Software Version Control and Automation

TU Delft CITG Lunch Seminar
31-Mar-2014

Doug Latornell
Earth, Ocean & Atmospheric Sciences
University of British Columbia, Vancouver, Canada
dlatornell@eos.ubc.ca
http://douglatornell.ca
@dlatornell

Part 1 - Version Control

- What is it?
- Why use it?
- What for?
- Key Concept
- History, Tools, Pros and Cons
- Mercurial and Git
- Key Disciplines
- Tagging, Reverting, and Updating Backward
- GUIs
- Collaboration
- Bitbucket and GitHub

What Is Version Control (VC)?

Use software tools to keep a running record of 1 or more files.

What Is VC?

Use software tools to keep a running record of 1 or more files.

Why You Should Use VC?

- Lets you revert to earlier versions of your work
- Provides a record of what changed when
- Lets you mark significant points in time
- Allows you to play "what-if?"
- Facilitates organized collaboration (with your future self, as well as with other people)

What You Should Use VC For

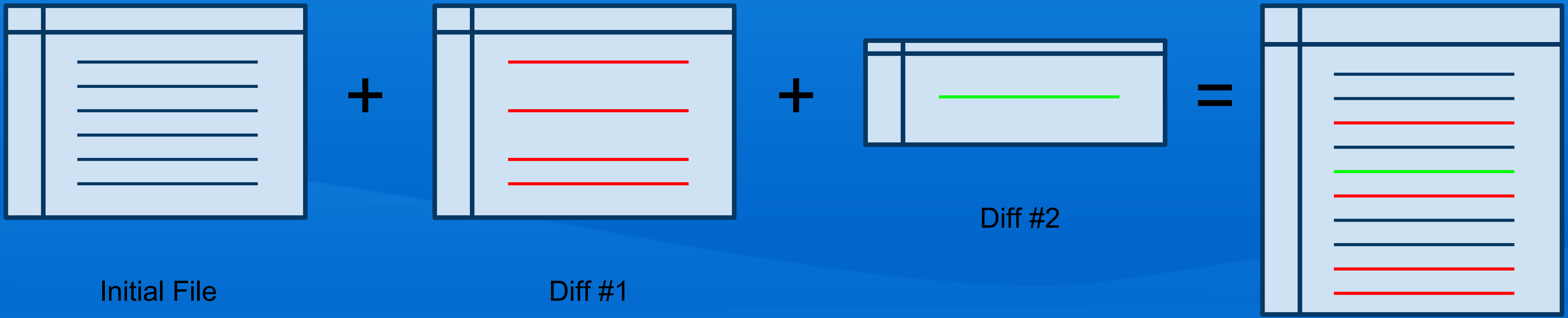
- Model Code
- Matlab Scripts
- Plotting Scripts
- Processed Data Files & Scripts That Made Them
- Complicated Marking Spreadsheets (especially if shared)
- Thesis
- Papers
- ToDo List

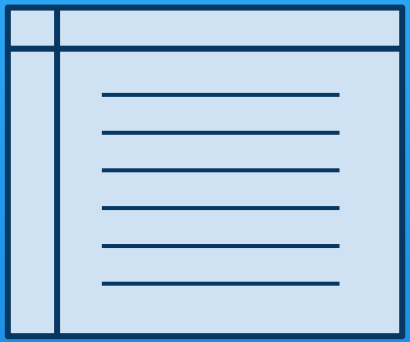
What You Should Use VC For

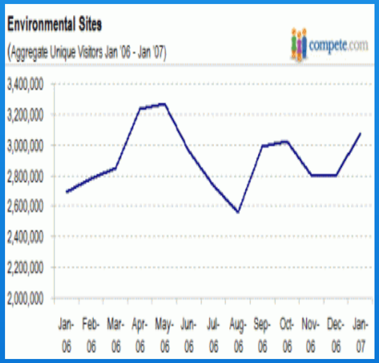
- Model Code
- Matlab Scripts
- Plotting Scripts
- Processed Data Files & Scripts That Made Them
- Complicated Marking Spreadsheets (especially if shared)
- Thesis
- Papers
- ToDo List

Key Concept

- Data differencing
- Unix utilities diff and patch
- Given a file, and a complete set of diffs between 1 state and another, any intermediate state for which there is a diff can be reconstructed.



