(Re) introduction to Unix

Sarah Medland

2025

Getting the most out of the workshop

1

Ask questions!!!

2

Don’t sit next to someone you already know

3

Work with someone with a different skillset and different experience level

4

You will have access to your files after you leave

5

Come to the social functions

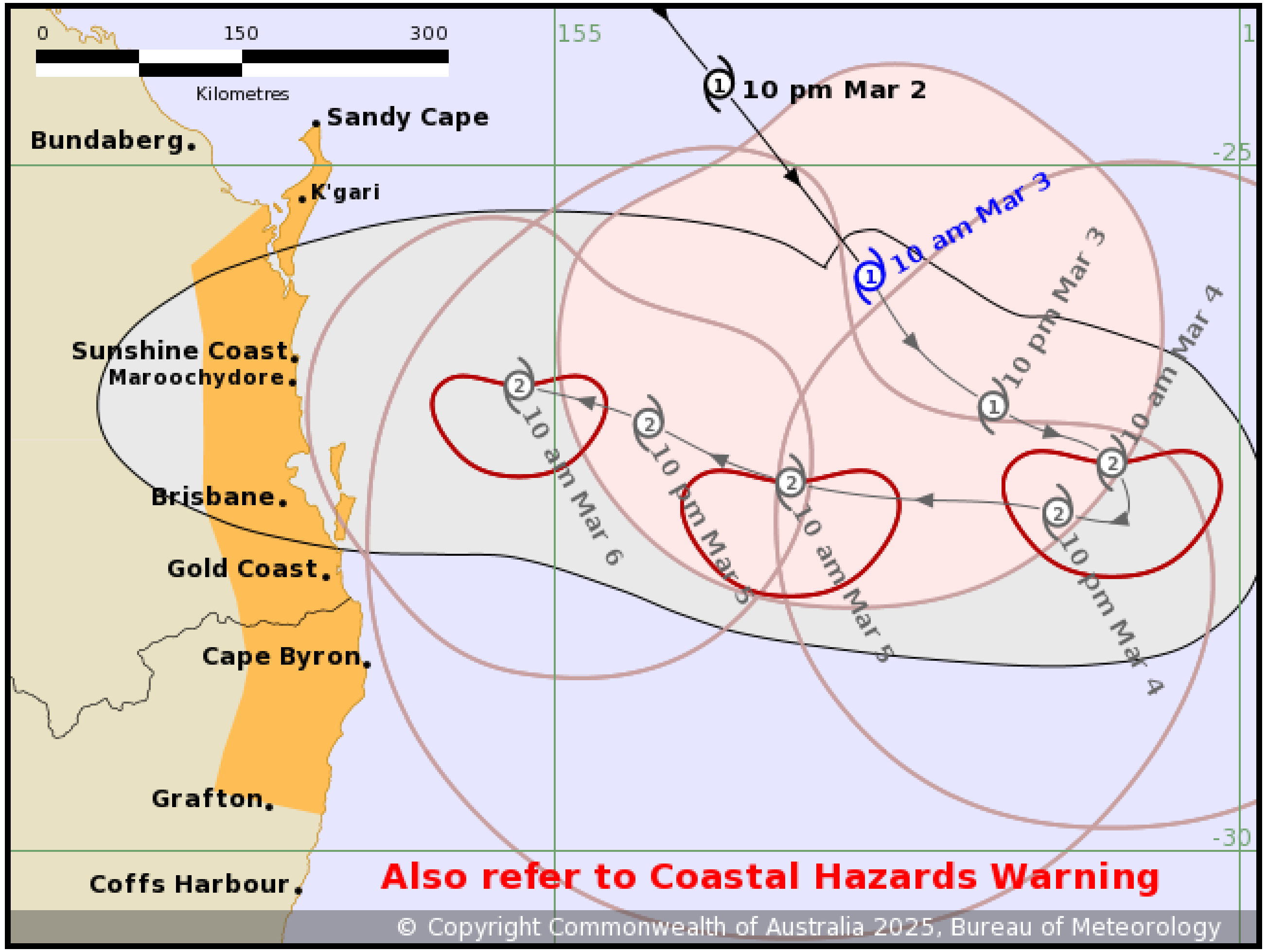
6

Ask questions!!!

Japan
South Korea
North Pacific Ocean
Indonesia
Papua New Guinea
B
Australia
New Zealand

