions, etc.
- Automation scripts might live here, or in a separate repository
- Absolutely a version control repository
- Example:
  - Idalia's mackenzie_canyon/ repository for NEMO-3.6


CANYONS/
mackenzie_canyon/
analysis/
results/
forcing/
NEMO-3.6/
XIOS-1.0/
coordinates/
bathymetry/
init_fields/
output/
runs/
code
code
code
field_def.xml
idealized/
realistic/
coordinates.nc
real_bathy.nc
init_salinity.nc
domain_def.xml
237x177grid/
ideal_bathy.nc
init_temperature.nc
iodef.xml
lat_visc_tuning/
namelist_cfg
iodef.xml
Legend
File(s)
Directory
Maybe Repository Root
Repository Root


CANYONS/
forcing/
results/
analysis/
mackenzie_canyon/
XIOS-1.0/
NEMO-3.6/
notebooks/
idealized/
realistic/
237x177grid/
lat_visc_tuning/
coordinates/
bathymetry/
init_fields/
output/
runs/
code
code
code
field_def.xml
coordinates.nc
real_bathy.nc
init_salinity.nc
domain_def.xml
ideal_bathy.nc
init_temperature.nc
iodef.xml
idealized/
realistic/
237x177grid/
lat_visc_tuning/
namelist_cfg
iodef.xml
CONFIG/
MackenzieCanyon/
MY_SRC/
init_fields/
atmospheric/
open_boundaries/
runoff/
Legend
File(s)
Directory
Maybe Repository Root
Repository Root

