LECTURE 6
SCRIPTS AND AUTOMATION

Fundamentals of Programming - COMP1005

Department of Computing
Curtin University
Updated: 8/4/19

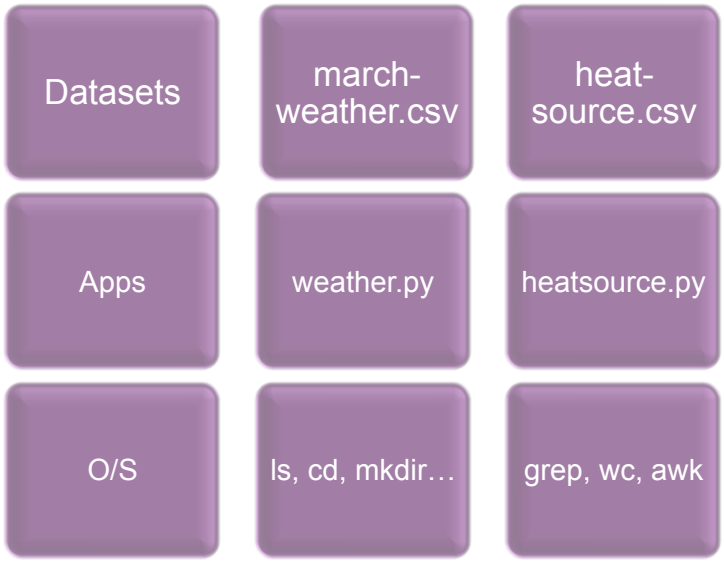
Copyright Warning

Learning Outcomes

- Understand and use Unix (bash) commands and scripts to automate workflows
- Understand and use Python as a scripting language
- Understand and modify supplied Python code to automate experimental workflows
- Apply Python scripts to implement a parameter sweep using an existing program

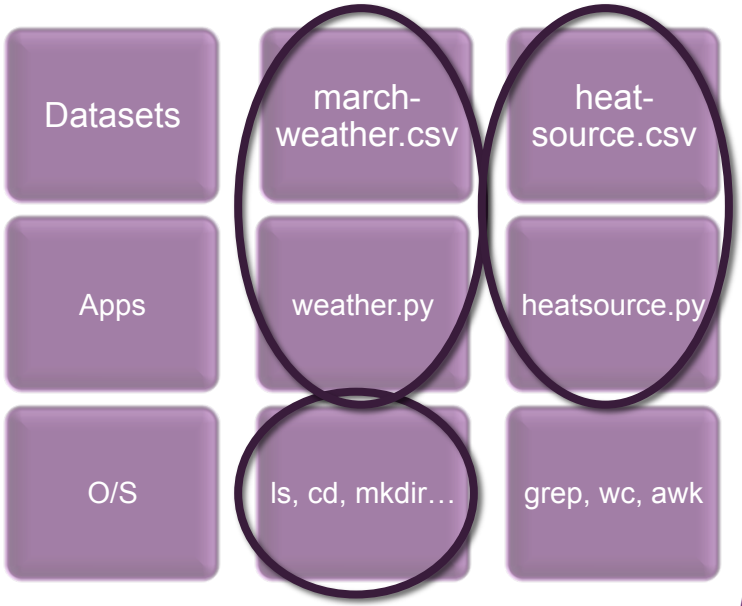
The story so far…

- We started by learning enough Unix/Linux to move around directories and zip up our files
- We then wrote programs that read from the keyboard and wrote to the screen
- We generated plots
- Most recently, we've broken the confines of the program to read and write files



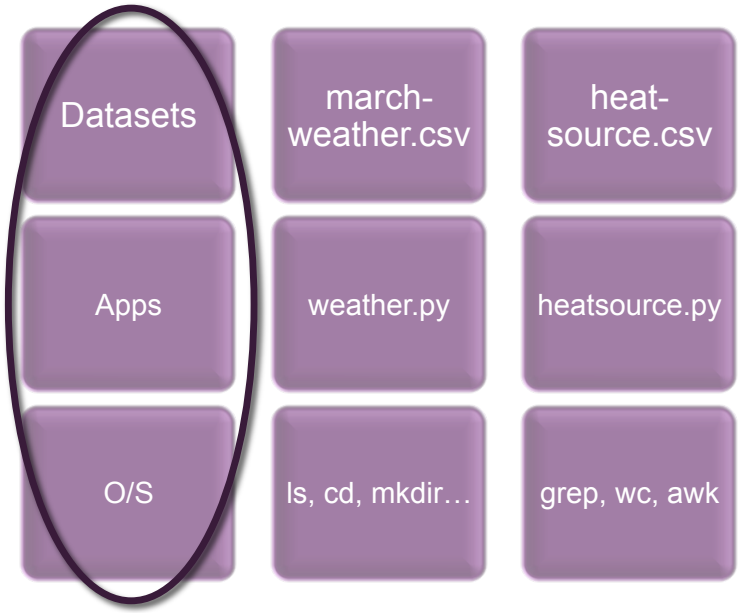
The story so far…

- We started by learning enough Unix/Linux to move around directories and zip up our files
- We then wrote programs that read from the keyboard and wrote to the screen
- We generated plots
- Most recently, we've broken the confines of the program to read and write files



The story so far…

- We haven't really seen the power of Unix
- We can use scripts to use operating system utilities like any other function in a program
- This can be done using bash scripts (or other shells)
- Python is also a scripting language – so it can also bridge from O/S to code to data