I work in Brisbane at QIMR

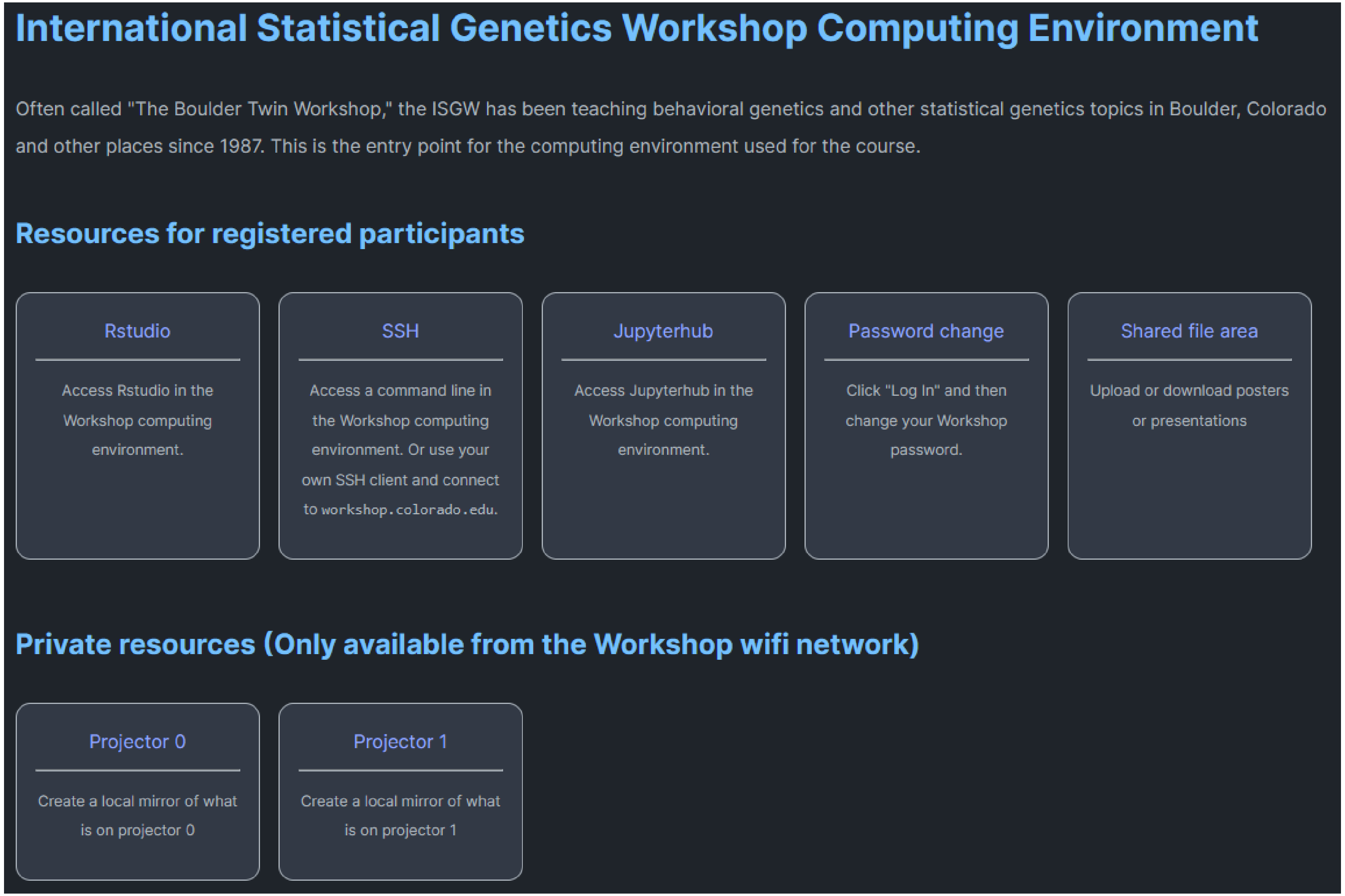
not



Also refer to Coastal Hazards Warning

© Copyright Commonwealth of Australia 2025, Bureau of Meteorology

https://workshop.colorado.edu/



International Statistical Genetics Workshop Computing Environment

Often called "The Boulder Twin Workshop," the ISGW has been teaching behavioral genetics and other statistical genetics topics in Boulder, Colorado and other places since 1987. This is the entry point for the computing environment used for the course.

Resources for registered participants

Rstudio

Access Rstudio in the Workshop computing environment.

SSH

Access a command line in the Workshop computing environment. Or use your own SSH client and connect to `workshop.colorado.edu`.

Jupyterhub

Access Jupyterhub in the Workshop computing environment.

Password change

Click "Log In" and then change your Workshop password.

Shared file area

Upload or download posters or presentations

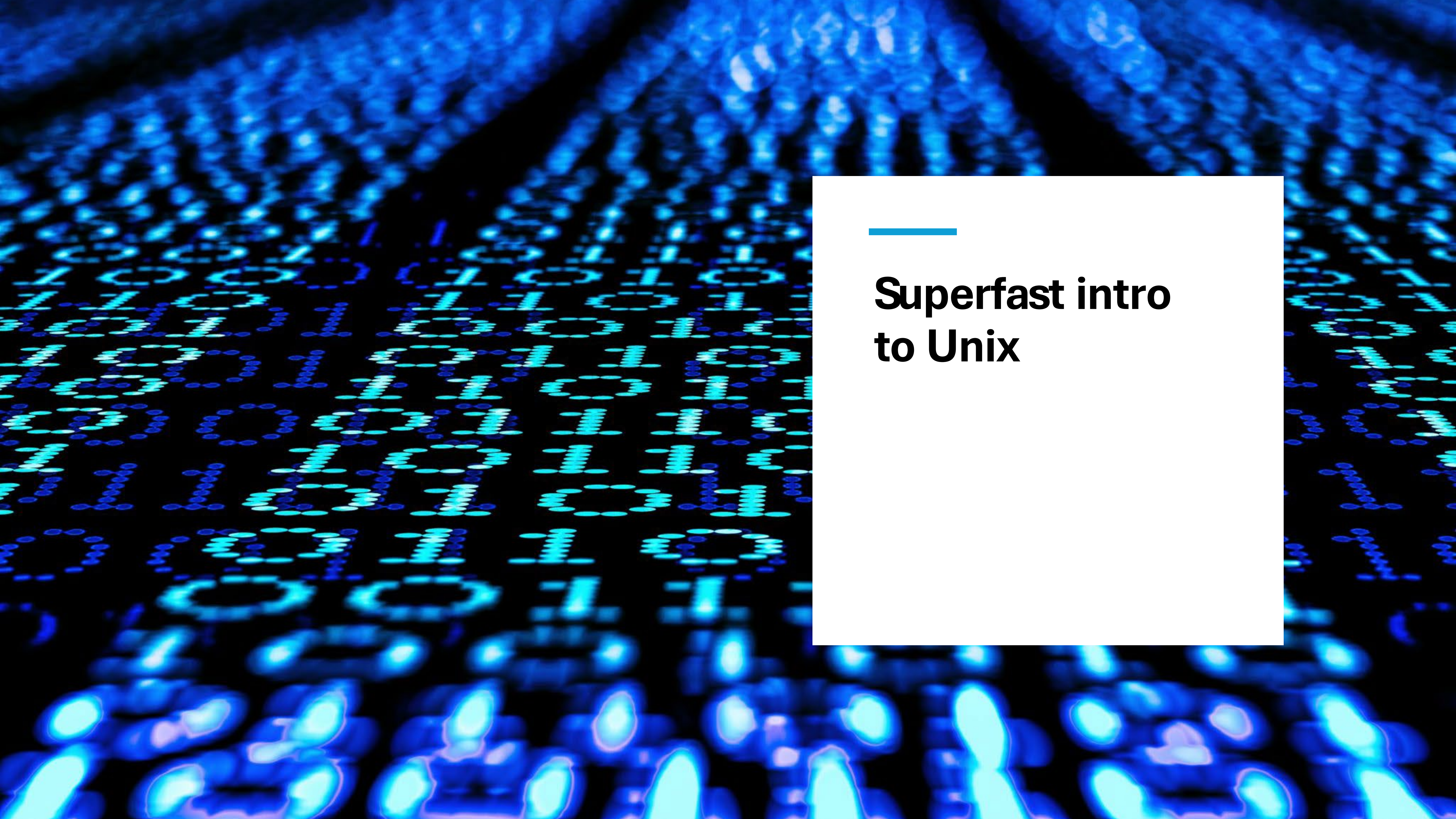
Private resources (Only available from the Workshop wifi network)

