Introduction to Unix

Slides at
http://stab.st-andrews.ac.uk/wiki/index.php/Intro_to_Unix_2017

Please fill out the pre-event section on the feedback forms.

Intro/My style

- Play about!
- Ask questions!
- Introvert/Anxious/English not your first language – I hope you feel comfortable enough to ask, but if not, ask Ramon or Chris, and he'll help or ask for you.

- This course will only help if you practice afterwards.

Type carefully

- Green text is what to type
- Copy-paste is fine, but you'll learn better by typing
  - Don't blindly copy-and-paste what's written. Think about *why* each word is there. Ask questions.
  - Copy-pasting errors into google is encouraged.
- Type carefully.
  - Case matters
  - Being completely accurate matters
  - Read the errors!

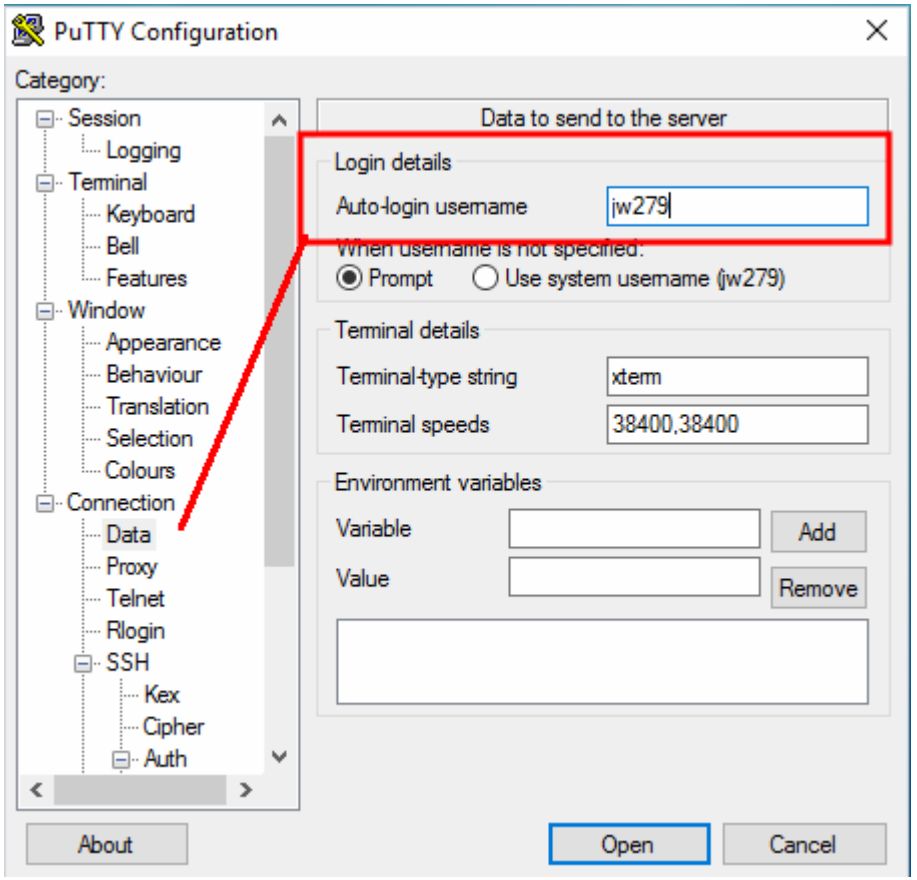
Unix Background

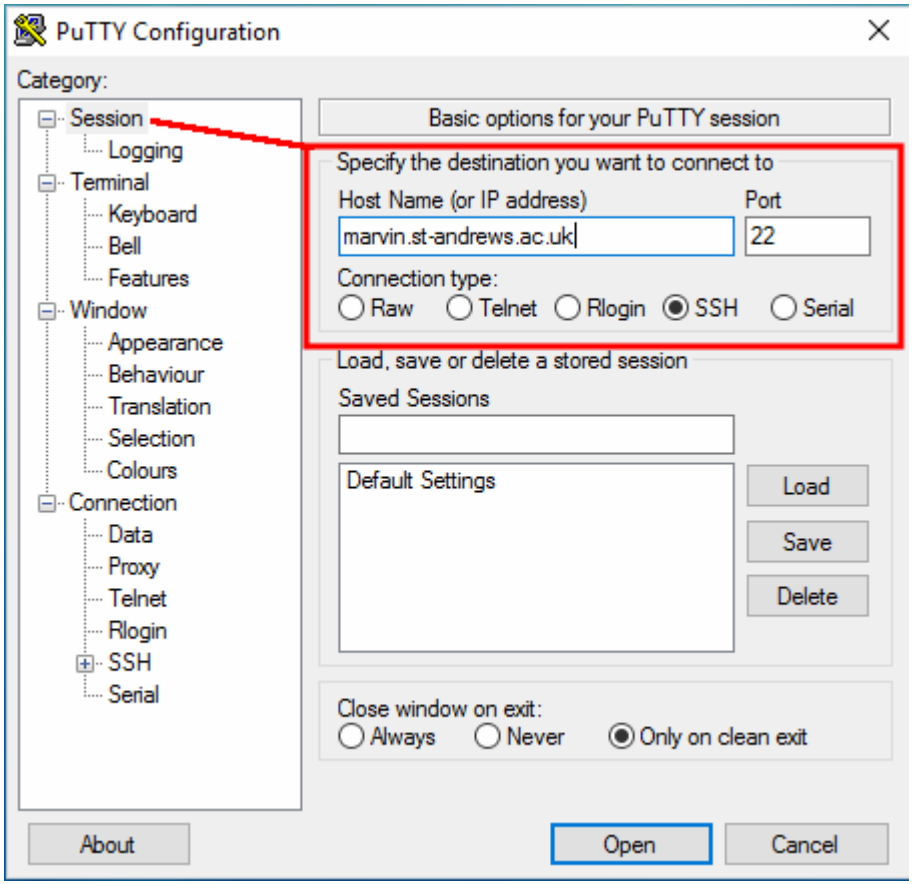
- Origins back in the ‘70s
- Many different backend bits
- What you see should, mostly, be the same
- Most of the internet.
- Influenced most commant lines (inc. R and Matlab)

- Pretty darn useful.

Logging in.