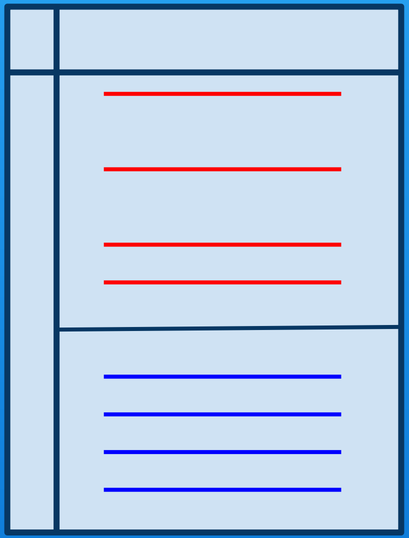




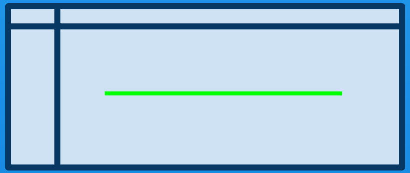
Environmental Sites

(Aggregate Unique Visitors Jan '06 - Jan '07)

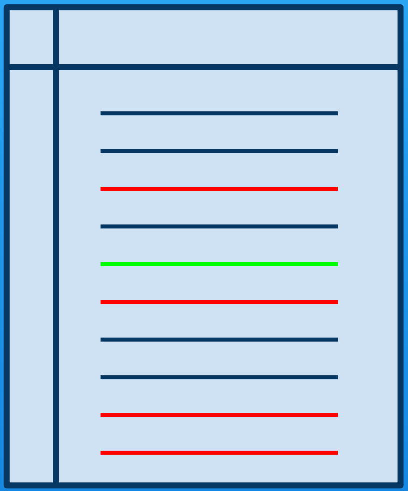
+

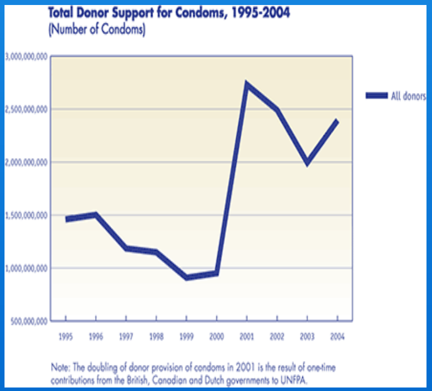


+



=





Total Donor Support for Condoms, 1995-2004

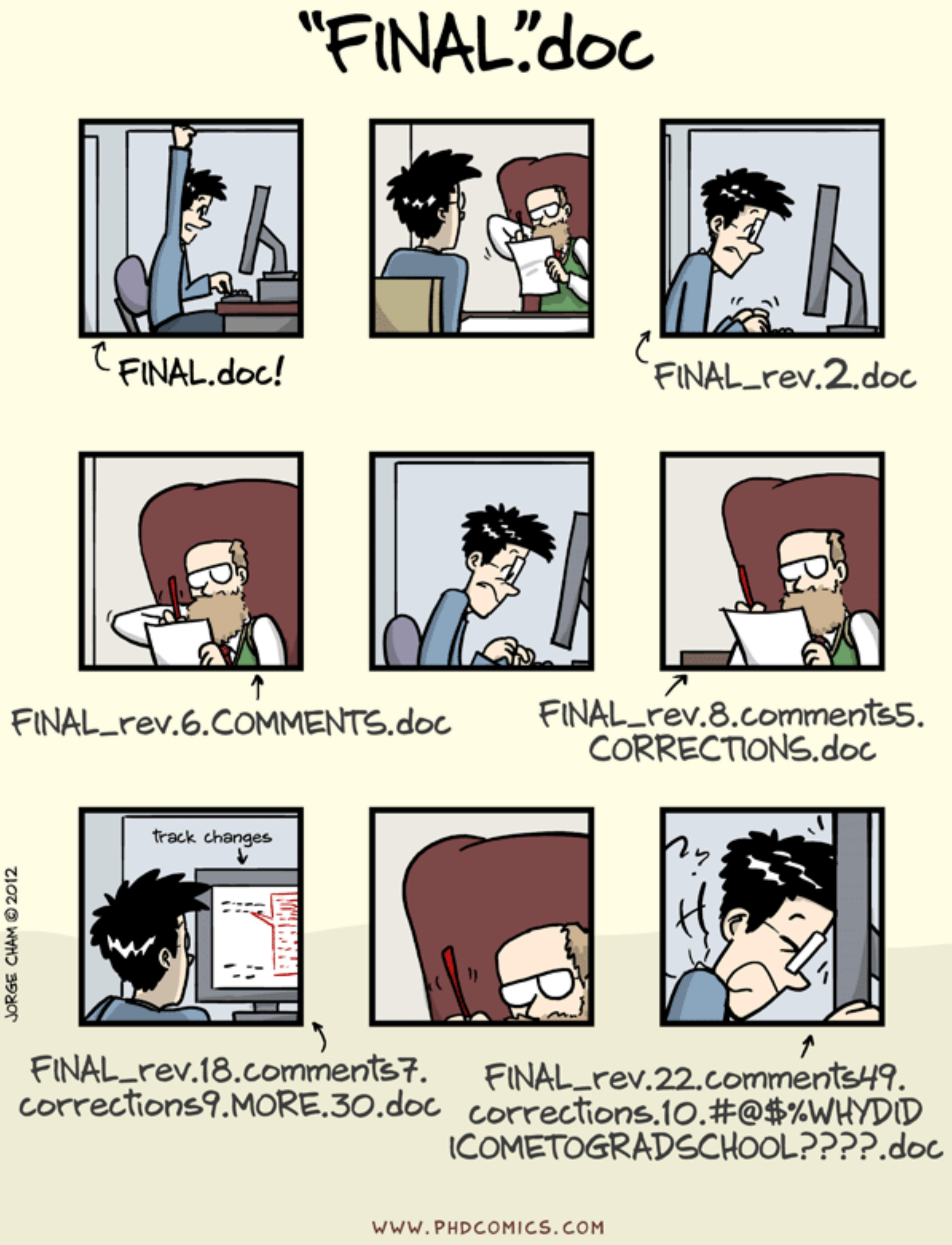
(Number of Condoms)