Projector 0

Create a local mirror of what is on projector 0

Projector 1

Create a local mirror of what is on projector 1



Superfast intro to Unix

File hygiene is very important

- Files are stored in Unix format not DOS or Mac
  - Changes the line ending characters
  - Use dos2unix, unix2dos, mac2unix, unix2mac to change formats
  - Can use the file command to check format
- Unix systems are case sensitive!
- NO SPACES in your file/directory names!!
- Wildcards ie dos2unix *.dat

Working in the terminal Input …. Output

- Input
  - Most commands don’t need input signifiers
  - < can be used to specify
- Output
  - Without specifying most output will print to the screen
  - > can be used to direct
    - type: `echo ‘this is a dummy file’`
    - `echo ‘this is a dummy file’ > dummy.txt`

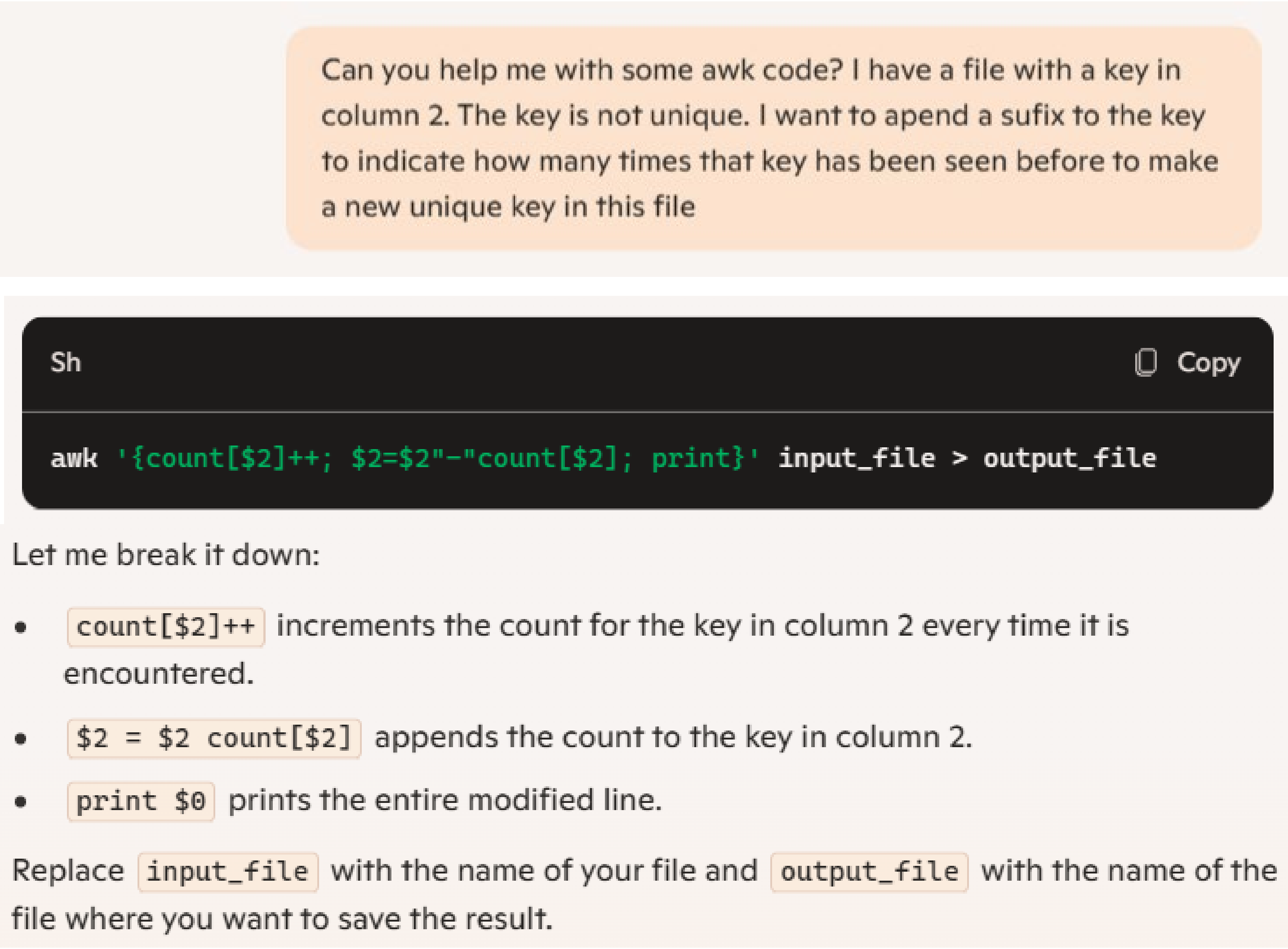
      | (pipe) | more pauses the output after a screen worth of text has appeared hit the space bar to get the next screens worth

The manual

- The man command can be used in conjunction with other commands to put up some basic instructions

- type: man ls
  - ls is the list command it pulls up a list of the files in the directory

Many helpful webpages w examples & AI
(But don’t include file paths or data in your AI request)



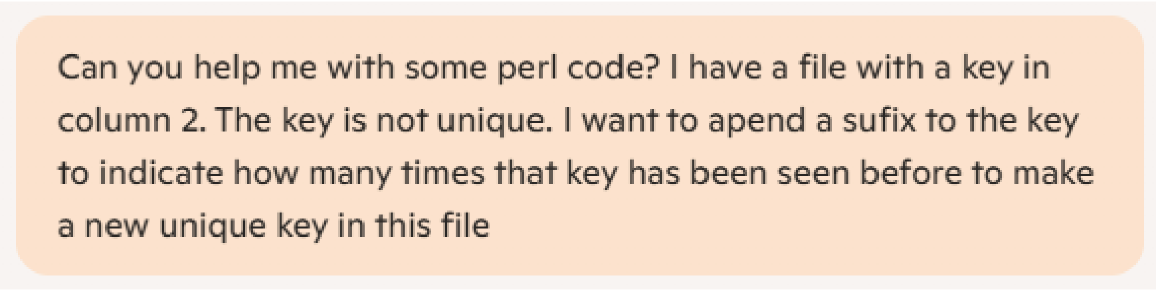
Can you help me with some awk code? I have a file with a key in column 2. The key is not unique. I want to apend a sufix to the key to indicate how many times that key has been seen before to make a new unique key in this file

```sh
awk '{count[$2]++; $2=$2"-"count[$2]; print}' input_file > output_file
```