Exercise #1b

Create a diagram of repositories and directories for your model research

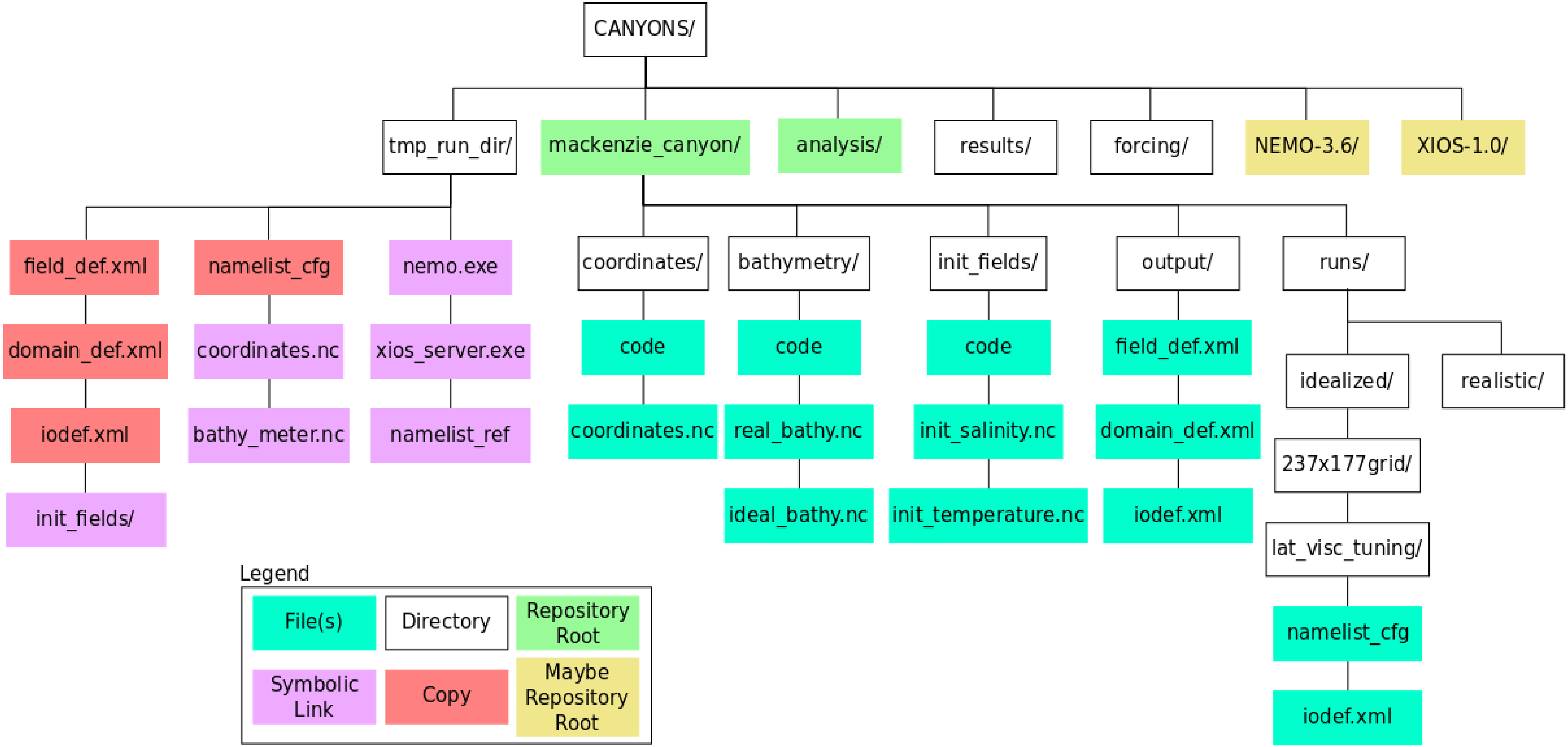
Temporary Run Directories

Execute each model run in a directory that you set up for that run only. Move the run results *and* the run configuration files to results storage.

- Provides checks on run intent and ready-to-run status
- Helps ensure reproducibility
- Well-suited to automation

Temporary Run Directories

- Create a specific, temporary directory for each model run
- Assemble the files needed for the run by copying or symbolic linking
- Configure the run so that its output goes into the directory
- Execute the run
- Do routine post-processing of model output
- Move the run results to a directory in the results/ tree
- Delete the symbolic links and the temporary directory


CANYONS/
tmp_run_dir/
mackenzie_canyon/
analysis/
results/
forcing/
NEMO-3.6/
XIOS-1.0/
field_def.xml
domain_def.xml
iodef.xml
init_fields/
namelist_cfg
coordinates.nc
bathy_meter.nc
nemo.exe
xios_server.exe
namelist_ref
coordinates/
code
coordinates.nc
bathymetry/
code
real_bathy.nc
ideal_bathy.nc
init_fields/
code
init_salinity.nc
init_temperature.nc
output/
field_def.xml
domain_def.xml
iodef.xml
runs/
idealized/
237x177grid/
lat_visc_tuning/
namelist_cfg
iodef.xml
realistic/
Legend
File(s)
Directory
Repository Root
Symbolic Link
Copy
Maybe Repository Root

Directories and Repositories

- It will probably take you a few iterations to get a structure that works well for you, your research, and your automation
- Don't be afraid to change the structure if you need to, but think about your changes to make sure that they are an improvement, then commit to them