Note: The doubling of donor provision of condoms in 2001 is the result of one-time contributions from the British, Canadian and Dutch governments to UNFPA.

Version Control Tools

http://en.wikipedia.org/wiki/Revision_control

Ad hoc:


"FINAL".doc
FINAL.doc!
FINAL_rev.2.doc
FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.
CORRECTIONS.doc
track changes
FINAL_rev.18.comments7.
corrections9.MORE.30.doc
FINAL_rev.22.comments49.
corrections.10.#@$%WHYDID
ICOMETOGRADSCHOOL????.doc
JORGE CHAM © 2012
WWW.PHDCOMICS.COM

Version Control Tools

http://en.wikipedia.org/wiki/Revision_control

Ad hoc

thesis2.tex, JFM-21mar.doc
pooh.txt, ...

Mists of time...

SCCS
RCS

Version Control Tools

http://en.wikipedia.org/wiki/Revision_control

Ad hoc

thesis2.tex, JFM-21mar.doc
pooh.txt, ...

Mists of time...

SCCS
RCS

Proprietary:

Visual SourceSafe
Perforce
BitKeeper

Version Control Tools

http://en.wikipedia.org/wiki/Revision_control

Ad hoc

thesis2.tex, JFM-21mar.doc
pooh.txt, ...

Mists of time...

SCCS
RCS

Proprietary:

Visual SourceSafe
Perforce
BitKeeper

Old School (Client/Server):

CVS (Concurrent Versions System)
SVN (Subversion)

Version Control Tools

http://en.wikipedia.org/wiki/Revision_control

Ad hoc

thesis2.tex, JFM-21mar.doc
pooh.txt, ...

Mists of time...

SCCS
RCS

Proprietary:

Visual SourceSafe
Perforce
BitKeeper

Old School (Client/Server):

CVS (Concurrent Versions System)
SVN (Subversion)

Distributed & Open Source:

GNU arch
Darcs
Monotone
Bazaar

Git
Mercurial

Pros and Cons

Ad Hoc

- Easy to do, if you think of it
- Works best if you have a system
  - `stuff1.txt, stuff2.f90, stuff4.m` probably isn't a good enough system
- Hard to provide yourself with enough metadata

Pros and Cons

Client/Server

- Good for centrally controlled project; e.g. ROMS
- Work required to set up and administer
- Committing feels like a big deal
- Requires network connection

Distributed

- Almost zero set up
- No network required
- Every copy of a repository is a full backup
- Scalable to big projects
- Usable for central control

Mercurial

http://mercurial.selenic.com/
http://mercurial.selenic.com/wiki/Tutorial
Mercurial: The Definitive Guide http://hgbook.red-bean.com/

```
$ hg help
```