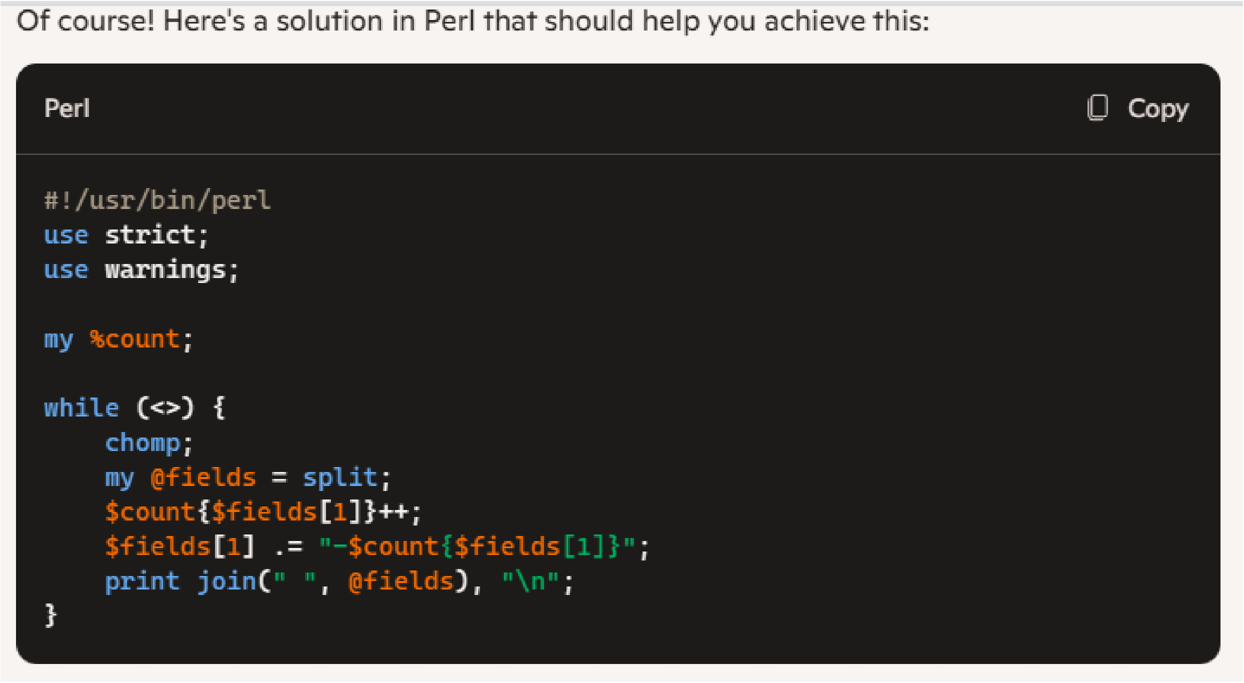
Let me break it down:

- `count[$2]++` increments the count for the key in column 2 every time it is encountered.
- `$2 = $2 count[$2]` appends the count to the key in column 2.
- `print $0` prints the entire modified line.

Replace `input_file` with the name of your file and `output_file` with the name of the file where you want to save the result.



Can you help me with some perl code? I have a file with a key in column 2. The key is not unique. I want to apend a sufix to the key to indicate how many times that key has been seen before to make a new unique key in this file

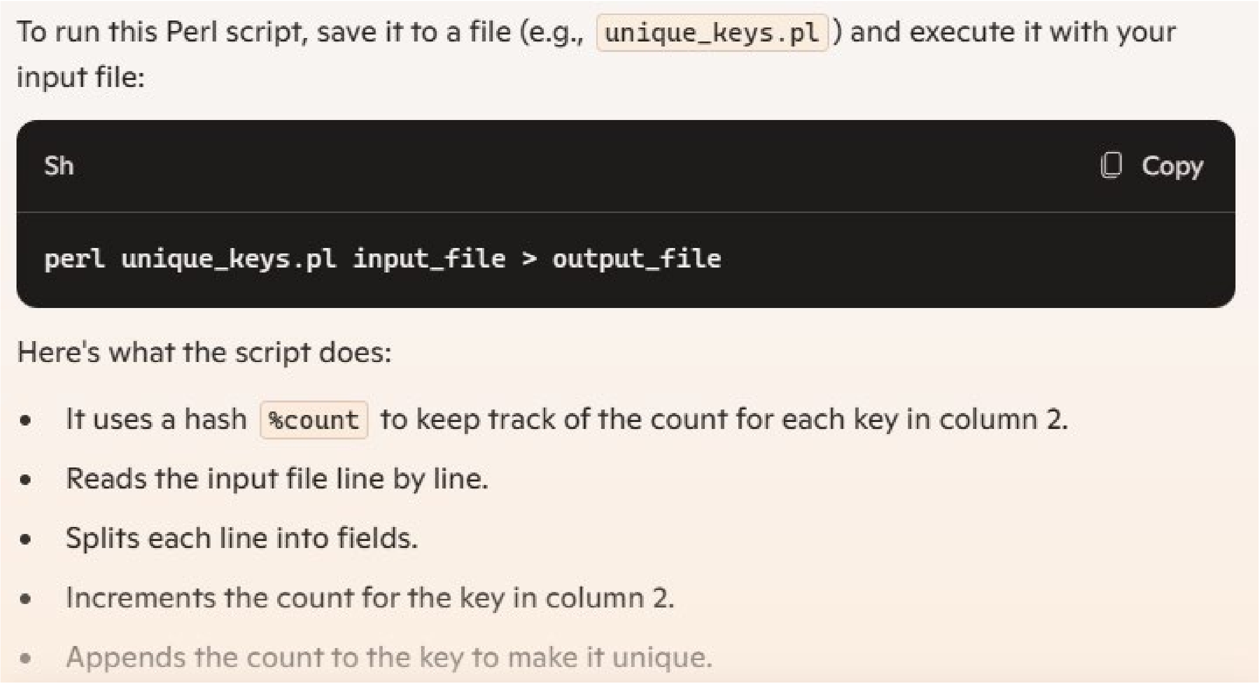


Of course! Here's a solution in Perl that should help you achieve this:

```perl
#!/usr/bin/perl
use strict;
use warnings;

my %count;

while (<>) {
    chomp;
    my @fields = split;
    $count{$fields[1]}++;
    $fields[1] .= "-$count{$fields[1]}";
    print join(" ", @fields), "\n";
}
```