WORKFLOWS

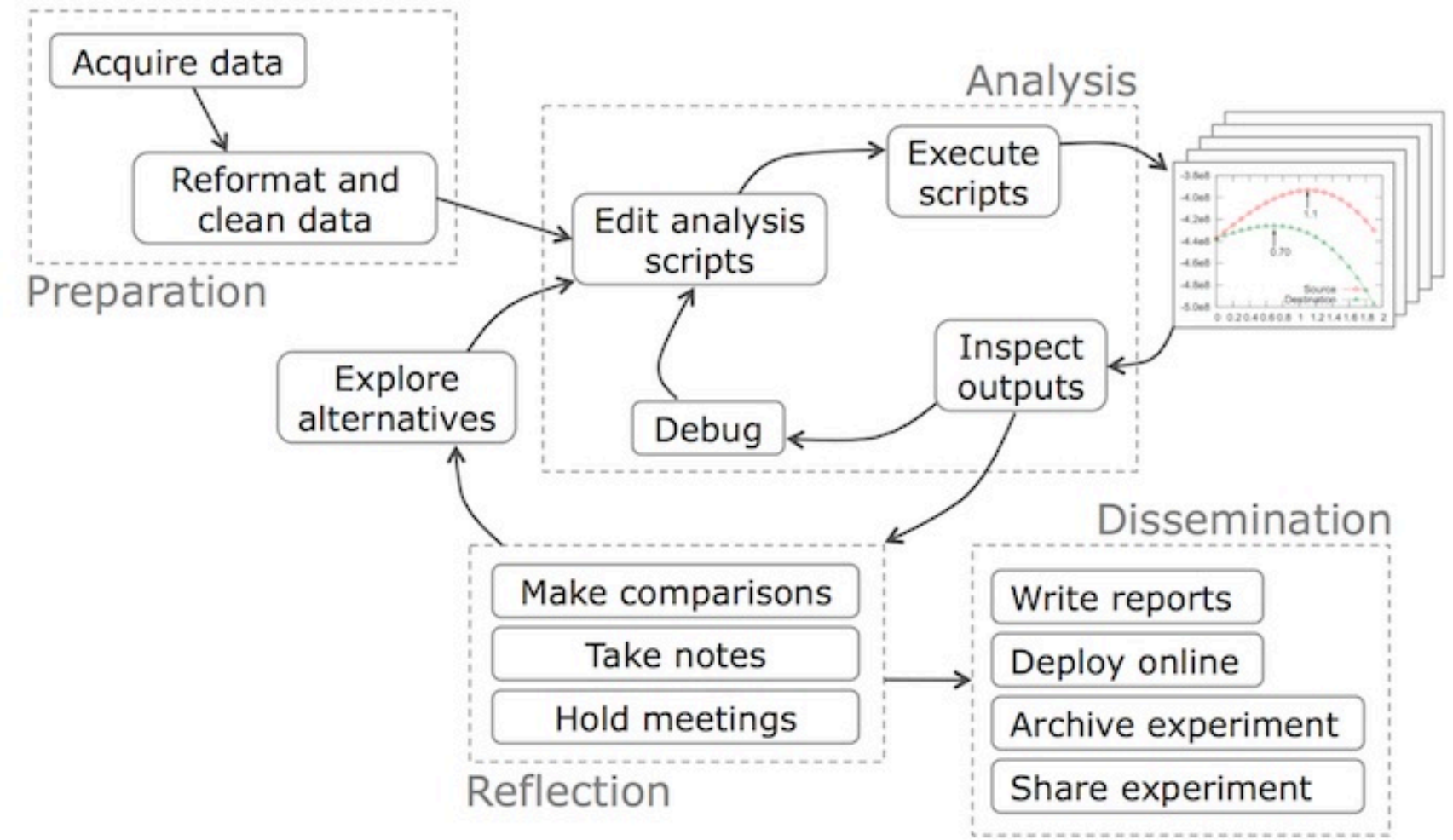
Fundamentals of Programming

Lecture 6

Workflows

- In the first lecture we talked about having a program that would:
  - Read input
  - Do processing
  - Write output
- In scientific applications, that program becomes one (potentially reusable) block
- These building blocks can then be put together into an overall workflow

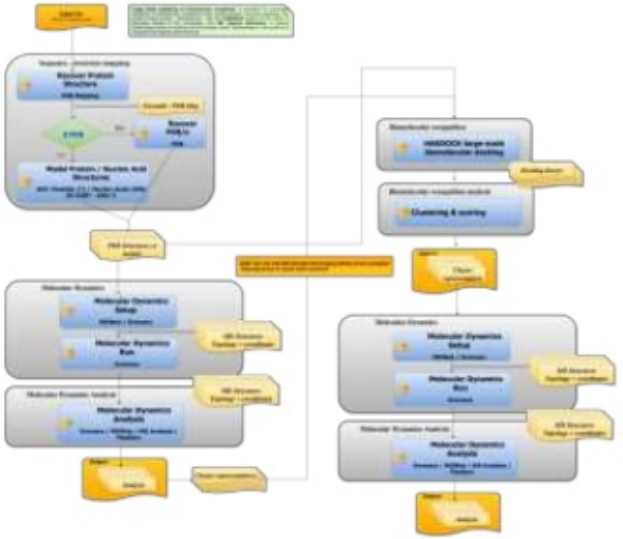
Data Science Workflow



 http://cacm.acm.org/blogs/blog-cacm/169199-data-science-workflow-overview-and-challenges/fulltext

What is a Workflow?

With thanks to Bertram Ludascher: WORKS 2015 Keynote



Automation

- Automate computational aspects
- Repetitive pipelines, sweep campaigns

Scaling – compute cycles

- Make use of computational infrastructure & handle large data

Abstraction – people cycles

- Shield complexity and incompatibilities
- Report, re-use, evolve, share, compare
- Repeat – Tweak - Repeat
- First class commodities

Provenance - reporting

- Capture, report and utilize log and data lineage auto-documentation
- Traceable evolution, audit, transparency
- Compare

Findable
Accessible
Interoperable
Reusable
(**R**eproducible)

https://www.slideshare.net/carolegoble/advances-in-scientific-workflow-environments