Git

http://git-scm.com/
http://git-scm.com/documentation
Pro Git http://git-scm.com/book

```
$ git help
```

hg Commands to Start a Project

```
$ hg init myhgproject
$ cd myhgproject

    add/create some files

$ hg add
$ hg commit -m "Initial commit."
```

git Commands to Start a Project

```
$ git init mygitproject
$ cd mygitproject

    add/create some files

$ git add
$ git commit -m "Initial commit."
```

Key Disciplines

Commit Early, Commit Often

- Small incremental changes are easier to understand
- You can't revert to a diff that doesn't exist

Make Commit Messages Informative

- 1st line is a summary; sometimes that's all you need
- Add more details in subsequent paragraphs
- Use present tense; e.g. "Fix typos."
- See http://tbaggery.com/2008/04/19/a-note-about-git-commit-messages.html

hg Commands to See What's Going On

```
$ hg log
```

Print revision history of files or whole repository

```
$ hg diff
```

Show differences between revisions

```
$ hg status
```

Show status of files in repository (e.g. modified, added, removed, missing, not tracked)

N.B. There are lots of options for each command

See `hg help command`

Tags

Tags are symbolic names for specific revisions in the repository. Most often you assign a tag to the current revision (tip) to mark a significant event.

```
$ hg tag -m"1st submission to JGR." jgr_1
```

Tag the current revision as `jgr_1`

```
$ hg tags
```

Print a list of the tags in the repository

Reverting

```
$ hg revert -r67 paper.tex
```

Revert `paper.tex` to the contents it had at revision 67; `paper.tex` will be marked as modified

```
$ hg revert --all
```

Discard all changes since last commit

`hg revert` changes file contents, but not the working directory parents, so you have to commit the reverted file(s)

Use `revert` if you made a mistake and want to go back (but repository history is *always* preserved)

Updating Backward

```
$ hg update -d"<2010-10-01"
```

Update the repository to the last revision prior to 2010-10-01

```
$ hg update -r jgr_1
```

Update the repository to revision `jgr_1`

`hg update` changes file contents, *and* the working directory parents, so there are no changes to commit

Jumping around in time

GUIs

Mercurial:

- http://mercurial.selenic.com/wiki/OtherTools
- http://tortoisehg.bitbucket.org/

Git:

- http://git-scm.com/downloads/guis
- http://gitx.frim.nl/

hg Commands to Join a Shared Project

```
$ hg clone project_repo
$ cd project

    edit some files

$ hg commit -m "My changes."
$ hg push
```

project_repo can be a path, or a URL (http, https, ssh)

git Commands to Join a Shared Project

```
$ git clone project_repo
$ cd project

    edit some files

$ git add <files>
$ git commit -m "My changes."
$ git push
```

project_repo can be a path, or a URL (http, https, ssh)

Collaboration

Mercurial has a built-in web server

```
$ hg serve
```

Okay for quick, ad-hoc repo sharing
A little more complicated if you need 24/7/365 uptime

Git has instaweb and daemon commands
but they are more complicated right from the start

Bitbucket and GitHub

https://bitbucket.org/

- Mercurial or Git
- Free unlimited public repos
- Free private repos with 5-8 collaborators; unlimited with educational identity
- Issue trackers, wikis
- Forking, pull requests

https://github.com/

- Git only
- Free unlimited public repos
- Monthly fee for private repos
- Issue trackers, wiki
- Forking, pull requests
- More buzz

Bitbucket and GitHub

Getting Started Guides:

Bitbucket 101

GitHub Bootcamp

Part 2 - Software Automation

- Python and the scientific Python stack
- SoG-bloomcast - an automation example
- Requests - HTTP for humans
- Parsing web data - XML, HTML, CSV, netCDF, GIS
- Spawning sub-processes
- Vectorized and N-dimensional array calculations
- Graphs and figures
- String interpolation and templating
- Shell scripts and cron jobs

Python

- http://python.org
- Created in 1989 by Guido van Rossum
- Clear, readable syntax
- General purpose language
- Well documented, free, and cross-platform
- Expressive
- Dynamic execution
- Very high level, dynamic data types
- Extensive standard library, and ecosystem of 3rd-party packages
- Easily extended in C and C++