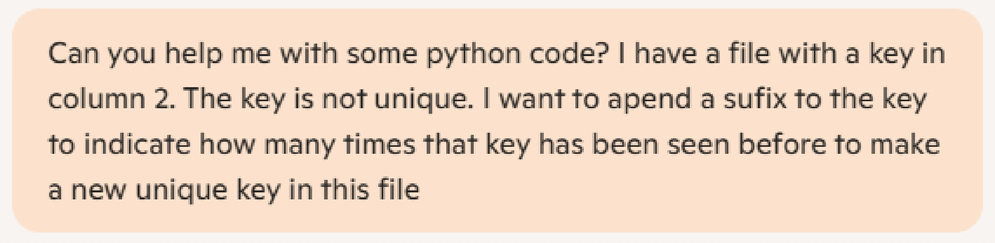


To run this Perl script, save it to a file (e.g., `unique_keys.pl`) and execute it with your input file:

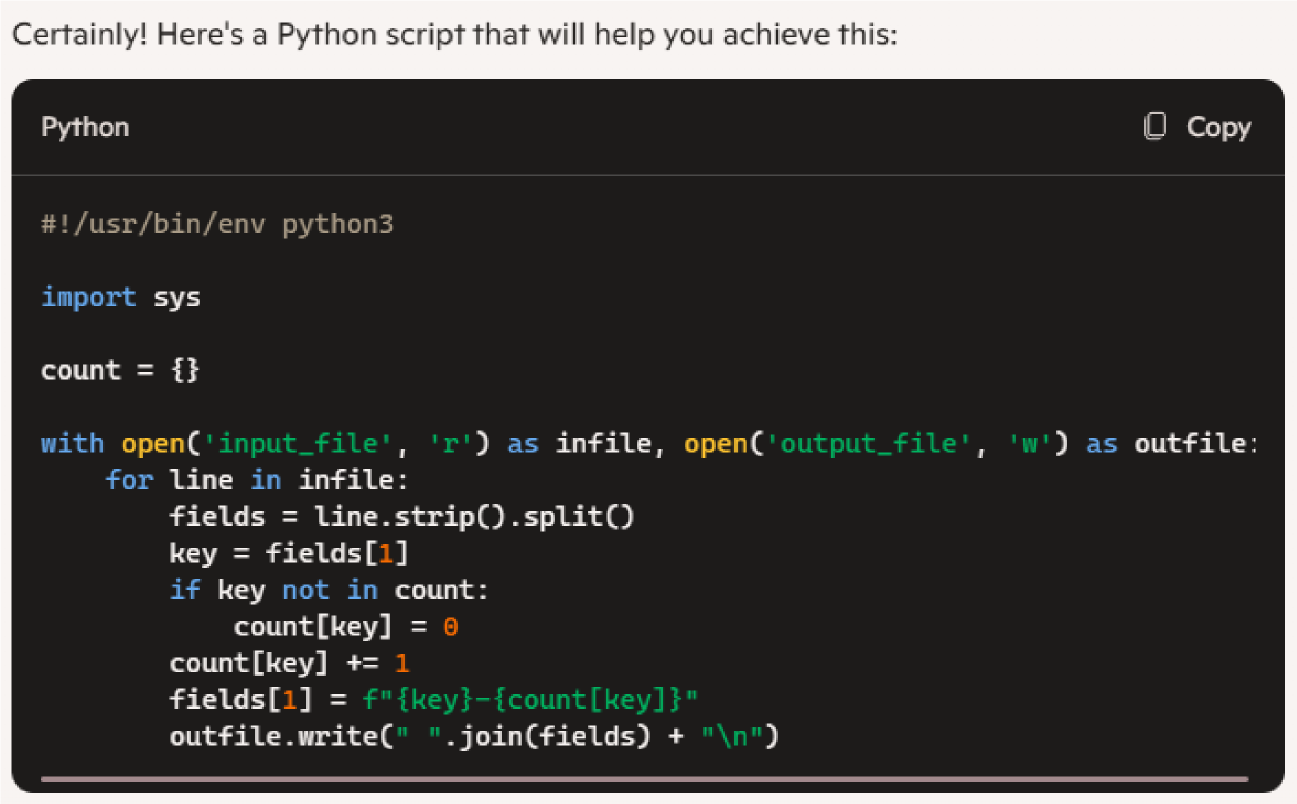
```sh
perl unique_keys.pl input_file > output_file
```

Here's what the script does:

- It uses a hash `%count` to keep track of the count for each key in column 2.
- Reads the input file line by line.
- Splits each line into fields.
- Increments the count for the key in column 2.
- Appends the count to the key to make it unique.



Can you help me with some python code? I have a file with a key in column 2. The key is not unique. I want to apend a sufix to the key to indicate how many times that key has been seen before to make a new unique key in this file



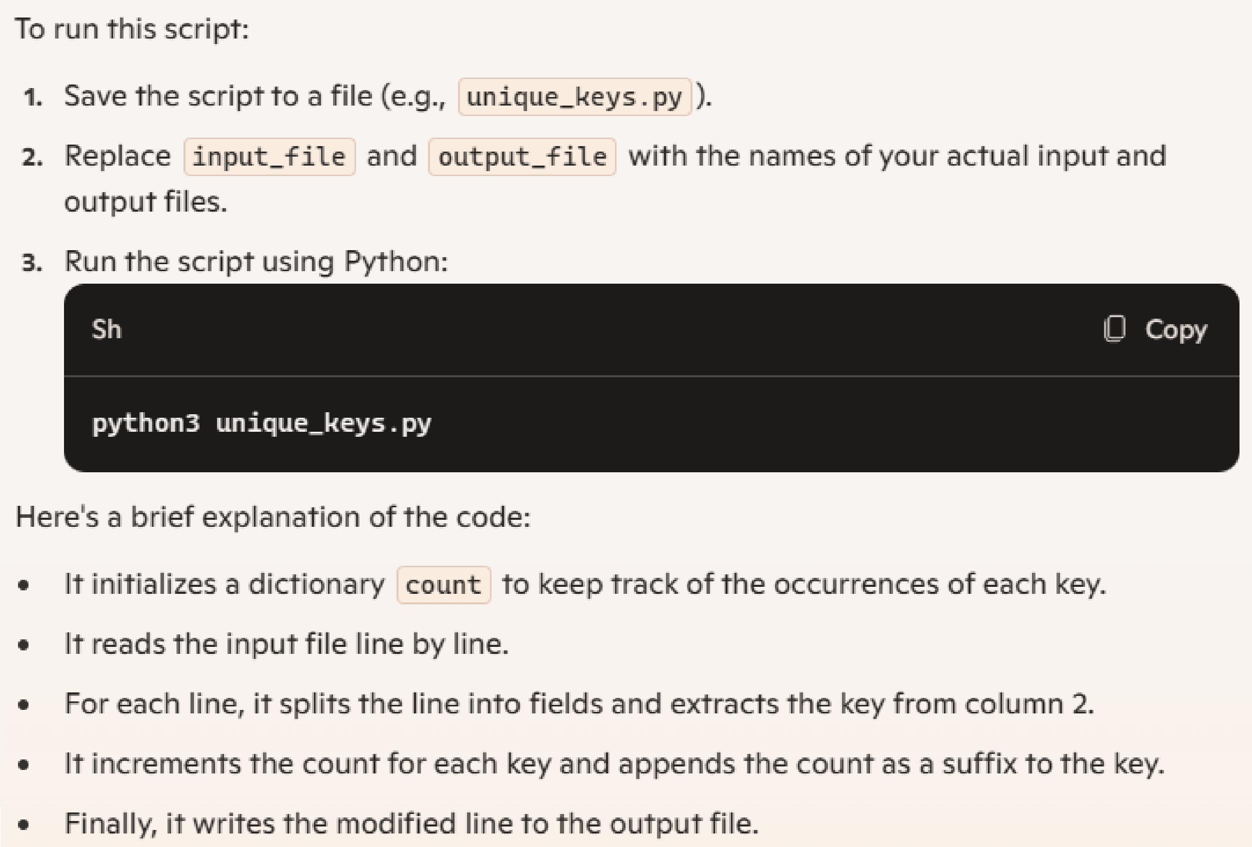
Certainly! Here's a Python script that will help you achieve this:

```python
#!/usr/bin/env python3

import sys

count = {}

with open('input_file', 'r') as infile, open('output_file', 'w') as outfile:
    for line in infile:
        fields = line.strip().split()
        key = fields[1]
        if key not in count:
            count[key] = 0
        count[key] += 1
        fields[1] = f"{key}-{count[key]}"
        outfile.write(" ".join(fields) + "\n")
```



