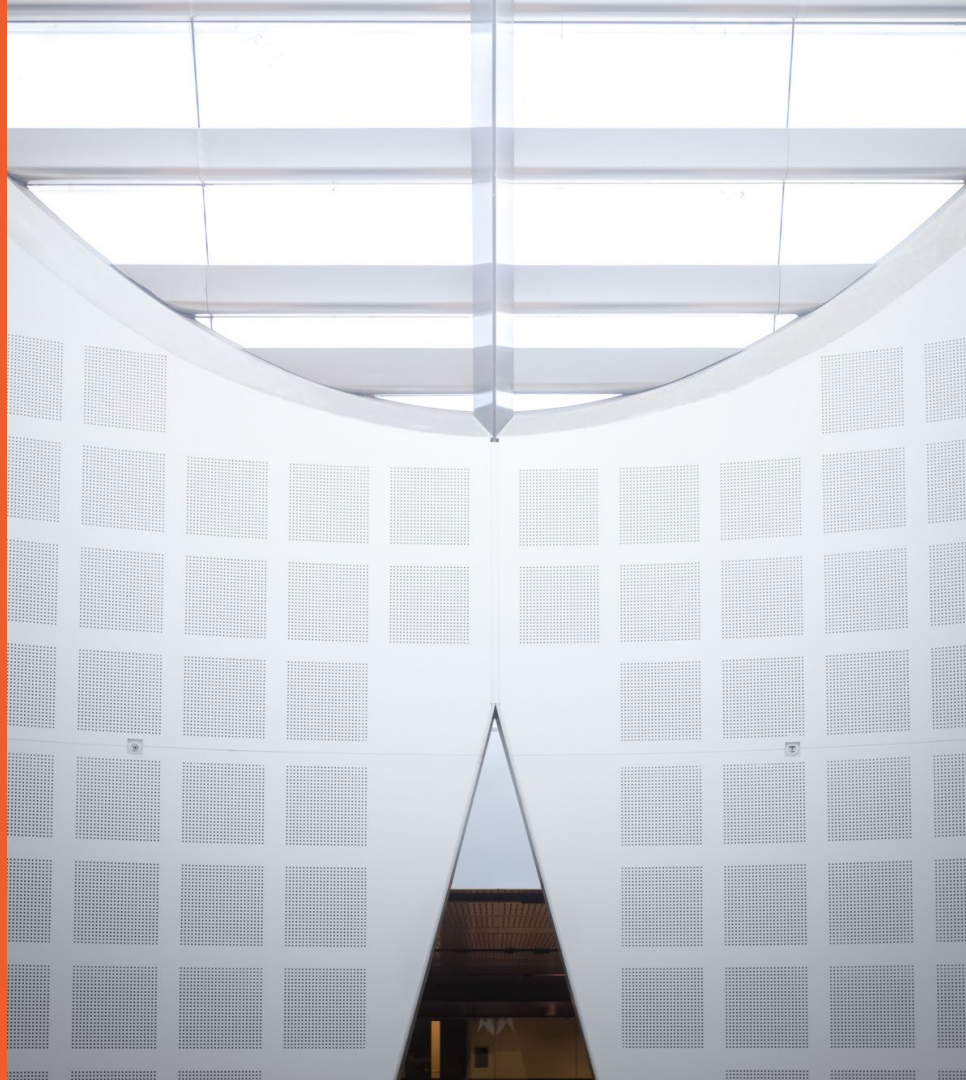
DATA2901: Data Science, Big Data and Data Diversity (adv)