Categorising Workflows

Instrument pipelines

+

Data wrangling & analytics

+

Simulations

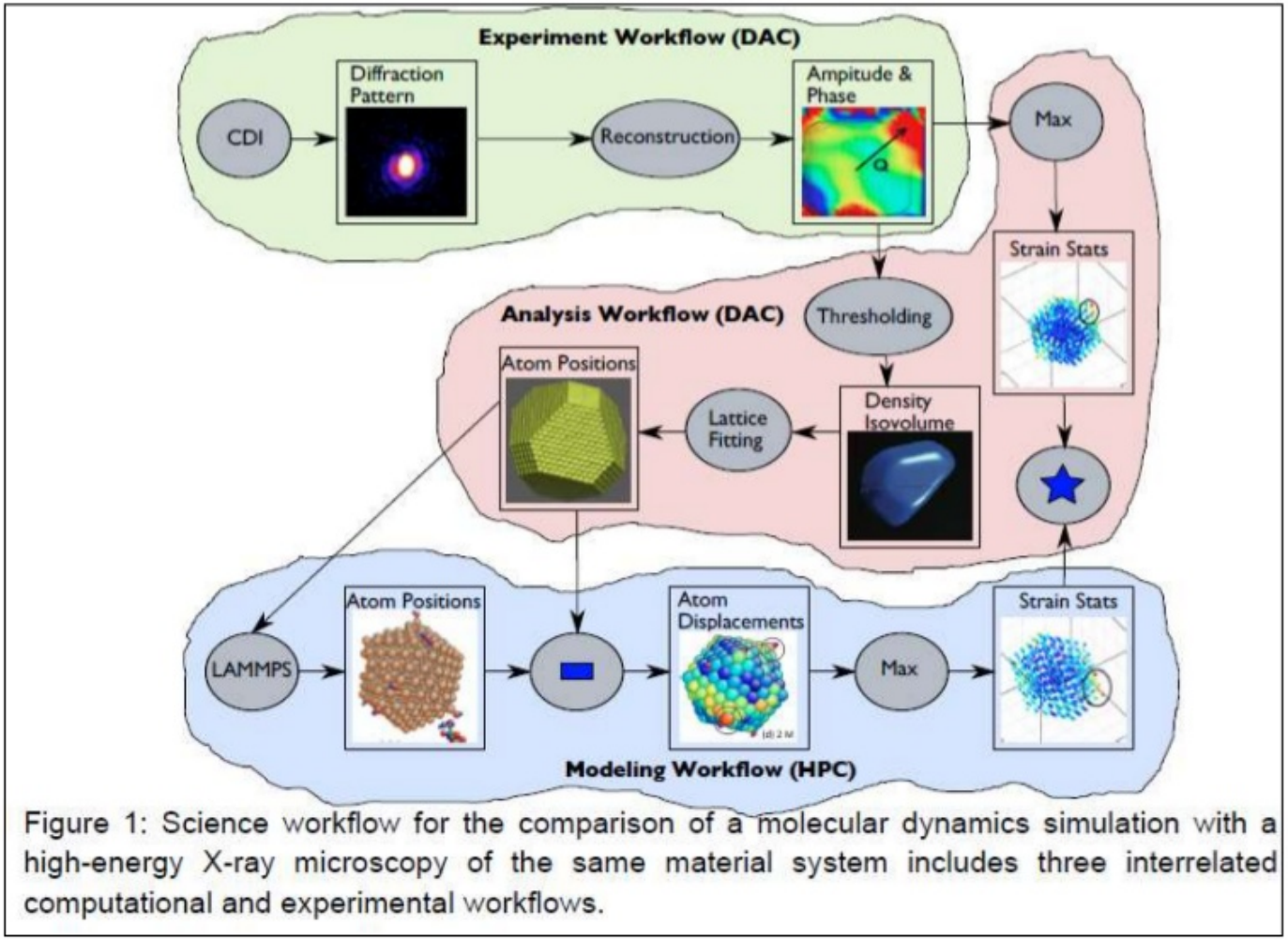
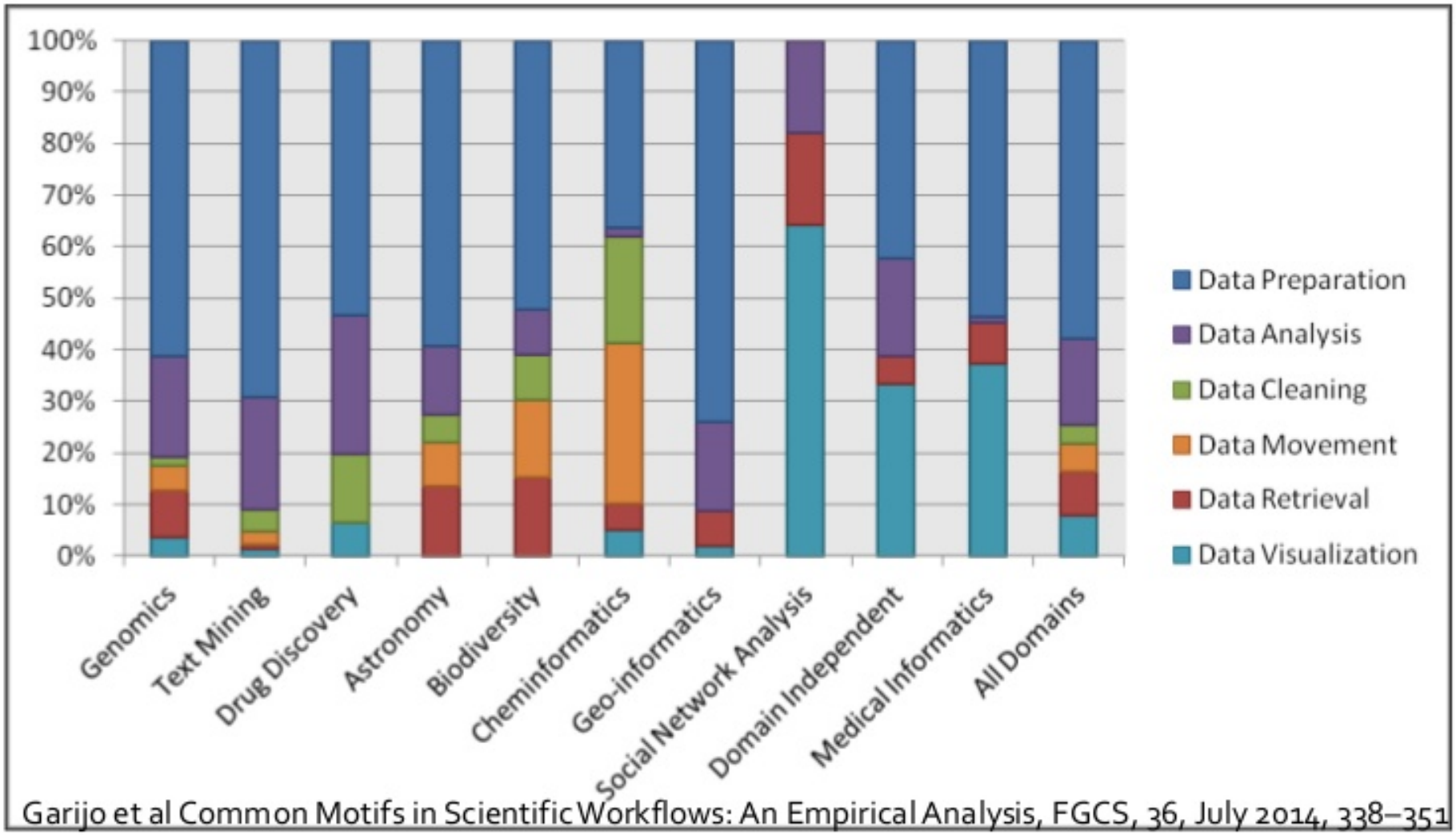


Figure 1: Science workflow for the comparison of a molecular dynamics simulation with a high-energy X-ray microscopy of the same material system includes three interrelated computational and experimental workflows.

http://tpeterka.github.io/maui-project/

https://www.slideshare.net/carolegoble/advances-in-scientific-workflow-environments

Workflow Activities



Garijo et al Common Motifs in Scientific Workflows: An Empirical Analysis, FGCS, 36, July 2014, 338–351

https://www.slideshare.net/carolegoble/advances-in-scientific-workflow-environments

Weather Workflow

- We've worked through a simple workflow in the pracs…
  ... as it goes in a straight line (no cycles)
  this is also a "pipeline"

UNIX POWER TOOLS

Unix Power Tools

- So far, Unix may seem a little lame
- We're using a very small subset of the commands available
- Once we have a few more commands and a way to put them together – we can harness the…

POWER OF UNIX!!!

Useful commands

Command	Description	Command	Description
cd [dirname]	Change directory to “dirname”	cp [src] [dest]	Copy file from src to dest
ls [dirname]	List contents of directory “dirname”	mv [src] [dest]	Move file from src to dest
mkdir [dirname]	Make directory	rm [filename(s)]	Delete file “filename”
rmdir [dirname]	Remove directory (only if empty)	cat [filename(s)]	Concatenate files (print to screen)
pwd	Print working (current) directory	less [filename]	Print file to screen one page at a time
man [cmd]	View manual for “cmd”	wc [filename]	Output word count for “filename”
history	List recent commands	head [filename]	Print first 10 lines
grep [string] [file]	Print out lines with “string” from “file”	tail [filename]	Print last 10 lines

Piping

- Pipes are a UNIX feature which allows you to connect several commands together
- The output of one command becomes the input to the next
- Most UNIX commands get input from stdin and pass output to stdout
- The pipe symbol "|" directs UNIX to connect stdout from the first command to the stdin of the second command
  - cat /etc/motd | wc
- **>** will redirect the output to a file, **>>** will append to an existing file, **<** will redirect input from a file
  - history > hist.txt
  - grep Curtin /etc/motd >> Curtin_motd.txt
  - python awesome.py < inputfile.txt

Weather Workflow

- How could we do this with Unix tools?
- wget – gets files from the web/Internet
- grep – prints out lines matching a pattern
- awk – filters a file by fields
- gnuplot – allows command line plotting

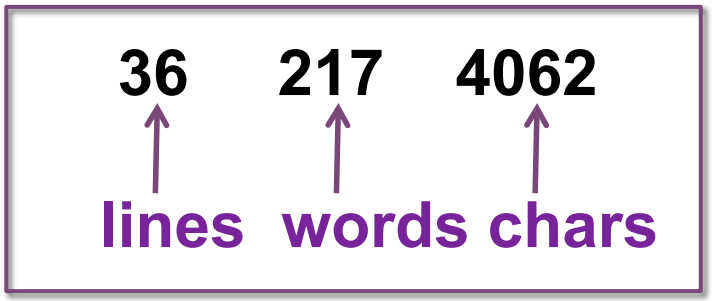
Workflow – Download and Wordcount

> wget http://www.bom.gov.au/climate/dwo/201902/text/IDCJDW6111.201902.csv

```
--2019-04-06 23:21:10--  http://www.bom.gov.au/climate/[..]/IDCJDW6111.201902.csv
Resolving www.bom.gov.au... 23.48.222.34
Connecting to www.bom.gov.au|23.48.222.34|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4062 (4.0K) [text/plain]
Saving to: 'IDCJDW6111.201902.csv.1'

IDCJDW6111.201902.csv.1         100%
[=====================================>]   3.97K  --.-KB/s    in 0.007s

2017-04-06 23:21:11 (551 KB/s) - 'IDCJDW6111.201902.csv.1' saved [4062/4062]
```

> wc IDCJDW6111.201902.csv

```
 36     217    4062
```



Workflow – Check File Format

$ head IDCJDW6111.201902.csv

"Daily Weather Observations for Perth, Western Australia for February 2019"

"Prepared at 16:08 GMT on Sunday 2 April 2019 IDCJDW6111.201902"

"Copyright 2003 Commonwealth Bureau of Meteorology"

"A combination of observations from Mount Lawley and Perth Airport."