Session Plan

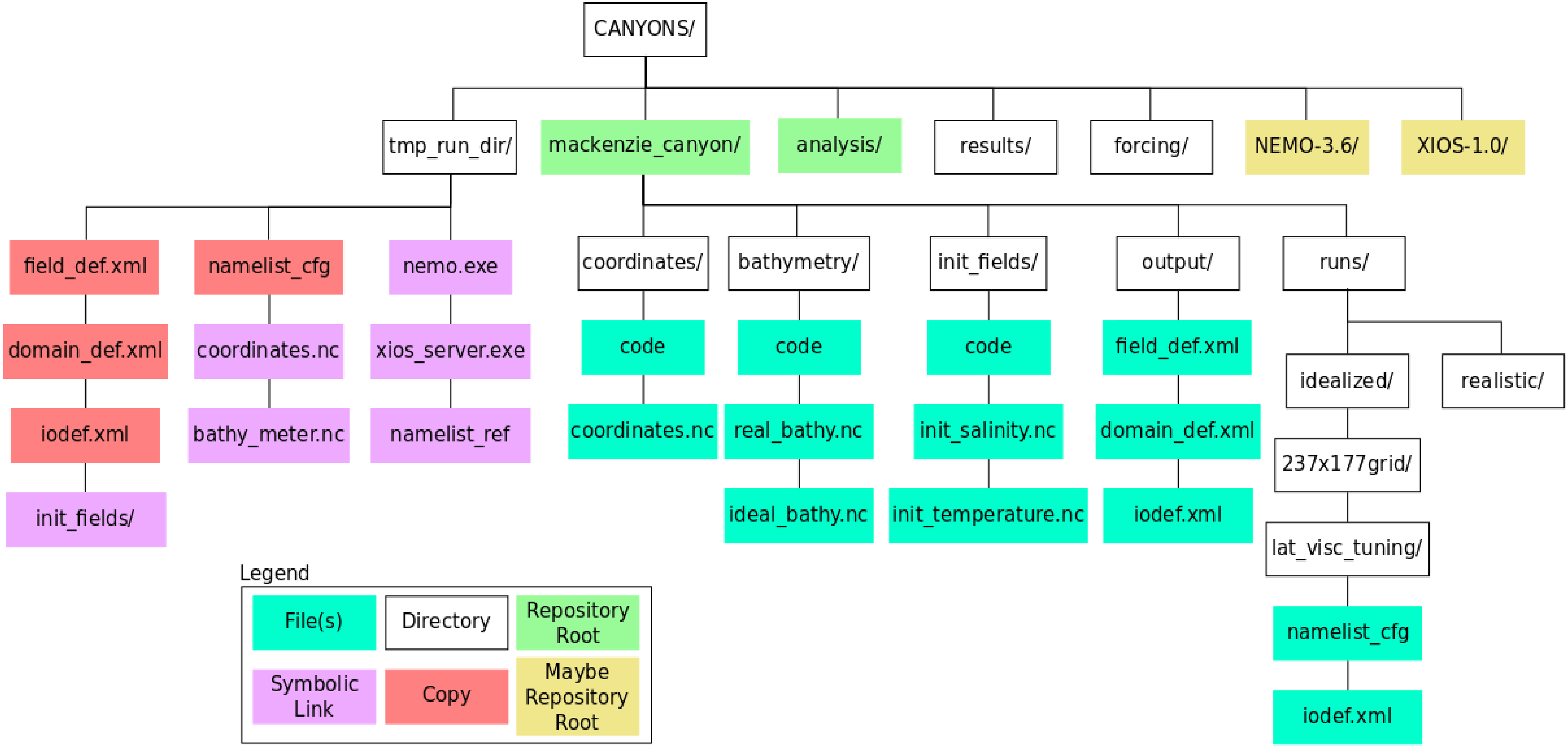
- Definition
- Software Tools
- Organizing Your Modeling Research Life
- Exercise #1
- Writing Automation Tools
- Exercise #2
- NEMO command processor

Writing Automation Tools

- Shell scripts
- Python

Shell Scripts

- bash on Linux, cmd.exe on Windows
- Good starting point for repeated commands


CANYONS/
tmp_run_dir/
mackenzie_canyon/
analysis/
results/
forcing/
NEMO-3.6/
XIOS-1.0/
field_def.xml
domain_def.xml
iodef.xml
init_fields/
namelist_cfg
coordinates.nc
bathy_meter.nc
nemo.exe
xios_server.exe
namelist_ref
coordinates/
code
coordinates.nc
bathymetry/
code
real_bathy.nc
ideal_bathy.nc
init_fields/
code
init_salinity.nc
init_temperature.nc
output/
field_def.xml
domain_def.xml
iodef.xml
runs/
idealized/
237x177grid/
lat_visc_tuning/
namelist_cfg
iodef.xml
realistic/
Legend
File(s)
Directory
Repository Root
Symbolic Link
Copy
Maybe Repository Root

Shell Scripts

```
NEMO=${HOME}/CANYONS/NEMO-3.6/CONFIG/MackenzieCanyon
CONFIG=${HOME}/CANYONS/mackenzie_canyon
RUN=${CONFIG}/runs/idealized/237x177grid/lat_visc_tuning
TMP_RUN=${HOME}/CANYONS/tmp_run_dir
cd ${TMP_RUN}
ln -s ${NEMO}/EXP00/opa nemo.exe
ln -s ${NEMO}/EXP00/xios_server.exe
ln -s ${CONFIG}/bathymetry/ideal_bathy.nc bathy_meter.nc
...
cp ${CONFIG}/output/field_def.xml ./
cp ${RUN}/iodef.xml ./
cp ${RUN}/namelist_cfg ./
```

```
$ mkdir -p ${HOME}/CANYONS/tmp_run_dir
$ bash prep_run_dir.sh
```