Python for Engineering & Science

- http://scipy.org
- NumPy - N-dimensional arrays
- SciPy - Library of fundamental scientific algorithms (in many cases just Python wrappers around time-tested Fortran and C implementations)
- Matplotlib - 2D plotting
- IPython Notebook - enhanced Python shell in the browser with rich text, math notation, inline plots, ...
- The list goes on...
- Curated distributions:
  - Anaconda from Continuum Analytics
  - Canopy from Enthought

SoG-Bloomcast - An Automation Example

Daily, operational forecast of the 1st spring phytoplankton bloom in the Strait of Georgia:

1. Get near real-time forcing data from web services
    - wind, weather, river flows
2. Process forcing data into format for model input
3. Run the SOG model 3 (or 30+) times concurrently
4. Analyze the run results to calculate the forecast bloom date as well as early and late bounds
5. Create time series and depth profile plots
6. Render a results commentary and the plots as an HTML page via a template
7. Push the HTML page to a web site

Do all of that while I get on with other research!

SoG-Bloomcast - An Automation Example

Daily, operational forecast of the 1st spring phytoplankton bloom in the Strait of Georgia:

1. Get near real-time forcing data from web services
   - ○ wind, weather, river flows
2. Process forcing data into format for model input
3. Run the SOG model 3 (or 30+) times concurrently
4. Analyze the run results to calculate the forecast bloom date as well as early and late bounds
5. Create time series and depth profile plots
6. Render a results commentary and the plots as an HTML page via a template
7. Push the HTML page to a web site

Do all of that while I get on with other research!

Requests - HTTP for Humans

http://docs.python-requests.org/en/latest/

```
url = 'http://climate.weather.gc.ca/climateData/...
params = {
   'station_id': 6831,
   'format': 'xml',
   'Year': 2014,
   'Month': 3,
   'Day': 29,
   ...
}
response = requests.get(url, params=params)
print(response.text)
```

Requests - With Session Data

```
with requests.session() as s:
   s.post(disclaimer_url, data='I Agree')
   time.sleep(5)
   response = s.get(data_url, params=params)
print(response.text)
```

SoG-Bloomcast - An Automation Example

Daily, operational forecast of the 1st spring phytoplankton bloom in the Strait of Georgia:

1. Get near real-time forcing data from web services
   - wind, weather, river flows
2. Process forcing data into format for model input
3. Run the SOG model 3 (or 30+) times concurrently
4. Analyze the run results to calculate the forecast bloom date as well as early and late bounds
5. Create time series and depth profile plots
6. Render a results commentary and the plots as an HTML page via a template
7. Push the HTML page to a web site

Do all of that while I get on with other research!

Data Processing & Transformation

- XML
  - Python standard library: xml.etree.ElementTree
  - lxml (if you need to do lots, and do it faster)
- HTML (web scraping)
  - BeautifulSoup
  - scrapy
- CSV
  - numpy.genfromtxt
- netCDF
  - python-netCDF4
- GIS
  - GDAL/OGR Bindings

SoG-Bloomcast - An Automation Example

Daily, operational forecast of the 1st spring phytoplankton bloom in the Strait of Georgia:

1. Get near real-time forcing data from web services
   - wind, weather, river flows
2. Process forcing data into format for model input
3. Run the SOG model 3 (or 30+) times concurrently
4. Analyze the run results to calculate the forecast bloom date as well as early and late bounds
5. Create time series and depth profile plots
6. Render a results commentary and the plots as an HTML page via a template
7. Push the HTML page to a web site

Do all of that while I get on with other research!

Subprocess Module

```
cmd = 'nice -n 19 SOG < infile > outfile 2>&1'

proc = subprocess.Proc(cmd, shell=True)

while True:
   if proc.poll() is None:
     time.sleep(30)
   else:
     print('Done!)
     break
```

SoG-Bloomcast - An Automation Example

Daily, operational forecast of the 1st spring phytoplankton bloom in the Strait of Georgia:

1. Get near real-time forcing data from web services
   - wind, weather, river flows
2. Process forcing data into format for model input
3. Run the SOG model 3 (or 30+) times concurrently
4. Analyze the run results to calculate the forecast bloom date as well as early and late bounds
5. Create time series and depth profile plots
6. Render a results commentary and the plots as an HTML page via a template
7. Push the HTML page to a web site

Do all of that while I get on with other research!