"Significant variations can be experienced across the Perth metropolitan area on individual days. [..] Some cloud observations are from automated equipment, these are somewhat different to those made by a human observer and may not appear every day."

"Temperature, humidity, wind, pressure and rainfall observations are from Perth Metro {station 009225}"

"Cloud, evaporation and sunshine observations are from Perth Airport {station 009021}"

head = first 10 lines
head -n = first n lines

,"Date","Minimum temperature (?C)","Maximum temperature (?C)","Rainfall (mm)","Evaporation (mm)","Sunshine (hours)","Direction of maximum wind gust ","Speed of maximum wind gust (km/h)","Time of maximum wind gust","9am Temperature (?C)","9am relative humidity (%)","9am cloud amount (oktas)","9am wind direction","9am wind speed (km/h)","9am MSL pressure (hPa)","3pm Temperature (?C)","3pm relative humidity (%)","3pm cloud amount (oktas)","3pm wind direction","3pm wind speed (km/h)","3pm MSL pressure (hPa)"

,2019-02-1,16.5,30.3,2.8,1.0,11.2,ESE,28,11:17,22.9,64,1,ESE,9,1017.9,28.7,54,2,E,13,1015.7

Workflow – Check File Format

$ tail IDCJDW6111.201902.csv

,**2019-02**-19,23.4,37.7,0,11.0,12.3,W,33,13:07,30.1,33,0,NE,17,1014.3,34.2,38,1,NW,
13,1011.4
,**2019-02**-20,20.4,26.4,0,8.8,11.3,SW,35,16:41,22.9,65,3,SSW,
11,1013.4,23.0,56,4,WSW,17,1010.7
,**2019-02**-21,16.5,22.1,2.4,12.0,10.2,WSW,48,09:50,19.1,57,7,SW,
26,1014.3,21.3,45,4,SW,22,1015.5
,**2019-02**-22,10.1,24.1,0,7.0,11.9,SSW,33,16:07,19.0,59,3,SE,
13,1021.9,22.8,49,1,SSW,17,1019.8
,**2019-02**-23,12.7,30.6,0,7.8,11.9,SSW,31,16:07,22.2,58,1,E,7,1021.9,29.1,42,0,SSW,
13,1018.7
,**2019-02**-24,16.2,36.5,0,9.6,12.1,NE,33,09:35,26.4,42,0,ENE,
17,1020.6,35.3,18,0,ENE,13,1016.8
,**2019-02**-25,18.3,39.3,0,9.8,12.1,NE,31,08:16,29.9,28,0,NE,15,1015.5,39.1,12,0,SE,
7,1011.3
,**2019-02**-26,18.0,29.4,0,10.8,12.1,SW,30,18:45,24.0,53,1,SSW,
13,1011.7,26.5,53,1,WSW,13,1011.6
,**2019-02**-27,15.7,29.8,0,5.8,11.8,SW,33,16:59,23.4,55,1,S,6,1015.3,28.5,47,1,WSW,
13,1013.5
,**2019-02**-28,17.9,33.0,0,7.4,11.9,SW,24,13:17,25.5,57,1,ESE,
7,1014.8,31.0,45,3,WSW,13,1012.3

tail = last 10 lines
tail -n = last n lines

note that "2019-02-" is in every line that we want

Workflow – More or Less

$ more IDCJDW6111.201902.csv

,**2019-02**-19,23.4,37.7,0,11.0,12.3,W,33,13:07,30.1,33,0,NE,
17,1014.3,34.2,38,1,NW,13,1011.4
,**2019-02**-20,20.4,26.4,0,8.8,11.3,SW,35,16:41,22.9,65,3,SSW,
11,1013.4,23.0,56,4,WSW,17,1010.7
,**2019-02**-21,16.5,22.1,2.4,12.0,10.2,WSW,48,09:50,19.1,57,7,SW,
26,1014.3,21.3,45,4,SW,22,1015.5

Scroll through file one page at a time, press space or 'n' to continue. 'q' to exit

$ less IDCJDW6111.201902.csv

,**2019-02**-19,23.4,37.7,0,11.0,12.3,W,33,13:07,30.1,33,0,NE,
17,1014.3,34.2,38,1,NW,13,1011.4
,**2019-02**-20,20.4,26.4,0,8.8,11.3,SW,35,16:41,22.9,65,3,SSW,
11,1013.4,23.0,56,4,WSW,17,1010.7
,**2019-02**-21,16.5,22.1,2.4,12.0,10.2,WSW,48,09:50,19.1,57,7,SW,
26,1014.3,21.3,45,4,SW,22,1015.5

Scroll through file one page at a time. Go backwards ('b') as well

Workflow - Filtering

> grep **2019-02-** IDCJDW6111.201902.csv > data.csv

- *reads through data file and prints out those including the string "2017-02-" – redirect output to data.csv*

> awk **-F ","** '{print **$3, $4, $11, $17**}' data.csv > data4.csv

- *reads through data.csv and prints fields 3,4,11 & 17, using "," as the field separator (-F ",")*
- *counts fields from 1*
- *redirects output to data4.csv*
- *could also use "cut"*

Workflow - Plotting

> gnuplot plotcmd.txt

- runs gnuplot with the plotting commands in plotcmd.txt

plotcmd.txt:

```
plot for [col=1:4] './data4.csv' using 0:col with lines
```

- plots four lines from columns 1 to 4 in data4.csv
- x values using defaults
- can also add labels and titles etc.