Shell Scripts

Fairly easy to accept arguments from command-line:

```
NEMO=${HOME}/CANYONS/NEMO-3.6/CONFIG/MackenzieCanyon
CONFIG=${HOME}/CANYONS/mackenzie_canyon
RUN=${CONFIG}/runs/${1}
TMP_RUN=${2}
cd ${TMP_RUN}
ln -s ${NEMO}/EXP00/opa nemo.exe
ln -s ${NEMO}/EXP00/xios_server.exe
ln -s ${CONFIG}/bathymetry/ideal_bathy.nc bathy_meter.nc
...
cp ${CONFIG}/output/field_def.xml ./
cp ${RUN}/iodef.xml ./
cp ${RUN}/namelist_cfg ./
```

```
$ TMP_RUN=${HOME}/CANYONS/tmp_run_dir
$ mkdir -p ${TMP_RUN}
$ bash prep_run_dir.sh idealized/237x177grid/lat_visc_tuning ${TMP_RUN}
```

Shell Scripts

Easy to loop over collections of files, sequences of numbers:

```
for d in {15..21}
do
    python -m nowcast.workers.download_PSY4 \
      ${NOWCAST_YAML} --run-date 2017-02-${d}
done
```

```
$ for d in {15..21}; do python -m nowcast.workers.download_PSY4 $NOWCAST_YAML
--run-date 2017-02-$d; done
```

Shell Scripts

- Syntax is somewhat clunky and dated
- Processing options from command-line is verbose and painful

Python

- General purpose programming language
- Choose Python 3
- “Code is more often read than written”
- Core language: https://docs.python.org/3/
- Standard library: https://docs.python.org/3/library/
- 3rd party packages: https://pypi.python.org/pypi
- Anaconda distribution: https://www.continuum.io/downloads

Exercise #2

Managing Ariane output

Create a Python tool to run in an Ariane run directory that takes 2 arguments:

- a model run date
- a results directory parent

The tool will:

- Create a new results directory under the results directory parent
- The name of the directory will be derived from the model run date argument
- Move the input and output files from Ariane into the new results directory
- Rename the files to include the model run date; e.g. traj_20160417.txt