- Session → Hostname : marvin.st-andrews.ac.uk

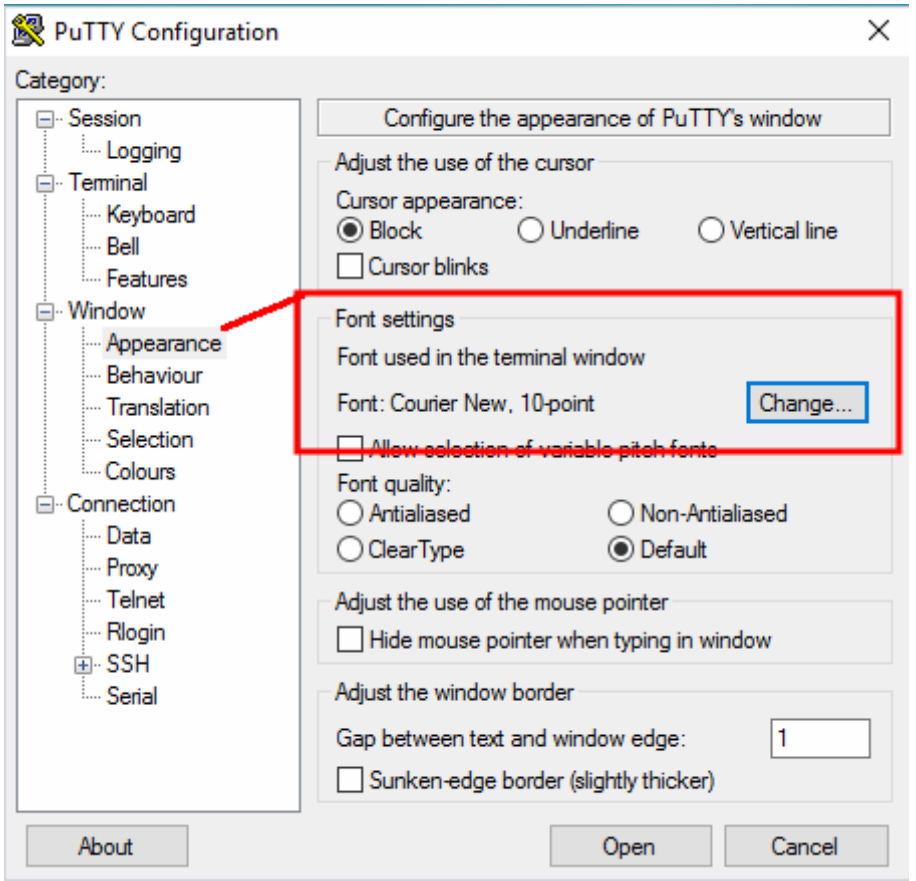


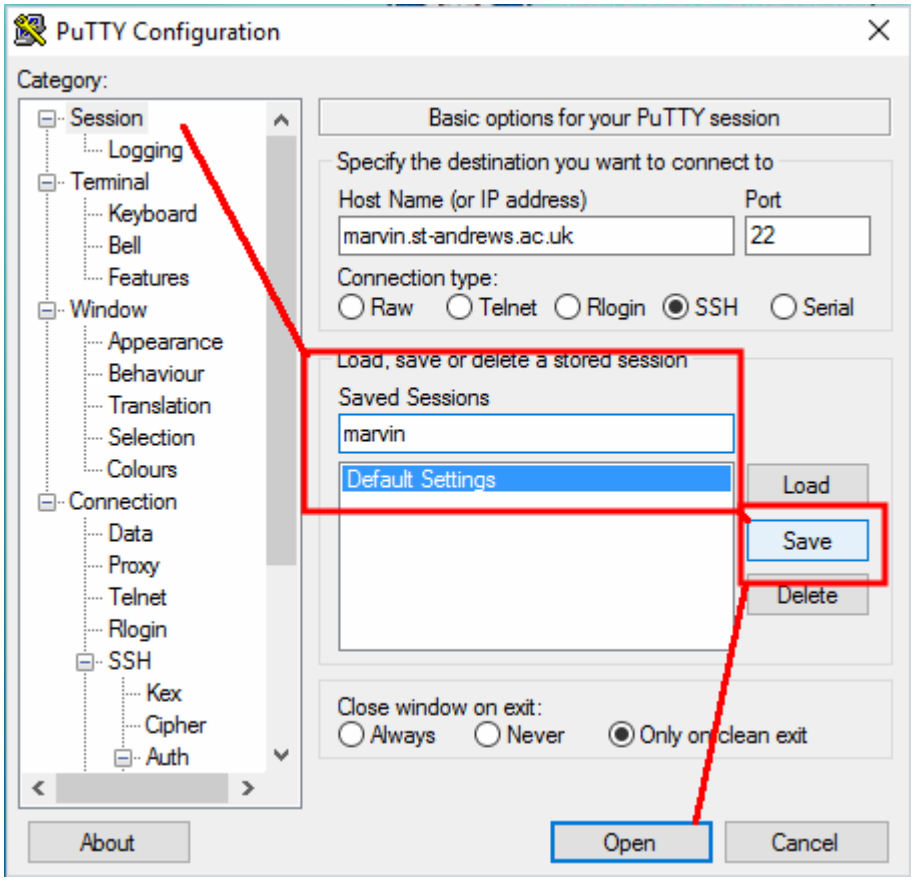


- Connection:Data → Login details is your username

Logging in.