Workflow - Complete

> wget http://www.bom.gov.au/climate/dwo/201902/text/IDCJDW6111.201902.csv

> grep 2019-02- IDCJDW6111.201902.csv > data.csv

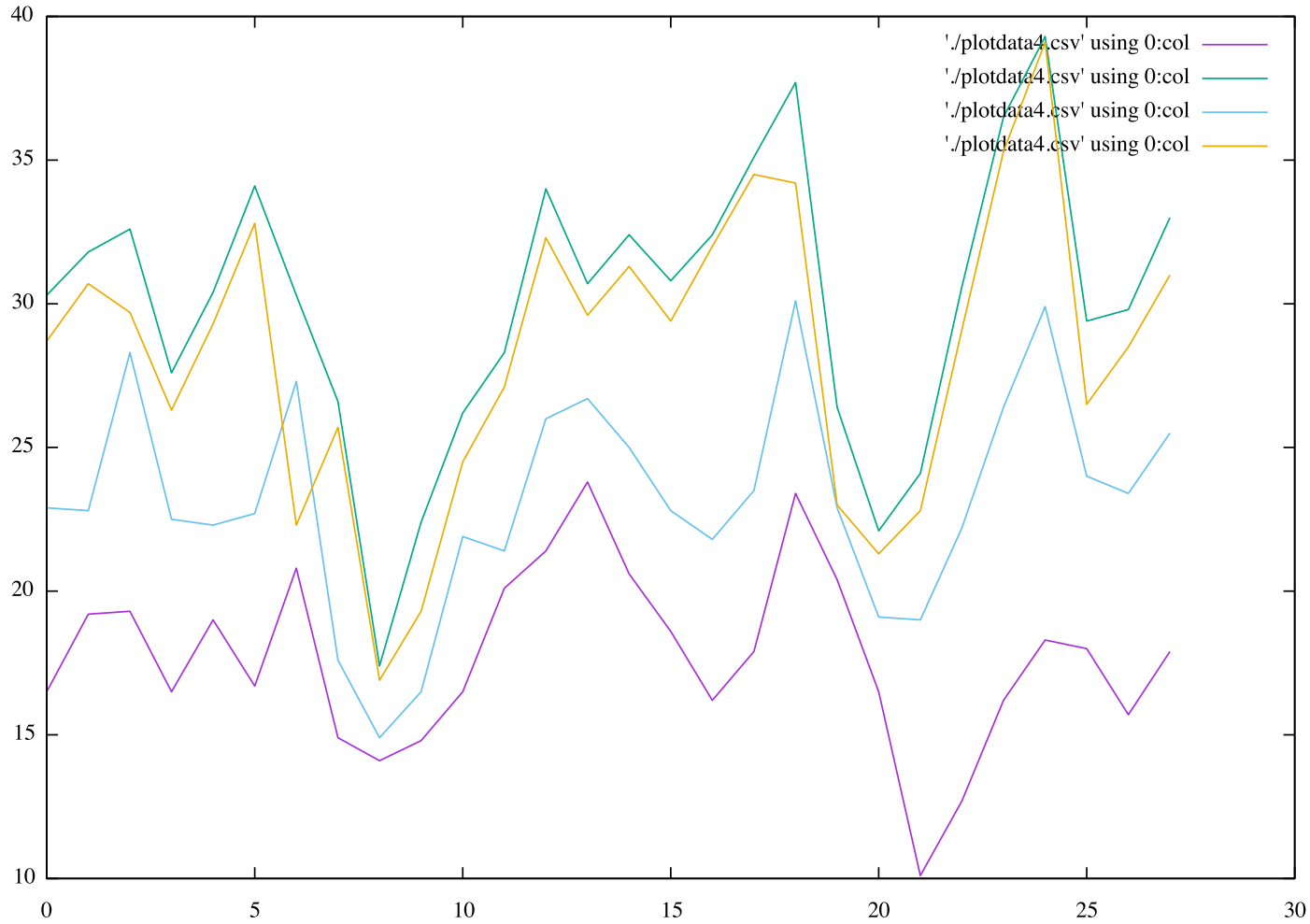
> awk -F "," '{print $3, $4, $11, $17}' data.csv > data4.csv

> gnuplot plotcmd.txt

plotcmd.txt:

```
plot for [col=1:4] './data4.csv' using 0:col with lines
```

Workflow - Plot



More Compact Version with Pipes

```
$ wget ….
$ grep 2019-02- infile.csv | awk -F "," '{print $3, $4, $11, $17}' >
plotdata4.csv
$ gnuplot plotcmd.txt
```

BASH SCRIPTS

Bash Scripts

- Bundle up repeated commands into a script
- Work through it step by step, then save to a file (history -10 > hist.txt)
- Run with "bash plotting.sh"

plotting.sh

```
cd ~/ScriptsAuto/
mkdir exp1
cp plotcmd.txt exp1
cd exp1
mkdir run1
mkdir run2
wget http://www.bom.gov.au/climate/dwo/201902/text/IDCJDW6111.201902.csv
grep 2019-02- IDCJDW6111.201902.csv > data.csv
awk -F "," '{print $3, $4, $11, $17}' data.csv > data4.csv
gnuplot plotcmd.txt
```

Looping and command line args

- Can take command line arguments to customise scripts
  - e.g. $1, $2 ($0 is the original command)

middle.sh:

```
# Select lines from the middle of a file.
# Usage: bash middle.sh filename end_line num_lines
head -n "$2" "$1" | tail -n "$3"
```

- Loop through numbers of times or through sequences
  - $@ is all cmd line args

do-stats.sh:

```
# Calculate reduced stats for data files at J = 100 c/bp.
for datafile in "$@"
do
    echo $datafile
    bash goostats -J 100 -r $datafile stats-$datafile
done
```

PYTHON MODULES AND SCRIPTS

Modules

- We've imported modules and packages to gain access to functions written by others
- We've created functions inside our programs to reuse throughout the programs
- If we want to reuse the functions in many programs…
  - Create our own module
  - Import the module into our programs

Module textfun.py

- We can create a module with some text-related function – textfun.py
- In it we will create some methods:
  - novowels(inString)
  - reverseupper(inString)
  - upperskip2(inString)
- To use our functions, we can add:
  - import textfun

    …at the start of out programs

Module textfun.py

```
#
# textfun.py – module of text-related functions
#
vowels = 'aeiouAEIOU'

def novowels(inString):
    outString=''
    for i in inString:
        if not i in vowels:
            outString = outString + i
    return outString

def reverseupper(inString):
    return(inString[::-1].upper())
```

Test Program – testing.py

testing.py:

```
#
# testing.py – test program for textfun.py
#
import textfun

testString = 'helloHELLO'

print(textfun.novowels(testString))
print(textfun.reverseupper(testString))
```

```
> python testing.py
hllHLL
OLLEHOLLEH
```

Packages

- If we had a group of related modules, we could group them in a package
- The module files are placed in a directory together
  - A special file, __init__.py indicates the directory is a package
- So, if we had modules **textfun.py** and **numfun.py**. we might group them in a package fun
- Then we could import them using:

```
import fun.textfun as tfun
import fun.numfun as nfun
```

Scipy.ndimage package

```
$ ls anaconda3/pkgs/scipy-0.18.1-np111py35_0/lib/python3.5/site-packages/
scipy

BENTO_BUILD.txt   _build_utils     linalg          sparse
HACKING.rst.txt   _lib             linalg.pxd      spatial
INSTALL.rst.txt   cluster          misc            special
LICENSE.txt       constants        ndimage         stats
THANKS.txt        fftpack          odr             version.py
__config__.py     integrate        optimize
__init__.py       interpolate      setup.py
__pycache__       io               signal

$ ls anaconda3/pkgs/scipy-0.18.1-np111py35_0/lib/python3.5/site-packages/
scipy/ndimage/

__init__.py              filters.py              morphology.py
__pycache__              fourier.py              setup.py
_nd_image.so             interpolation.py        tests
_ni_label.so             io.py
_ni_support.py           measurements.py
```

Paths

- A filesystem is big, really big.
- Modules and Packages need to be located quickly by Python
- We *could* have lots of modules in the local directory '.' – which would get messy
- The operating system has a "PATH" variable to give it a list of directories to search through
- Part of the installation of a program **updates the path** to include the new program
- **Anaconda** looks after this for us.

```
[12345678@saeshell01p ~]$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/
sbin:/sbin:/opt/anaconda/bin:/home/12345678/.local/
bin:/home/12345678/bin
```

__main__

- Python scripts and python modules are just scripts
- Python provides a way to tell if you are running code directly (python3) or just using the functions (import)
- If the code is called directly, the variable __name__ will equal "__main__"
- Otherwise __name__ will refer to the calling code
- We can include an if statement to check for this and run specific code
- Any other code that's not in a function definition will run at import time/run time
  - e.g. vowels = 'aeiouAEIOU'

textfun.py

```
def reverseupper(inString):
    return(inString[::-1].upper())

def main():
    print('\nTesting textfun.py')
    testString = 'helloHELLO'
    print('\nTest string is: ', testString)
    print('novowels: ', novowels(testString))
    print('reverseupper: ', reverseupper(testString))
    print('Testing complete')

if __name__ == '__main__':
    main()
```

Main as test code for textfun.py

```
$ python textfun.py

Testing textfun.py

Test string is:  helloHELLO
novowels:  hllHLL
reverseupper:  OLLEHOLLEH
Testing complete
```