Week 2: Unix Tools with Jupyter

Presented by Dr Ali Anaissi
School of Computer Science

THE UNIVERSITY OF SYDNEY



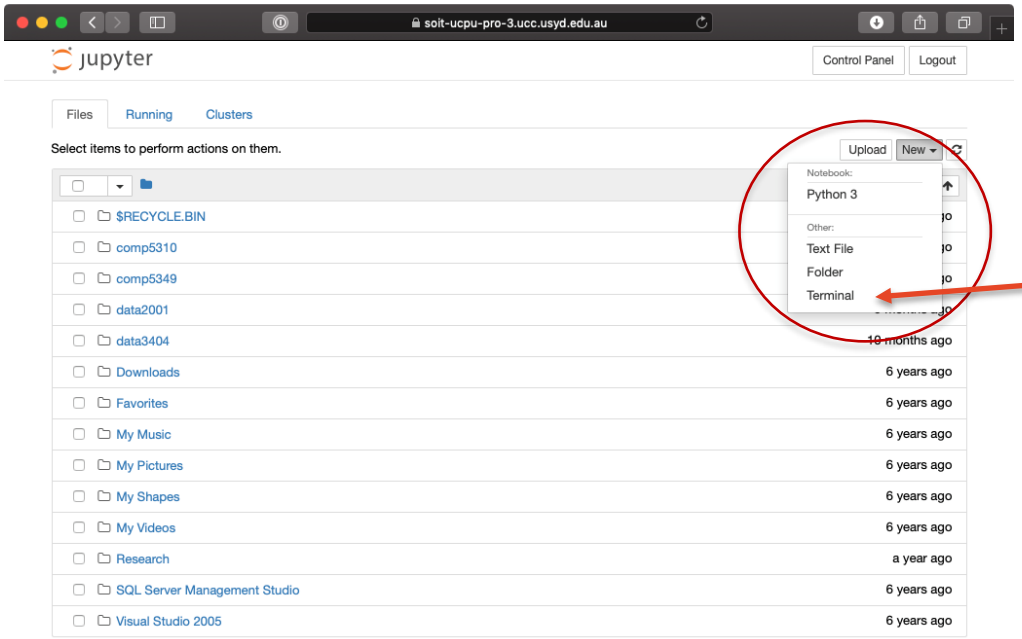
Using Unix Tools with Jupyter

- Different ways to integrate Unix tools with Jupyter Notebooks
  - at least as long as Jupyter server is running on Unix itself

- **Option A:** Start a terminal on the server
  - independent of notebooks themselves, but access to same folders

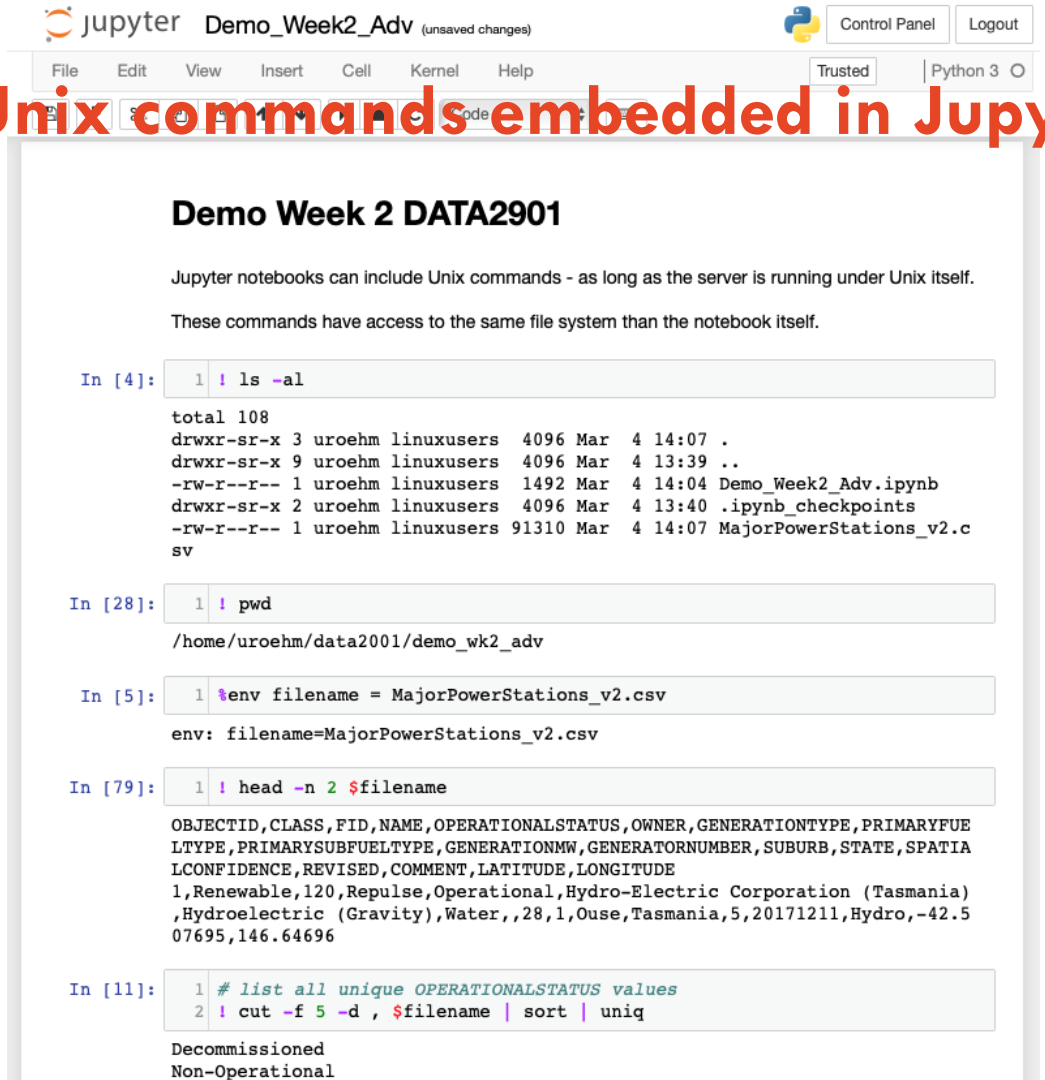
- **Option B:** Include Unix commands in cells
  - simply execute a unix command in a Code cell starting with a **!**
  - to define environment variables, use **%env**