- Bigger font!
  Window:Appearance→ Click change and turn it up to 12 or 13. Whatever you want.





- Session→ Name the settings and click save.
- Click open at the bottom

What can you see

- Scary? Not sure what to do? SO MUCH POWER
- Currently a big bucket of unknown-unknowns.

Where am I? (Navigating)

- pwd
- This **p**rints the **w**orking **d**irectiory (tells you where you are).
- If you forget where you are, or need to know where you're going next, use this. I do. Often.

  (I get lost easily).
- /storage/home/users/<your username>/
  - This is your "home" folder.

List what’s in a directory

- ls (**li**st, (it’s an L not an i))
- Use this often. Very often.
- Colors indicate folders or files

Changing directory

- cd unixCourse
- This changes where you are.
  - In this case, it moves us into the “unixCourse” directory.
  - Now run pwd, then ls again
- Move into the where folder, then into the list folder.
  - need a hand? Try: cd where, then ls, then cd list

Anatomy of a command

- Example:

  $ cd unixCourse

  - First thing is the command we're running (here, change directory)
  - Second thing is the argument to the command (here, the directory we want to move to)

- Often we'll have other flags in it too

  $ ls -l ./

  - Flags change the behavior of the command.

More complex ls

  - (make sure we're in the ~/unixCourse/where/list/ folder)
  - cd ~/unixCourse/where/list/

- Arguments can change the output of ls
  - Try ls
  - Try that example: ls -l ./
  - Notice the difference? The -l means output as a list. ./ means "the current folder we're in" (more on that later)
  - More complex arguments: list (l) by date created (t), all files including hidden ones (a)
  - ls -lta ./

Wild cards

- * means anything and any number of them
- ? means any one character
- ls -l a* (shows everything starting with a)
- ls -l *.csv (shows all .csv files)
- ls -l * (what did this do? Why?)
- ls -l ?f.*

How do we know this stuff?!

- man <command> to see the manual
- Google
  - “<thing you want to do> command line”
  - “how do I <do thing> on linux”
- If man doesn’t work, try
  - <command> -h
  - <command> --help
  - Try man ls (q exits)
  - Try ls --help

Tab Completion

- Laziness is your friend
  - Less typing means less effort and fewer mistakes.
- ls -l t<tab> (fancy, eh?)
- ls -l a<tab twice> (shows you the options you have)
- **Remember this.** It makes life a **lot** easier. (We'll come back to it in a minute)
- Also works on the first word. Try l<tab twice>. It gives you a list of all commands that start with l.

Shortcut to commands

- Up arrow and down arrow scroll through old commands

- history shows a list of commands you've already used.
- !<number> re-runs the command next to that number.

Escaping from where you are

- cd .. (goes up a level)
  - I.e if your pwd says /home/<username>/folder, using cd .. takes you to /home/username/
- cd - (takes you back to where you previously were)
- cd (takes you home)

Tasks:

- Move to the unixCourse/where/tabCompletion folder.
  - Hint: we're in ~/unixCourse/where/list so going up a folder should get us closer to where we need to be, then move ino the tabCompletion folder
  - Play about with tab completion using ls<tab, repeatedly>

Absolute and relative paths

- ~ is your home folder (/storage/home/users/<username>)
- ../ is the folder above where you are.
- ./ is your current folder
- Move to the unixCourse/where/list folder
  - cd ~/unixCourse/where/list
  - This is an absolute path. Works regardless of where you are
- Move to the unixCourse/where/mv_cp_rm folder
  - cd ../mv_cp_rm
  - This is a relative path, it only works if you're in the right place.

Moving and renaming files

  - (make sure you're in ~/unixCourse/where/mv_cp_rm/)
  - cd ~/unixCourse/where/mv_cp_rm/
  - This isn't just a silly exercise, it's similar to my day-to-day cleanup in folders I'm working in, just less dull.
- Situation: the garden is messy and the as-yet un-named rabbits have escaped.