Intro to Python for Code Automation

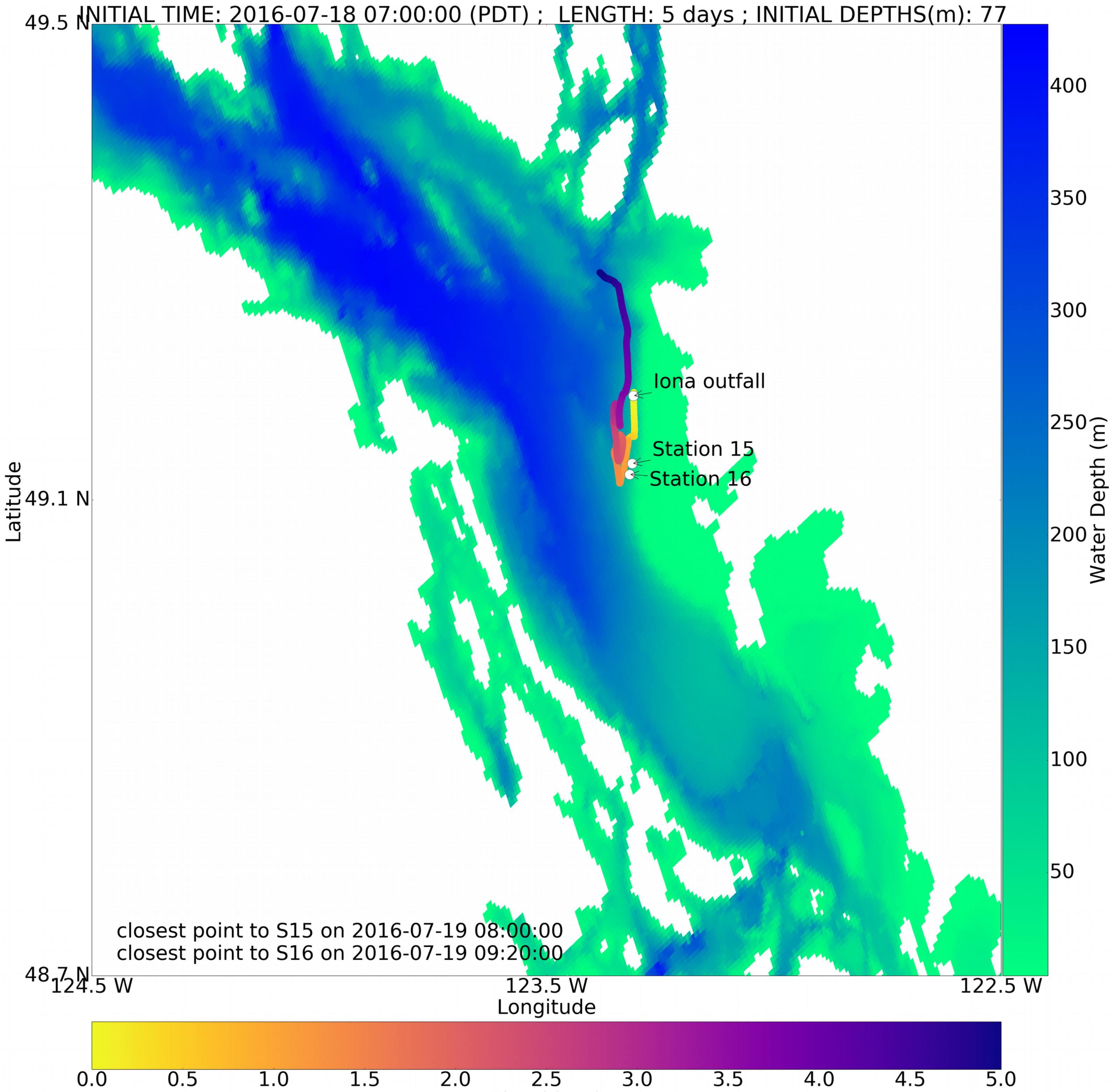
Jupyter Notebook link on nbviewer

- Boilerplate for a Python module to run from the command-line
- Handling command-line arguments
- Working with dates and times
- Working with files and directories

Ariane Particle Tracking Tool

http://stockage.univ-brest.fr/~grima/Ariane/

- Offline calculation of 3D streamlines in the output velocity field of ocean models
- NEMO, ROMS, Symphonie, MIT-GCM



INITIAL TIME: 2016-07-18 07:00:00 (PDT) ; LENGTH: 5 days ; INITIAL DEPTHS(m): 77

closest point to S15 on 2016-07-19 08:00:00
closest point to S16 on 2016-07-19 09:20:00

Ariane Runs

Input files:

- namlist, initial_positions.txt

Output files:

- traj.txt, log.txt

Exercise #2

Managing Ariane output

Create a Python tool to run in an Ariane run directory that takes 2 arguments:

- a model run date
- a results directory parent

The tool will:

- Create a new results directory under the results directory parent
- The name of the directory will be derived from the model run date argument
- Move the input and output files from Ariane into the new results directory
- Rename the files to include the model run date; e.g. traj_20160417.txt

Exercise #2

```
ssh -XY djl@mingan.uqar.ca
```

```
$ module load python/3.6
```

```
$ wget https://bitbucket.org/douglatornell/uqar-winter-school/downloads/particle_tracking.tar.gz
$ tar -xvzf particle_tracking.tar.gz
```

```
particle_tracking/
├── ariane_run/
│   ├── initial_positions.txt
│   ├── log.txt
│   ├── namelist
│   └── traj.txt
└── move_ariane_results.py
```

If your favourite editor isn’t available, try gedit or nano. (Minimal, but almost always available on Linux.)

```
$ cd ariane_run
$ python3 ../move_ariane_results.py 2017-03-06 ../results
```

```
$  rsync -a particle_tracking/ particle_tracking_test
```

Other Packages

Standard llibrary:

- glob
- pathlib
- subprocess

3rd party:

- arrow
- pytz
- python-dateutil

3^{rd} Party Python Packages

Package managers:

- conda
- pip

- If you’re using Anaconda, try conda first; pip may trigger compilation of C extensions
- Both have search, install, list, uninstall, etc. sub-commands