Vector and Array Calculations

Lots of libraries for doing scientific calculations

NumPy is generally the foundation

For specific application areas and algorithms:

- SciPy
- Pandas
- SciKits

SoG-Bloomcast - An Automation Example

Daily, operational forecast of the 1st spring phytoplankton bloom in the Strait of Georgia:

1. Get near real-time forcing data from web services
   - wind, weather, river flows
2. Process forcing data into format for model input
3. Run the SOG model 3 (or 30+) times concurrently
4. Analyze the run results to calculate the forecast bloom date as well as early and late bounds
5. Create time series and depth profile plots
6. Render a results commentary and the plots as an HTML page via a template
7. Push the HTML page to a web site

Do all of that while I get on with other research!

Matplotlib

```
fig, ax_left = matplotlib.pyplot.subplots(1, 1)
ax_right = ax_left.twinx()
ax_left.plot(
   nitrate.time,
   nitrate.values,
   color='blue')
ax_right.plot(
   diatoms.time,
   diatoms.values,
   color='green')
ax_left.set_ytitle('Nitrate Concentration [uM N]')
ax_right.set_ytitle('Diatom Biomass [uM N]')
ax_left.set_xtitle('Year Day in 2014')

fig.savefig('nitrate_diatoms_timeseries.png')
```

SoG-Bloomcast - An Automation Example

Daily, operational forecast of the 1st spring phytoplankton bloom in the Strait of Georgia:

1. Get near real-time forcing data from web services
    - wind, weather, river flows
2. Process forcing data into format for model input
3. Run the SOG model 3 (or 30+) times concurrently
4. Analyze the run results to calculate the forecast bloom date as well as early and late bounds
5. Create time series and depth profile plots
6. Render a results commentary and the plots as an HTML page via a template
7. Push the HTML page to a web site

Do all of that while I get on with other research!

String Interpolation & Templating

```
page_tmpl = """
<h1>Strait of Georgia Spring Bloom Prediction</h1>

The median bloom date calculate from a
{member_count} ensemble forecast is
{bloom_dates['median]:%Y-%m-%d}
...
"""
page = page_tmpl.format(
    member_count=len(members),
    bloom_dates=bloom_dates,
    ...
)
with open('page.html', 'rt') as f:
    f.write(page)
```

String Interpolation & Templating

Templating libraries:

- Mako
- Jinja2
- many more

SoG-Bloomcast - An Automation Example

Daily, operational forecast of the 1st spring phytoplankton bloom in the Strait of Georgia:

1. Get near real-time forcing data from web services
   - ○ wind, weather, river flows
2. Process forcing data into format for model input
3. Run the SOG model 3 (or 30+) times concurrently
4. Analyze the run results to calculate the forecast bloom date as well as early and late bounds
5. Create time series and depth profile plots
6. Render a results commentary and the plots as an HTML page via a template
7. Push the HTML page to a web site

Do all of that while I get on with other research!

Subprocess (again)

rsync, scp, sftp, hg, git, …

```
cmd = [
    'rsync', '-Rtvhz',
    '{}/./{}'.format(html_path, results_page),
    'shelob:/www/salishsea/data/'
]
subprocess.check_call(cmd)
```

SoG-Bloomcast - An Automation Example

Daily, operational forecast of the 1st spring phytoplankton bloom in the Strait of Georgia:

1. Get near real-time forcing data from web services
   - wind, weather, river flows
2. Process forcing data into format for model input
3. Run the SOG model 3 (or 30+) times concurrently
4. Analyze the run results to calculate the forecast bloom date as well as early and late bounds
5. Create time series and depth profile plots
6. Render a results commentary and the plots as an HTML page via a template
7. Push the HTML page to a web site

Do all of that while I get on with other research!

Shell Script and Cron Job

```
# cron script to run SoG-bloomcast
#
# make sure that this file has mode 744
# and that MAILTO is set in crontab

VENV=/data/dlatorne/.virtualenvs/bloomcast
RUN_DIR=/data/dlatorne/SOG-projects/SoG-bloomcast/run
. $VENV/bin/activate && cd $RUN_DIR && \
   $VENV/bin/bloomcast config.yaml
```

```
MAILTO=dlatorne@eos.ubc.ca

BLOOMCAST_DIR=/data/dlatorne/SOG-projects/SoG-bloomcast

# m h  dom mon dow   command
  0 9   *   *   *    $BLOOMCAST_DIR/cronjob.sh
```

Resources

- software-carpentry.org
- UBC EOAS Software Carpentry Bootcamp
- Salish Sea MEOPAR Project on Bitbucket
- Salish Sea MEOPAR Project Tools Documentation
- douglatornell.ca