Option 1: Terminal from Jupyter NB



start Unix terminal

Option 2: Unix commands embedded in Jupyter NB



Demo Week 2 DATA2901

Jupyter notebooks can include Unix commands - as long as the server is running under Unix itself.

These commands have access to the same file system than the notebook itself.

In [4]:
```
! ls -al
```

```
total 108
drwxr-sr-x 3 uroehm linuxusers  4096 Mar  4 14:07 .
drwxr-sr-x 9 uroehm linuxusers  4096 Mar  4 13:39 ..
-rw-r--r-- 1 uroehm linuxusers  1492 Mar  4 14:04 Demo_Week2_Adv.ipynb
drwxr-sr-x 2 uroehm linuxusers  4096 Mar  4 13:40 .ipynb_checkpoints
-rw-r--r-- 1 uroehm linuxusers 91310 Mar  4 14:07 MajorPowerStations_v2.c
sv
```

In [28]:
```
! pwd
```

```
/home/uroehm/data2001/demo_wk2_adv
```

In [5]:
```
%env filename = MajorPowerStations_v2.csv
```

```
env: filename=MajorPowerStations_v2.csv
```

In [79]:
```
! head -n 2 $filename
```

```
OBJECTID,CLASS,FID,NAME,OPERATIONALSTATUS,OWNER,GENERATIONTYPE,PRIMARYFUE
LTYPE,PRIMARYSUBFUELTYPE,GENERATIONMW,GENERATORNUMBER,SUBURB,STATE,SPATIA
LCONFIDENCE,REVISED,COMMENT,LATITUDE,LONGITUDE
1,Renewable,120,Repulse,Operational,Hydro-Electric Corporation (Tasmania)
,Hydroelectric (Gravity),Water,,28,1,Ouse,Tasmania,5,20171211,Hydro,-42.5
07695,146.64696
```

In [11]:
```
# list all unique OPERATIONALSTATUS values
! cut -f 5 -d , $filename | sort | uniq
```

```
Decommissioned
Non-Operational
```

Exchanging Data between Unix and Python?

- You can exchange data easily between the Python kernel and the Unix shell
  - Use Python variables in { and } to send data to Unix
  - Use ! on the RHS of a Python variable assignment to receive Unix output
- Example:

```
text = "Shakespeare in Love"
print(text)

modified_text = ! echo "{text}" | sed -e 's/Love/London/g'
print(modified_text)
```

Core Question: Send Code-to-Data or bring Data-to-Code?

- If we load data from external sources into a Jupyter notebook
  - it has to fit main memory (there is no buffering or streaming), and
  - it needs be converted to internal Python representation
- While convenient to program, it takes up space

- If we pre-process / transform / filter data, we can sometimes reduce the amount of data needed to be loaded
  - Unix tools are helping here
  - When done in a notebook can make this explicit and be documented