To run this script:

1. Save the script to a file (e.g., `unique_keys.py`).
2. Replace `input_file` and `output_file` with the names of your actual input and output files.
3. Run the script using Python:

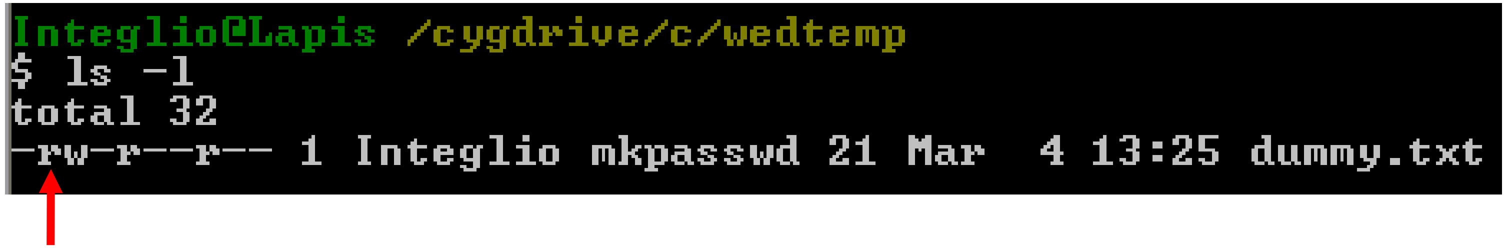
```sh
python3 unique_keys.py
```

Here's a brief explanation of the code:

- It initializes a dictionary `count` to keep track of the occurrences of each key.
- It reads the input file line by line.
- For each line, it splits the line into fields and extracts the key from column 2.
- It increments the count for each key and appends the count as a suffix to the key.
- Finally, it writes the modified line to the output file.

Permissions the ability to **read**, **write** and **execute** files

- type: ls –l



```
Integlio@Lapis /cygdrive/c/wedtemp
$ ls -l
total 32
-rw-r--r-- 1 Integlio mkpasswd 21 Mar  4 13:25 dummy.txt
```

- These are the permissions
- 1st a directory flag (d or -)
- then 3 letters to define the owners permissions
- 3 letters to define the groups permissions
- 3 letters to define the everyone else's permissions

Permissions the ability to **r**ead, **w**rite and e**x**ecute files

- **r**ead access
- **w**rite access
- e**x**ecute
  - ▫ to ‘run’ script or a program the file must be made executable

Permissions the ability to **r**ead, **w**rite and e**x**ecute files

- To change the mode/permissions use chmod
  - a number of ways to do this
  - type: echo “this is a test” > dummy.txt
  - ls –l
  - chmod +x dummy.txt
  - ls –l
  - chmod -x dummy.txt
  - ls –l
  - what happened?

Useful ‘one liners’

- cp copy
- mv move = rename
- rm remove
- ls list
- echo
- head looks at the top 10 lines
- tail looks at the last 10 lines
- wc counts number of lines, words, characters

Useful ‘one liners’

- pwd find where you are
- ~/ get to your home directory
- file reports type of file
- cd without a destination will take you home
- sed find and replace
- grep find and report
- awk restructure files
- uniq find unique lines

Grep

- search **g**lobally for lines matching the **r**egular **e**xpression, and **p**rint them
  - For association output for chromosome 2
  - To extract the result for snp rs59831
  - Type: grep ‘rs59831’ output.txt > summary.txt

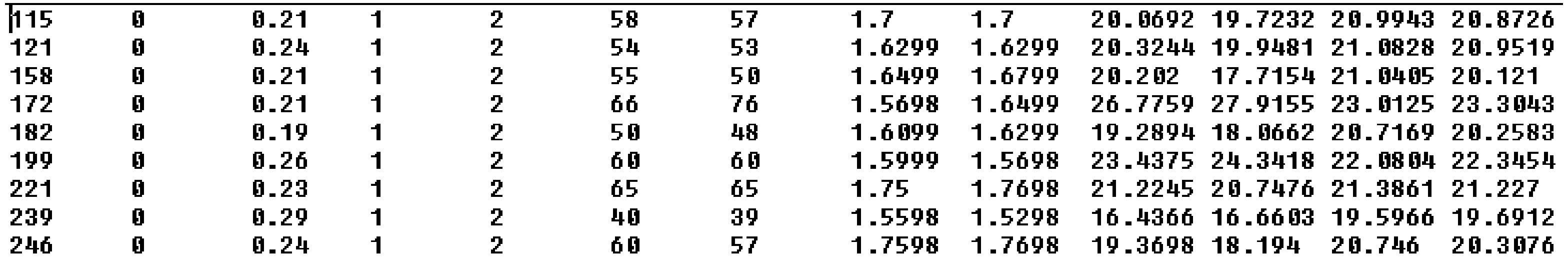
Grep

- Useful flags
  - -v
    - reverse grep select line that does not have the pattern
  - -C x
    - To x rows before and after the target
  - -n
    - Print the line number before the line
  - Many more…