A good reference:
http://www.scipy-lectures.org/intro/language/reusing_code.html

SYSTEM CALLS AND ARGUMENTS

System calls

- You may wish to work with directories or other aspects of the operating system from within you program
- The os module provides these and other methods…
  - mkdir(string)
  - listdir()
  - chdir(string)
  - getcwd()
  - rename(src,dest)

Working with directories

```
import os

newdir = 'test'

if newdir not in os.listdir():
   os.mkdir(newdir)
os.chdir(newdir)

newdirs = ['test1','test2','test3']

for d in newdirs:
    os.mkdir(d)

print(os.listdir())
```

```
$ python osmod.py
['test1', 'test2', 'test3']
$ ls
osmod.py  test
$ ls test
test1  test2  test3
```

Command line arguments

- Used to add input to your program as you run it
- Often used for filenames
- Arguments are in a list in the **sys.argv** variable

```
import sys

print(sys.argv)

fileo = open(sys.argv[1])

print(fileo.readlines())
```

['arguments.py', 'arguments.txt']

["Man: Ah. I'd like to have an argument, please. \n", 'Receptionist: Certainly sir. Have you been here before? \n', "Man: No, I haven't, this is my first time. \n", 'Receptionist: I see. Well, do you want to have just one argument, or were you thinking of taking a course?\n', 'Man: Well, what is the cost?\n', "Receptionist: Well, It's one pound for a five minute argument, but only eight pounds for a course of ten.\n", 'Man: Well, I think it would be best if I perhaps started off with just the one and then see how it goes.\n', "Receptionist: Fine. Well, I'll see who's free at the moment. \n", '(Pause)']

File Safety

- Files are a major cause of program crashes
- They should be wrapped in lots of checking
- Use the os calls to see if files or directories exist before trying to open them
- Be extra, extra careful before opening a file to write – even your programs could be overwritten!

EXAMPLES

Parameter Sweeps

Data Management

Parameter Sweeps

- Provides a separation of concerns
- Useful for:
  - Finding the optimum value of a parameter
  - For studying the sensitivity of the design performance to certain parameters
  - Running a series of simulations with a set of varying parameters
- Loop through all permutations of the values
- Analyse the results after loops are complete

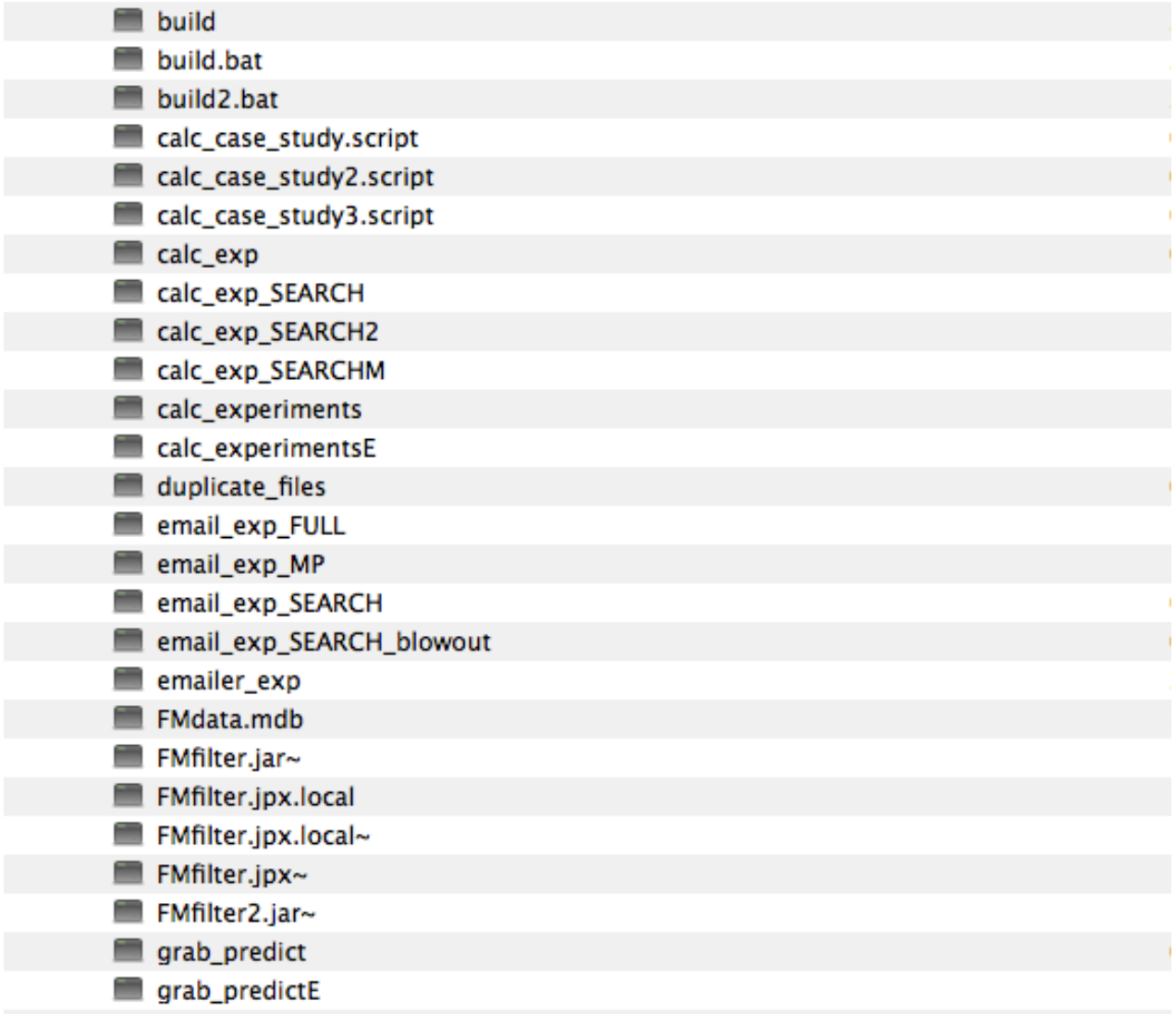
Parameter Sweeps

- Could be linear or logspace values
- May be string values in a list
- Good to have this part of the code controlled through input files or command line arguments
- Can call a python script from a driver bash script to give the parameter sweep
- We'll work on this in the prac

Data Management Example

- Scripts to automate experiments
- Use additional scripts to do multiple runs
- Create directory structure for each experiment and copy supporting files
- Use date and other meaningful information in directory names
- Bundle results for each stage of work – matching “bundle” for code

Scripts for automation…



```
#!/bin/bash

echo
echo "############################################################"
echo
echo "                    CALCULATOR CASE STUDY"
echo
echo "############################################################"
echo

base=$1

./process ${base}1
./process ${base}2
./process ${base}3
./process ${base}4
./process ${base}5

./process ${base}1i2
./process ${base}2i2
./process ${base}3i2
./process ${base}4i2
./process ${base}5i2

./grab_predict $base
```

```
#!/bin/bash

# laptop version 15/6/05
# Re-saved to run on Mac - CR/LF problem, 31/12/08
# Rejigged paths for Mac 31/12/08
# Needed old xerces - see http://archive.apache.org/dist/xml/xerces-j/ for version 1_4_4
# Had to change j48.J48 to J48 for Weka 31/12/08

echo
echo "############################################################"
echo
echo "            INTELLIGENT COMPONENT SELECTION"
echo
echo "############################################################"
echo

# see http://www.tldp.org/LDP/abs/html/io-redirection.html for
# output redirection options


if [ -z "$1" ]; then
  echo "ERROR: No file base given - exiting."
  echo
  echo "Usage: process file_base"
  echo
  exit 1
fi

echo "Setting up directory for running experiment on >> $1 <<"

EXPERIMENT=$1
WORKDIR="${1}_`date +%F_%H_%M`"
TEMPDIR="TEMP_`date +%F_%H_%M`"
WEKAHOME="/Users/valeriemaxville/_Thesis/dev_new/Weka-3-6-0/weka.jar"
#WEKAHOME="/Users/valeriemaxville/_Thesis/dev_new/Weka-3-4-14/weka.jar"
#WEKAHOME="/Users/valeriemaxville/_Thesis/dev_new/weka-3-0-6.jar"
```