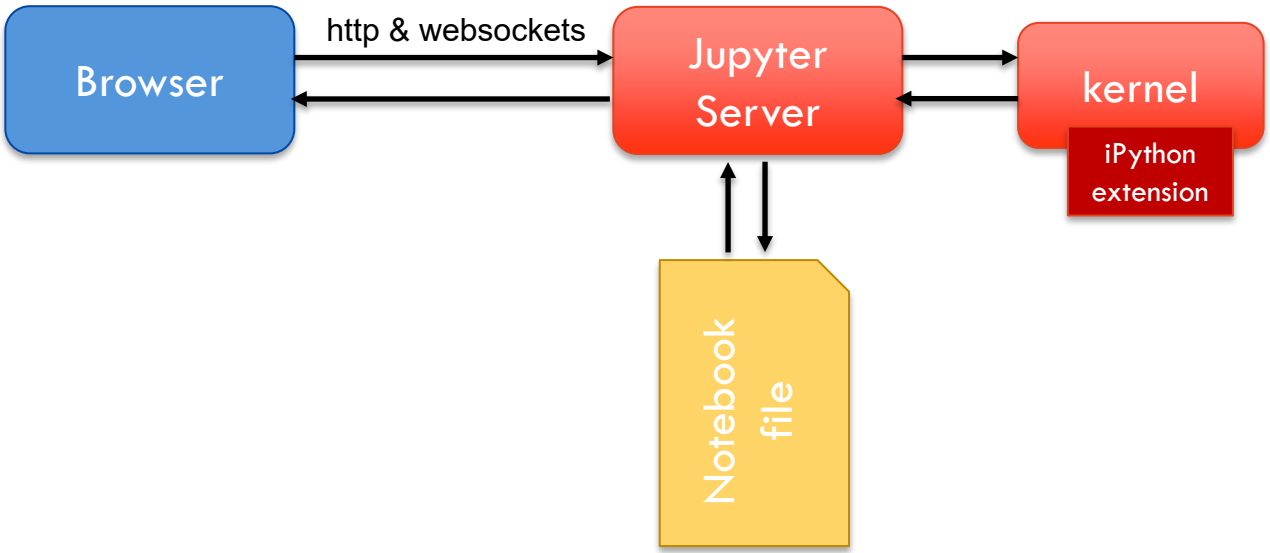
Jupyter Magics and Extensions

Jupyter Ecosystem

Jupyter is very modular and extensible

1. Jupyter Extensions
   - Extends the front-facing user interface, such as **JupyterLab**
2. Jupyter Kernels
   - Jupyter Servers support different **kernels,** not just Python code
   - Default in our installation is the **IPython** kernel, but others can be installed
     - E.g. R, JavaScript, Scala/Spark, Matlab, **bash**
3. IPython Magics
   - custom code extending functionality in iPython cells, indicated by a %
   - Examples: %matplotlib inline %env %timeit
   - For list see **github.com/ipython/ipython/wiki/Extensions-Index**

Jupyter Server Architecture



Jupyter Ecosystem cont'd

- There are more extension possible
  - **IPython widgets**
    - for interactive Jupyter / IPython GUI
  - **Contents Manager**
    - by default, Jupyter loads/saves files from the local file system
    - with different Contents Manager can also support cloud data access (Amazon S3, Google Cloud Storage) or from PostgreSQL or HDFS etc.
  - **NBConvert Exporters**
    - Extensible way to export a Jupyter notebook, e.g. PDF, LaTeX, HTML…
- For more details and ideas, see:
  https://blog.jupyter.org/99-ways-to-extend-the-jupyter-ecosystem-11e5dab7c54

IPython built-in Magic Commands

- IPython magics start with a % (line magics) or a %% (cell magics)
- Examples:
  - %lsmagic list available magic functions
  - %pwd print current working directory
  - %cd change current working directory
  - %env list or set environment variables
  - %conda invoke conda package manager
  - %config configure IPython
  - %matplotlib switch matplotlib interactively or inline
  - %pprint pretty printing on/off
  - %precision floating point precision for pretty printing
  - %time or %timeit time the execution of a command or cell (then with %%)
  - %load_ext load additional magic extension
  - %quickref

IPython built-in Cell Magic Commands

- IPython cell magics start a %%
  - %%bash run cells with bash as subprocess
  - %%html render a cell as block of HTML
  - %%js run the cell block of Javascript code
  - %%latex render the cell as latex
  - %%perl run cell with perl in a sub-process
  - %%ruby run cell with ruby in a sub-process
  - %%script similar to #! Line at start of a Unix shell scripts
  - %%svg render the cell as an SVG literal
  - %%writefile write the content of the cell to a file
- Many of those commands have options; cf. ipython.readthedocs.io