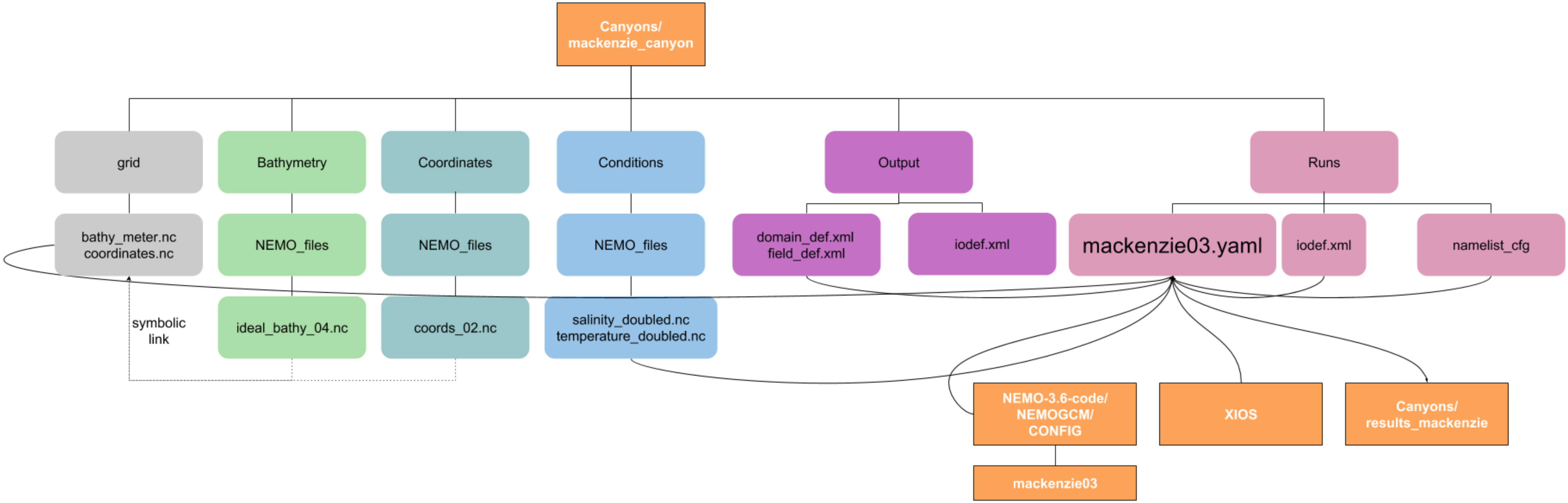
Session Plan

- Definition
- Software Tools
- Organizing Your Modeling Research Life
- Exercise #1
- Writing Automation Tools
- Exercise #2
- NEMO command processor

NEMO Command Processor

Example of building out automation tools over time to the point of executing a complete NEMO run from one file with one command

- Documentation: https://nemo-cmd.readthedocs.io/en/latest/
- Code Repository: https://bitbucket.org/salishsea/nemo-cmd


Canyons/
mackenzie_canyon
grid
Bathymetry
Coordinates
Conditions
Output
Runs
bathy_meter.nc
coordinates.nc
NEMO_files
NEMO_files
NEMO_files
domain_def.xml
field_def.xml
iodef.xml
mackenzie03.yaml
iodef.xml
namelist_cfg
symbolic
link
ideal_bathy_04.nc
coords_02.nc
salinity_doubled.nc
temperature_doubled.nc
NEMO-3.6-code/
NEMOGCM/
CONFIG
XIOS
Canyons/
results_mackenzie
mackenzie03

nemo run Command

```
$ nemo run vert_eddy_diff.yaml
/ocean/dlatorne/CANYONS/results/realistic/237x177grid/vert_eddy_diff_1e-5/

nemo_cmd.prepare WARNING: There are uncommitted changes in /results/nowcast-
sys/NEMO-3.6-code/

nemo_cmd.run INFO: Created run directory
/ocean/dlatorne/CANYONS/tmp_run_dirs/a8b899de-01be-11e7-ba4b-0025909a8460
```

Temporary Run Directory

- Copies of run configuration files
- Symbolic links to executables, bathymetry, etc.
- Version control system revision records
- Shell script to execute run