Awk

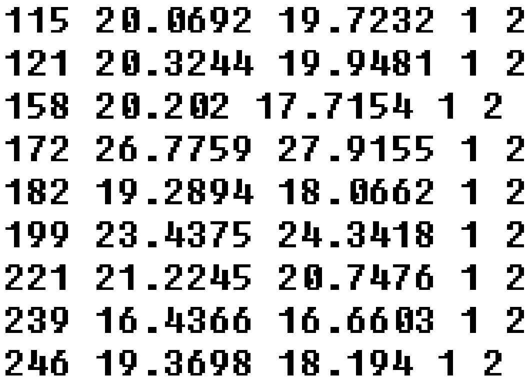
- derived from the surnames of its authors — Alfred **A**ho, Peter **W**einberger, and Brian **K**ernighan
- Many functions
- Very useful for restructuring data

- Ozbmi2.rec



```
115     0       0.21    1       2       58      57      1.7     1.7     20.0692 19.7232 20.9943 20.8726
121     0       0.24    1       2       54      53      1.6299  1.6299  20.3244 19.9481 21.0828 20.9519
158     0       0.21    1       2       55      50      1.6499  1.6799  20.202  17.7154 21.0405 20.121
172     0       0.21    1       2       66      76      1.5698  1.6499  26.7759 27.9155 23.0125 23.3043
182     0       0.19    1       2       50      48      1.6099  1.6299  19.2894 18.0662 20.7169 20.2583
199     0       0.26    1       2       60      60      1.5999  1.5698  23.4375 24.3418 22.0804 22.3454
221     0       0.23    1       2       65      65      1.75    1.7698  21.2245 20.7476 21.3861 21.227
239     0       0.29    1       2       40      39      1.5598  1.5298  16.4366 16.6603 19.5966 19.6912
246     0       0.24    1       2       60      57      1.7598  1.7698  19.3698 18.194  20.746  20.3076
```

- awk ‘{ print $1, $10, $11, $4, $5 }’ ozbmi2.rec > new.rec



```
115 20.0692 19.7232 1 2
121 20.3244 19.9481 1 2
158 20.202 17.7154 1 2
172 26.7759 27.9155 1 2
182 19.2894 18.0662 1 2
199 23.4375 24.3418 1 2
221 21.2245 20.7476 1 2
239 16.4366 16.6603 1 2
246 19.3698 18.194 1 2
```

Awk

- $1 = column 1
- Print $0 = print whole line
- add subtract multiply etc
- change number of decimals
- Many functions

Sort

- Useful flags
  - -f ignore case
  - -n numeric sort
  - -r reverse
  - -c check if a file is sorted
  - -u prints only unique lines
  - -k2 sort starting at column 2

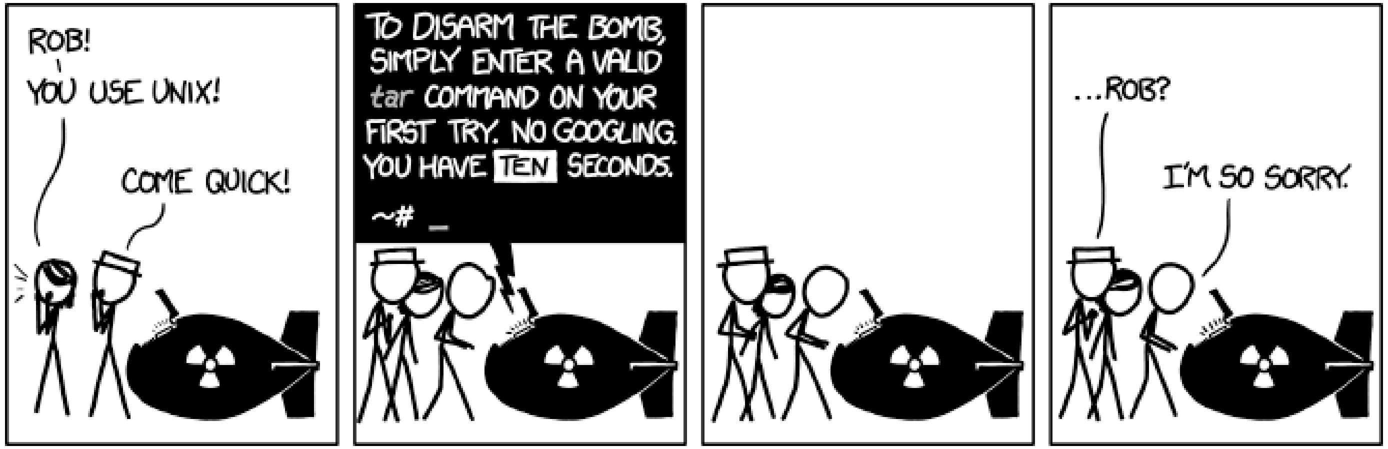
  - sort –fg –k 3 (sort in numeric order on column 3)

Zipping and unzipping

- zip
  - zip my1st.zip *txt
  - zip –mTr my1st.zip *txt
- unzip
  - unzip my1st.zip
- gzip
  - gzip example.txt
- Un-gzip
  - gzip –d example.txt.gz

tar

- Unzipping tar.gz files
  - tar -xzvf example.tar.gz
- Make Tar files
  - tar cvf MyProject.tar MyProject
- List contents
  - tar tvf my-archive.tar
  - tar tzvf my-archive.tar.gz


ROB!
YOU USE UNIX!
COME QUICK!
TO DISARM THE BOMB, SIMPLY ENTER A VALID tar COMMAND ON YOUR FIRST TRY. NO GOOGLING. YOU HAVE TEN SECONDS.
~# _
...ROB?
I'M SO SORRY.