Options to integrate bash with Jupyter

Bash Magic

- IPython bash magic example

Bash Kernel

- https://github.com/takluyver/bash_kernel

sed and AWK

THE UNIVERSITY OF SYDNEY

Processing content of files: sed

- **sed** is a powerful *streaming editor* for Unix
  - typically used to pipe data 'through' sed for some automated changes
  - For example to replace texts

- Example: Text replacement

```
!sed -e 's/from/to/g' $filename
```

  - *s* stands for substitution
  - *from* is the word to be matched
  - *to* is the replacement string
  - *g* specifies to apply this to all occurrences on a line, not just the first.

```
! echo "hello world" | sed -e 's/hello/bonjours/g' >output.txt
```

Processing content of files: awk

- A programming language for the special purpose of text processing and data extraction – and its corresponding tool
  - AWK was created at Bell Labs in the 1970s,[3] and its name is derived from the surnames of its authors—Alfred Aho, Peter Weinberger, and Brian Kernighan
- Very powerful pattern matching language, where code blocks can be executed for each match, and data be extracted into variables or send to output
  - Executes a sequence of ***pattern { action }*** on each line of the input
  - Lines are automatically separated into fields; $0 (line), $1, $2, … , NF
  - Field separators can be specified
  - Special BEGIN and END 'patterns' to execute at start or end of a file

AWK Example: Word Count

```
BEGIN {
      FS="[^a-zA-Z]+"
}
{
       for (i=1; i<=NF; i++)
            words[tolower($i)]++
}
END {
       for (i in words) print i, words[i]
}
```

WaterInfo Example

```
/* check which files are here */
ls
/* dump content of Stations_raw file */
cat Stations_raw.csv
cat Stations_raw.tsv
/* do the same using 'awk' */
awk '{print $0}' Stations_raw.tsv

/* show first two colums of Stations_raw file */
cut -f 1,2 Stations_raw.tsv
awk '{print $1,$2}' Stations_raw.tsv

/* show first two colums of Stations_raw CSV file (comma separator) */
cut -d , -f 1,2 Stations_raw.csv
awk 'BEGIN {FS="," } // {print $1,$2}' Stations_raw.csv
```

WaterInfo Example (cont'd)

```
/* want to convert to 'concatenated' stationIDs */
awk '{print $1$2}' Stations_raw.tsv

/* only for '409xyz' stations */
awk 'BEGIN {FS=","}
     /409/ {print $1 $2}' Stations_raw.csv

/* … and remove trailing 'c' on 409204C */
man awk
awk 'BEGIN {FS=","}
     /409/ {print $1 substr($2,1,3)}' Stations_raw.csv

/* make this SQL INSERT statements - Step 1 */
awk 'BEGIN {FS=","}
  /409/ {print $1 substr($2,1,3) ,$3,$4,$5,$6,$7,$8}' Stations_raw.csv
```

WaterInfo Example (still cont'd)

```
/* make this SQL INSERT statements - Step 2 */
awk 'BEGIN {FS=","}
    {print "INSERT INTO Stations VALUES ("$1 substr($2,1,3) ","$3","$4","$5","$6","$7","$8");"}'
     Stations_raw.csv

/* how to get the right quotes? Best using a script */
converter.awk:
  BEGIN { FS="," }
 /[0-9]+/ { print "INSERT INTO Stations VALUES (" $1 substr($2,1,3) "','" $3 "','" $4 "','" $5
"','" $6 "','" $7 "','" $8 "');"
  }
awk -f converter.awk Stations_raw.csv

/* how to include NULL values for unknown 'cease' dates? */
converter2.awk:
  BEGIN { FS="," }
 /[0-9]+/ {  if (length($7) > 0) { cease="'"$7"'" }  else { cease = "NULL" };
     print "INSERT INTO Stations VALUES (" $1 substr($2,1,3) "','" $3 "','" $4 "','" $5 "','" $6
"'," cease ",'" $8 "');"
  }
```

WaterInfo Example (still cont'd)

```
/* Let's format it a bit nicer and include NULL values for all strings? */
converter3.awk:
  BEGIN { FS="," }
  /[0-9]+/ {
     stationid = $1 substr($2,1,3)

     printf "INSERT INTO Stations VALUES (%i", stationid

     for (i = 3; i < NF; i++ ) {
          if (length($i) > 0) {
               printf ", '%s'", $i
          } else {
               printf ", NULL"
          };
     }

     print ");"
  }

awk -f converter3.awk Stations_raw.csv
```