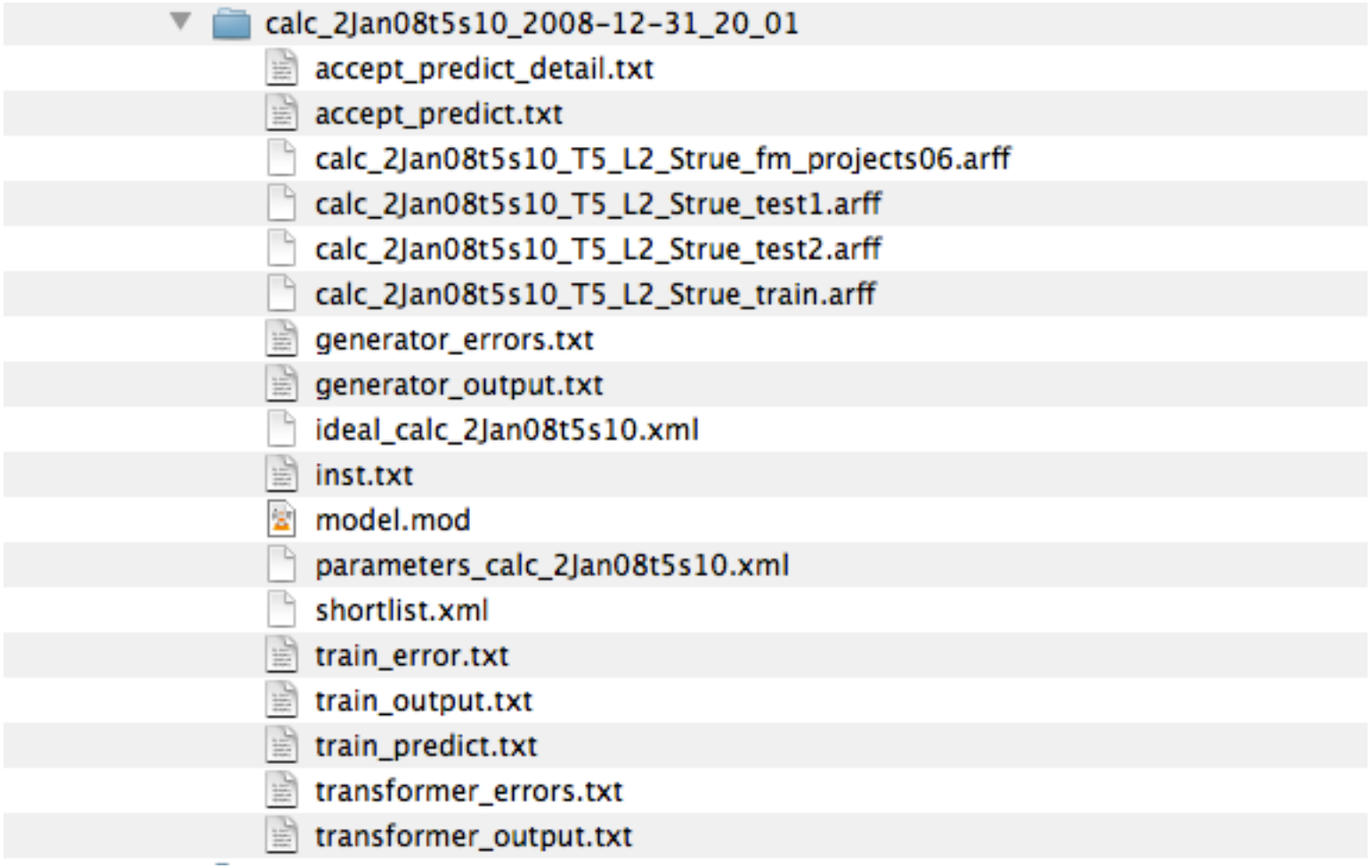
Parameter file…

parameters_email_12Sep07t1s24.xml

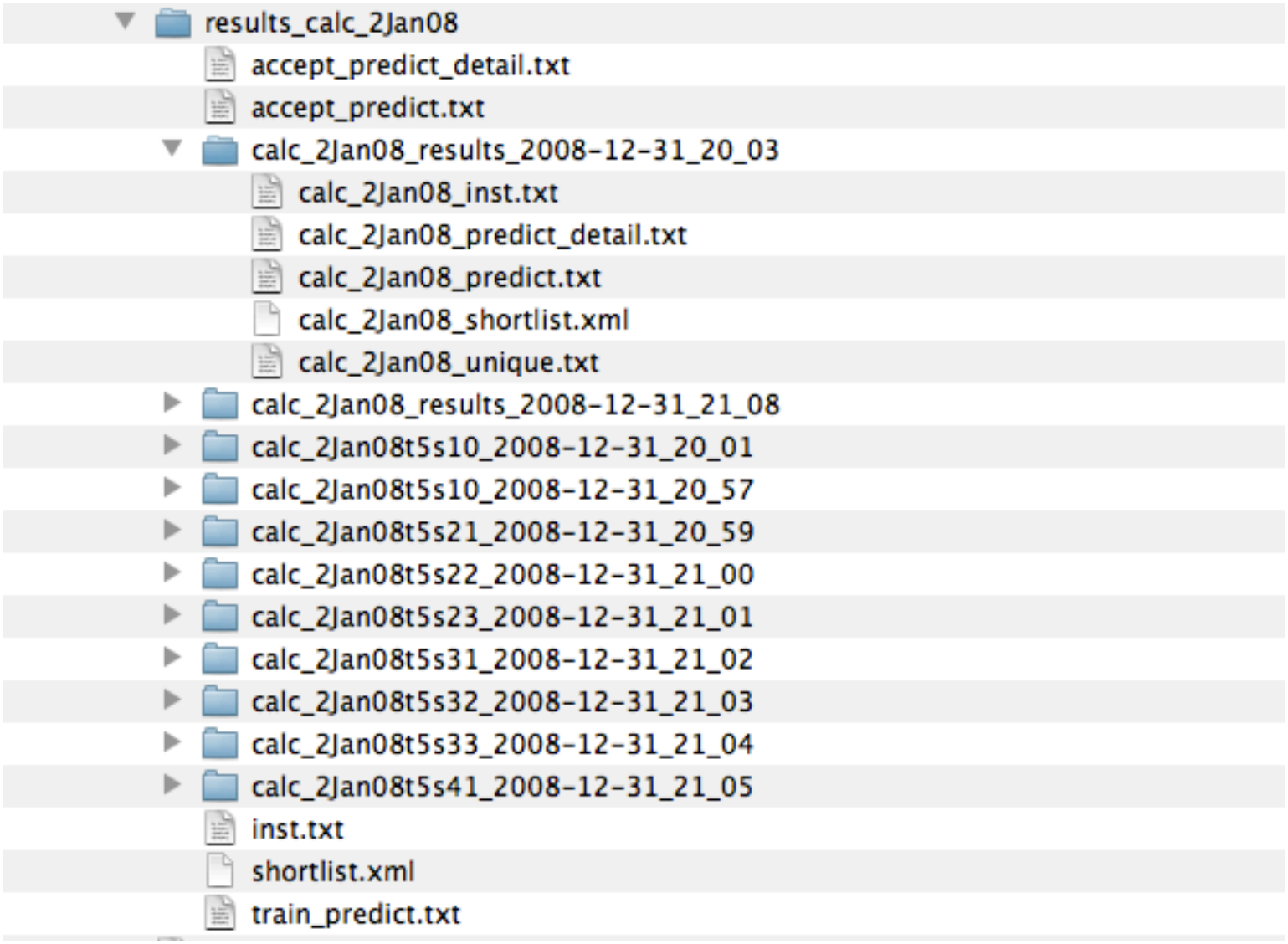
```
<!-- Parameter file version 0.2            -->

<!-- for CdCETransformer and Intelligent   -->

<!-- 19/10/04 Valerie Maxville             -->

<!-- Added MISSING as a parameter 19/10/04 -->
<!-- These files for final experiments 11/7/07 -->

<PARAMETERS>
    <TEMPLATE>CdCETemplate.xml</TEMPLATE>
    <FILE_BASE>fm_projects06</FILE_BASE>
    <IDEAL_SPEC>ideal_email_12Sep07t1s24.xml</IDEAL_SPEC>
    <VERSION>email_12Sep07t1s24</VERSION>
    <TRANSCODE>1</TRANSCODE>
    <ONTOLOGY_LEVEL>2</ONTOLOGY_LEVEL>
    <SKIP_UNSPECIFIED>true</SKIP_UNSPECIFIED>
    <MISSING>-999</MISSING>
</PARAMETERS>
```