- What have we got? (ls -l)
- Moving files uses the command mv
  - How do we find out how to use it?

mv – from the man page

- man mv

  NAME

  mv - move (rename) files

  SYNOPSIS

  mv [OPTION]... [-T] SOURCE DEST

- Remember q quits

mv

- Move the balls into the boxes
  - mv blue.ball BlueBox/ (repeat for all the balls).
  - Tab complete will save you effort.
- Move also renames things. Name the all the rabbits!
  - mv biggest.rabbit Elvis.rabbit
- Remember wildcards? Move all the rabbits!
  - mv *.rabbit RabbitHutch/ (feel free to name them all)
  - Check they’re all snuggled tight (ls RabbitHutch/)
- Mv also works for folders. Move the hutch into the garage
  - mv RabbitHutch/ Garage/
  - Check them again! (run ls on the RabbitHutch in the Garage)

Copy with cp

- Works the same as mv, mostly.
- Copy all the tools into the garage (yes, my analogy is failing here, sorry).
  - cp *.tool Shelf/
  - Works on folders too…
  - cp Shelf/ Garage/

Copy with cp

- Works the same as mv, mostly.
- Copy all the tools onto the shelf (yes, my analogy is failing here, sorry).
  - cp *.tools Shelf/
- Works on folders too…
  - cp Shelf/ Garage/
- But only if you tell it to do it recursively (which means it copies all of the contents too)
  - cp -r Shelf/ Garage/
    - (who has a shelf in their garden anyway?)

Removing things - rm

- rm IS A ONE WAY PROCESS. NO GOING BACK.
- Really really think hard about what you're doing.
- We're done with the tools, lets remove them all
- rm spade.tool this will remove the spade.tool file
- rm *.tool (this will remove ALL THE THINGS ending with .tool. Is this what we want to do? Will we need the pickaxe again?)
- Same as cp for folders, needs -r
  - rm -r Shelf/
  - This is even more dangerous. Be extra careful.

Removing things - rm

- rm IS A ONE WAY PROCESS. NO GOING BACK.
- Worried about it? Use rm -i <files>
  - Forces you to confirm each deletion.

Making things – mkdir and touch

- Move into the hutch (cd Garage/RabbitHutch/)
- Mkdir **mak**es the **dir**ectory.
  - Give the rabbits a bed directory
  - mkdir bed
- Touch, weirdly, creates empty files
  - Give the rabbits some straw
  - touch bed/straw.txt
  - Not just text files, you can name it whatever

Find

- There were 4 rabbits. We've lost one!
  - Seriously, I lose files more often than I'd like to admit.
- Lets *find* it.
- find ./ -name "*.rabbit"
- find ~/unixCourse/ -name "*.rabbit"

- Can you move it back into the hutch?

What

- Change directory to ~/unixCourse/what/

- What’s in the files?

Head and tail of a cat, more or less

- Many ways to see what's in a file
- cat randomlyGeneratedStory.txt
  - Prints the entire contents to the command line
- head randomlyGeneratedStory.txt
  - Prints the top few rows, try it with -n 1
- tail randomlyGeneratedStory.txt
  - Prints the bottom few rows, try it with -n 1
- less randomlyGeneratedStory.txt
  - Up and down arrows navigate. Space jumps pages
  - Search with /<word> (try /squash)
  - Exit with q

Grep - basics

- less bigListOfGenes.csv
  - When you've done looking at the many genes, quit with q
- grep <what you're searching for> <file>
  - If you forget either, it'll wait. Forever. Try it!
- grep bigListOfGenes.csv
  - "kill" the process with ctrl-c