Looking at your data

- less *filename*
  - Allows you to scroll through your data
- less –S *filename*
  - Shows a screen width of data (stops text wrapping)
- zless –S *filename*
  - Allows you to look at a gz file without unzipping

Nano/pico (text editor)

- nano *filename*
  - Commands at bottom of screen
  - Save = crtl+O
  - Exit = crtl +X

Putting it together

- Making a ‘shell’ script to automate analyses

- <contents of imaginary file inefficient.sh>
- pedstats –p 1.ped –d 1.dat –pdf --prefix:1
- merlin –p 1.ped –d 1.dat –m 1.map --vc --pdf --prefix:1
- pedstats –p 2.ped –d 2.dat –pdf --prefix:2
- merlin –p 2.ped –d 2.dat –m 2.map --vc --pdf --prefix:2
- pedstats –p 3.ped –d 3.dat –pdf --prefix:3
- merlin –p 3.ped –d 3.dat –m 3.map --vc --pdf --prefix:3

- To run this make inefficient.sh executable then type ./inefficient.sh

Loops 1

- <contents of imaginary file loop_a.sh>
- for $i in 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
- do
- pedstats –p $i.ped –d $i.dat --pdf --prefix:$i
- merlin –p $i.ped –d $i.dat –m $i.map --vc --pdf --prefix:$i
- done

Loops 2

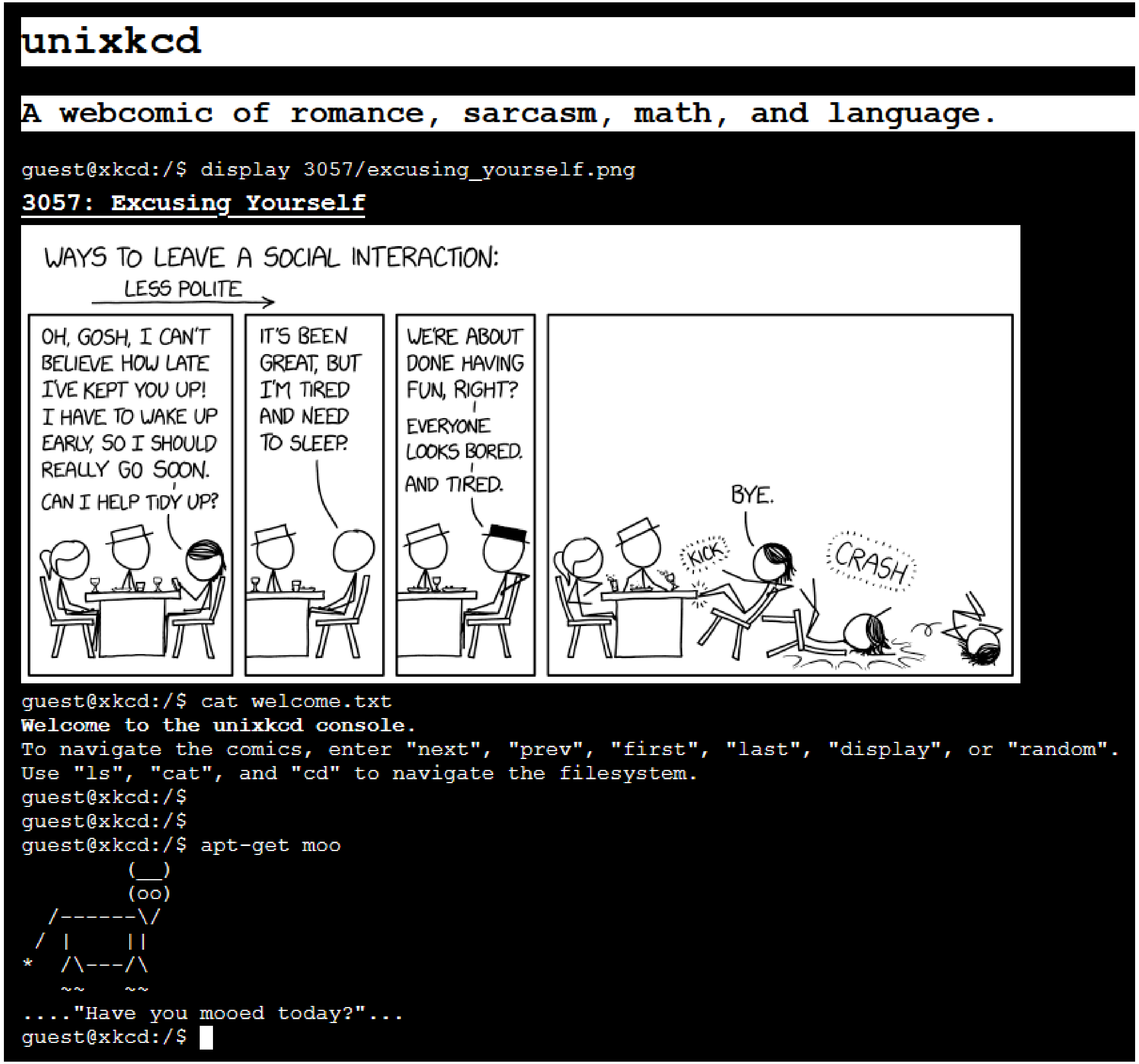
- `<contents of imaginary file loop_b.sh>`
- `for ((  i = 1 ;  i <= 22 ;  i++  ))`
- `do`
- `        pedstats –p $i.ped –d $i.dat --pdf --prefix:$i`
- `        merlin –p $i.ped –d $i.dat –m $i.map --vc --pdf --prefix:$i`
- `done`

Other bits

- When working on servers
  - Queuing systems
  - crtl+c
  - crtl+z
  - which

Shutting down your unix session

- exit
- logout
- quit
- q


WAYS TO LEAVE A SOCIAL INTERACTION:
LESS POLITE
OH, GOSH, I CAN'T BELIEVE HOW LATE I'VE KEPT YOU UP! I HAVE TO WAKE UP EARLY, SO I SHOULD REALLY GO SOON.
CAN I HELP TIDY UP?
IT'S BEEN GREAT, BUT I'M TIRED AND NEED TO SLEEP.
WE'RE ABOUT DONE HAVING FUN, RIGHT?
EVERYONE LOOKS BORED.
AND TIRED.
BYE.
KICK
CRASH

unixkcd

A webcomic of romance, sarcasm, math, and language.

```
guest@xkcd:/$ display 3057/excusing_yourself.png
```

3057: Excusing Yourself

```
guest@xkcd:/$ cat welcome.txt
Welcome to the unixkcd console.
To navigate the comics, enter "next", "prev", "first", "last", "display", or "random".
Use "ls", "cat", and "cd" to navigate the filesystem.
guest@xkcd:/$
guest@xkcd:/$
guest@xkcd:/$ apt-get moo
        (__)
        (oo)
  /------\/
 / |    ||
*  /\---/\
   ~~   ~~
...."Have you mooed today?"...
guest@xkcd:/$
```