Line: 1 Column: 1 XML Tab Size: 4

A single run…


calc_2Jan08t5s10_2008-12-31_20_01
accept_predict_detail.txt
accept_predict.txt
calc_2Jan08t5s10_T5_L2_Strue_fm_projects06.arff
calc_2Jan08t5s10_T5_L2_Strue_test1.arff
calc_2Jan08t5s10_T5_L2_Strue_test2.arff
calc_2Jan08t5s10_T5_L2_Strue_train.arff
generator_errors.txt
generator_output.txt
ideal_calc_2Jan08t5s10.xml
inst.txt
model.mod
parameters_calc_2Jan08t5s10.xml
shortlist.xml
train_error.txt
train_output.txt
train_predict.txt
transformer_errors.txt
transformer_output.txt

A bundle of results…



Snapshots…


calc 2004.zip
calc and emailer 2005.zip
calc_30_first_run_laptop.zip
calc_2008.zip
CdCE ARFF Jan2007.zip
ClassifierSuite_screenshots.zip
email 12Sep07_2.zip
email_28Aug07.zip
email_Sep12_ideal_with_date.zip
emailer_files.zip
FM_projects_CdCE.zip
fm-projects.zip
FMfilter.zip
ideal_calc_files.zip
infiles12Sep07.zip
keywords.zip
old_versions_and_results.zip
ontology_work.zip
pre_2006_saved.zip
results.zip

A completed PhD

academic_regalia_rules(071206).pdf	09/04/2010 7:17 PM	45 KB	Adobe...ument
Award_Abbreviations_Glossary.pdf	09/04/2010 7:23 PM	37 KB	Adobe...ument
dev_new	01/01/2009 2:38 PM	--	Folder
dev_new.zip	04/04/2012 11:41 AM	519.2 MB	ZIP archive
dev_submit	10/01/2012 11:50 AM	--	Folder
FMfilter	10/01/2012 11:58 AM	--	Folder
inputfiles	10/01/2012 11:57 AM	--	Folder
weka-3-0-6.jar	31/12/2008 7:01 PM	2.1 MB	Java JAR file
weka-3-4-14	18/12/2008 10:20 AM	--	Folder
weka-3-4-14.dmg	31/12/2008 6:54 PM	13.7 MB	Disk Image
weka-3-6-0	18/12/2008 7:30 AM	--	Folder
xerces-1_4_4	15/11/2001 1:51 PM	--	Folder
xerces-2_2_1	05/02/2008 1:34 AM	--	Folder
Xerces-J-bin.1.4.4.zip	31/12/2008 6:37 PM	4.3 MB	ZIP archive
xml-writer-0.2	05/02/2008 1:30 AM	--	Folder
dev_submit.zip	10/01/2012 12:00 PM	246.6 MB	ZIP archive
dev.zip	12/04/2012 2:33 PM	2.15 GB	ZIP archive
ECU admin	30/01/2012 12:08 PM	--	Folder
ECU rules	30/01/2012 12:08 PM	--	Folder
experiment	06/01/2007 9:00 AM	--	Folder
Literature	30/01/2012 12:08 PM	--	Folder
my_papers2	01/01/2010 8:51 PM	--	Folder
other stuff	28/03/2012 4:12 PM	--	Folder
PostNominals.pdf	09/04/2010 7:23 PM	246 KB	Adobe...ument
thesis - old versions	01/01/2010 8:54 PM	--	Folder
Thesis Citation Maxville 2012.doc	30/04/2012 2:44 PM	29 KB	Micro...ument
Thesis Citation Maxville 2012.docx	30/04/2012 2:44 PM	127 KB	Micro...ument
THESIS_2010	12/08/2011 5:08 PM	--	Folder
THESIS_2011	04/01/2012 11:08 AM	--	Folder
THESIS_FINAL	06/01/2012 10:55 AM	--	Folder
Valerie Maxville PhD 2012 Disk 1/2.fpbf	12/04/2012 2:57 PM	--	Burn Folder
dev_submit	30/01/2012 12:10 PM	1 MB	Alias
experiment	30/01/2012 12:10 PM	1 MB	Alias
Literature	30/01/2012 12:10 PM	1 MB	Alias
my_papers2	30/01/2012 12:10 PM	1 MB	Alias
THESIS_FINAL	30/01/2012 12:10 PM	1 MB	Alias
Valerie Maxville PhD 2012 Disk 2/2.fpbf	20/04/2012 2:05 PM	--	Burn Folder
dev_new.zip	12/04/2012 2:30 PM	127 KB	Alias
dev.zip	12/04/2012 2:33 PM	127 KB	Alias
writing guides	05/04/2010 5:26 PM	--	Folder

Summary

- We've defined workflows
- We've looked at Unix (bash) commands and scripts to automate workflows
- We've seen how to use Python as a scripting language
- We've modified Python code to be able to use it in automated workflows
- We've got an idea of how to implement a parameter sweep – more in the Pracs

Practical Sessions

- This week we will explore scripting with bash and Python
- We will also explore writing driver and worker code for parameter sweeps

Connecting from home

- There are some Linux servers for Science and Engineering that you can use to work from home:
  - saeshell01p.curtin.edu.au
  - saeshell02p.curtin.edu.au
  - saeshell03p.curtin.edu.au
  - saeshell04p.curtin.edu.au
- We use a secure shell (ssh) for remote login
  - ssh 12345678@saeshell01p.curtin.edu.au

Connecting from home

- On windows, you should use PuTTY or MobaXTerm
  - You can download MobaXterm Home Edition (Installer Edition) from the following link: http://mobaxterm.mobatek.net/download-home-edition.html
  - MobaXTerm can handle graphics/plotting
- On MacOS, use ssh from the command line:
  - ssh –X 12345678@saeshell01p.curtin.edu.au
  - adding –X allows graphics to render for plots
  - Current OS X systems do not have an X window system. Install the XQuartz package to allow for SSH with X11 forwarding on OSX systems: http://www.xquartz.org/

Assessments – Mid-semester test

- The mid-semester test will be held during the lecture period in week 8 (lecture 7):

8am on 16th April

Mid-semester Test

- Time available : 60 minutes
- Questions based on lectures and pracs in weeks 1-7 (Lectures 1-6, Pracs 1-5)

- Last semester's test is on the assessment page
- Revision session will hopefully be offered
  - I will send an announcement with details when known

Next …

- In a few moments...
- Lecture 7

- And next week...
- Mid-semester Test