Grep - basics

- grep <what you're searching for> <file>
- grep CYP51 bigListOfGenes.csv
  - Searches the file bigListOfGenes.csv for CYP51
- grep cyp51 bigListOfGenes.csv
  - It's case sensitive!
- grep -i cyp51 bigListOfGenes.csv
  - -i makes it ignore case
    - Point of note: You can grep everything in a folder using
      - grep <search term> ./* but this is just everything in the folder.
      - grep -R <search term> ./* will also search sub-folders.
    - Point of note 2: If you need to grep a compressed file, use zgrep

Wc and pipes

- wc counts the number of words, but it'll also count the number of lines with -l
- wc -l bigListOfGenes.csv

- What if we want to know the number of a specific gene family in the list?
  - Eg Forkhead box genes (FOX)

Wc and pipes

- wc counts the number of words, but it'll also count the number of lines with -l
- wc -l bigListOfGenes.csv

- What if we want to know the number of a specific gene family in the list?
- grep -i fox bigListOfGenes.txt | wc -l

Saving output

- > overwrites what's already in the file
- >> adds to the end of the file

- head -n 1 bigListOfGenes.csv > foxGenes.csv
  - Check what's in the file with cat foxGenes.csv
- grep -i fox bigListOfGenes.csv >> foxGenes.csv
  - Check it again
- echo "whoops" > foxGenes.csv
  - One last check. What happened?

Cat revisited

- Can use cat to merge files together
  - cd ~/unixCourse/what/cat/
- less aGene.fasta
- cat *.fasta > all.fasta
- less all.fasta

Practical example

- Move back to the ~/unixCourse/what/ folder
- Genuine example.
- Open the bigListOfGenes.csv
  - less bigListOfGenes.csv
- Are there any entries with NA on them?
  - /NA searches the file for any “NA”s, / repeats the search and moves to the next item.
- Have a look at the lines that have NA in them.
  - Is there anything odd/unexpected in them?
- How many?

Practical example

- Genuine example.
- Open the bigListOfGenes.csv
  - less bigListOfGenes.csv
- Are there any entries with NA on them?
  - /NA
- Have a look at the lines that have NA in them.
  - grep NA bigListOfGenes.csv
- How many?
  - grep NA bigListOfGenes.csv | wc -l

Practical example

- We want a list of all the genes, without the ones with NA on the padj row (at the end of the line).
    - How do we find this out?

Practical example

- We want a list of all the genes, without the ones with NA on the padj row (at the end of the line).
  - grep NA$ bigListOfGenes.csv
  - grep -v NA$ bigListOfGenes.csv
- So now we need to save this…

- Suggestions?

Practical example

- We want a list of all the genes, without the ones with NA on the padj row (at the end of the line).
  - grep NA$ bigListOfGenes.csv
  - grep -v NA$ bigListOfGenes.csv
- So now we need to save this…

- grep -v NA$ bigListOfGenes.csv > bigListOfGenes_removedNA.csv

Practical example

- Open the new file, check it.

- Does it look right?

Editing files with Vim

Editing files with Nano

- nano randomlyGeneratedStory.txt
- ^<letter> means <hold control><press letter>
- Arrows navigate, but <ctrl>V (down) and <ctrl>Y (up) skip pages
- “Write out” means “save” (<ctrl>o)
- Exiting is <ctrl>-x

- Bonus! - Undo: <alt>u, redo: <alt>e

Compressed files

- Tar (archive, not compressed)
- gzip, bzip2, zip.
- Listing contents
- Uncompressing
  - tar -xvf <file> for .tar files (also works on .tar.gz)
  - gunzip <file> for .gz files
  - bunzip2 <file> for bz2 files
  - unzip <file> for .zip files
- tar -xvf compressedFiles.tar.gz

Exercises

- Check the ~/unixCourse/exercises/ folder
  - Read the README.txt files for guides.