Run Description YAML File

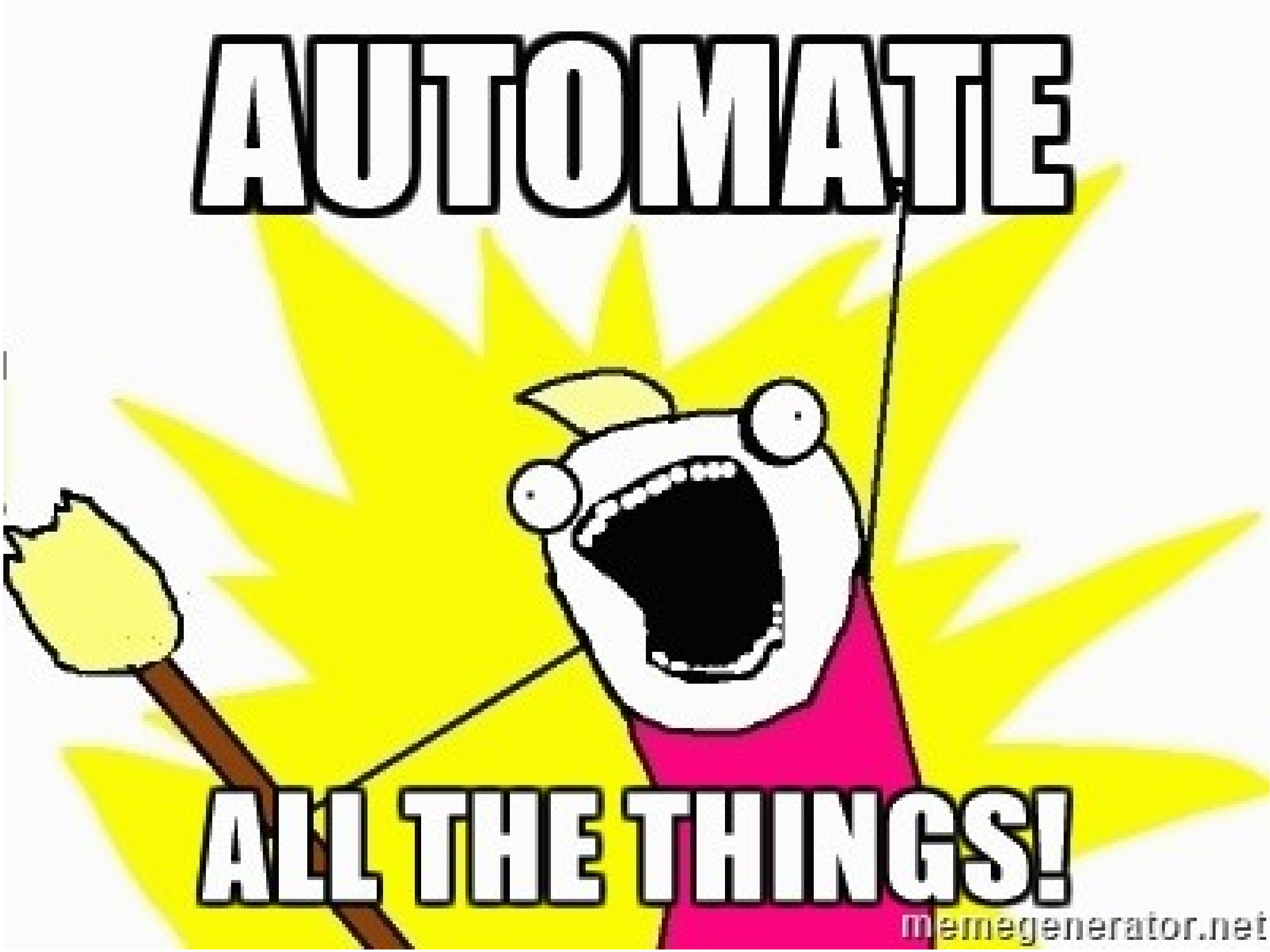
- YAML is easily readable by humans and computers
- Fully described the run
- Used by NEMO command processor to prepare the temporary run directory and the run shell script

Run Shell Script

- PBS directives
- Set up variables
- Create results directory
- Launch run
- Do routine results post-processing
- Move run and results files to results directory
- Delete temporary run directory

Wrap-Up

- Software Tools
  - Version Control and Backups
  - Model Run Automation and Visualization Scripting
- Organizing Your Modeling Research Life
  - Reproducibility and Automatability
- Writing Automation Tools
  - Shell and Python
  - Shift focus from details of running model to gaining research insights into model run results
- NEMO command processor


AUTOMATE
ALL THE THINGS!
memegenerator.net