- For the renderToTSV folder:
  - http://stab.st-andrews.ac.uk/wiki/index.php/Hdi2u_rendertotsv_exercise

- Really far ahead:
- http://www.docs.is.ed.ac.uk/skills/documents/3523/3523.pdf
- The murder.tar file is in ~/unixCourse/exercises/

Covered so far:

- Where am I?
  - pwd
  - ls
  - cd
  - mv
  - cp
  - rm
  - mkdir
- What's there?
  - less
  - cat
  - head, tail
  - nano
  - grep
  - zipped files
  - pipes (|)
  - redirects (> and >>)

Toolbox

- Loops
- Scripts
- Manipulate the contents of files
-
- dos2unix, mac2unix

Situation: My research

- RNA-seq analysis
- I’ve run 2 tools to pseudo-align the reads (Kallisto and Salmon)
- I’ve then run 3 tools on each of those (sleuth, edgeR, DESeq2)
- Result: lots of data.

- Caveat: This isn’t the best way to approach this specific problem, but it’s a convenient way to teach you loops.

Variables.

- Move to ~/unixCourse/where/list/

- Prefixed by $
- Save information for use later
- echo $HOME
- allFiles=$(ls *.txt)
- echo $allFiles

Loops

- Move to the loops folder
  - cd ~/unixCourse/loops/
- How many files do we have?
  - ls | wc -l
- Can we be bothered to run that grep line on 56 different files manually?
  - Hint: Nope.

Anatomy of a for loop

- for <variable> in <things>;

  do <action or actions you want to do to <variable>>;

  done

- What are we looping over?
- What is it we want to do?
- We can name our variable anything!

Anatomy of a for loop

- for <variable> in <things>;

  do <action or actions you want to do to <variable>>;

  done

- What are we looping over?
- All of the csv files
- $(ls *.csv)

Anatomy of a for loop

- for <variable> in $(ls *.csv);

  do <action or actions you want to do to <variable>>;

  done

- What is it we want to do?
- Our grep line from earlier:
- grep -v NA$ bigListOfGenes.csv > bigListOfGenes_removedNA.csv

Anatomy of a for loop

- for <variable> in $(ls *.csv);

  do <thing or things you want to do>;

  done

- What is it we want to do?
- Our grep line from earlier:
- grep -v NA$ <variable> > <file name but without .csv?>_removedNA.csv

Anatomy of a for loop

- for currentFile in $(ls *.csv);

  do <thing or things you want to do>;

  done

- What is it we want to do?
- Our grep line from earlier:
- grep -v NA$ $currentFile > <file name but without .csv?>_removedNA.csv

Anatomy of a for loop

- <file name but without .csv?>
- New tool: basename
- basename ~/unixCourse/loops/edgeR_cond1_left_dark24.csv
    - Strips the directory from the filename
- basename ~/unixCourse/loops/edgeR_cond1_left_dark24.csv **.csv**
    - Strips the directory *and specified extension* from the filename

Anatomy of a for loop

- for currentFile in $(ls *.csv);

  do <thing or things you want to do>;

  done

- What is it we want to do?
- Our grep line from earlier:
- grep -v NA$ $currentFile > $(basename $currentFile .csv)_removedNA.csv

Testing a for loop

- for currentFile in $(ls *.csv);

  do echo “grep -v NA$ $currentFile > $(basename $currentFile .csv)_removedNA.csv”;

  done

- Test the loop with echo: prints the command so we can check it’s right instead of blundering in and running it all (to potential disaster).

Anatomy of a for loop

- for currentFile in $(ls *.csv);

  do grep -v NA$ $currentFile > $(basename $currentFile .csv)_removedNA.csv;

  done
- Check it created the files, and check they look right (i.e. no NA)
- Thought process:
  - Identify what we need to loop over
  - Identify what we need to do on each item
  - Write the loop. It's really easy when you have to do it lots!

Scripts

- Files used to re-run things you've already written.
- Next level of lazyness.

- Text file ending in .sh

- Move into the scripts folder
  - cd ~/unixCourse/scripts/

Scripts: making the file

- Nano removeNAEntrys.sh
- Type in the loop we wrote before:
- for currentFile in $(ls *.csv);

  do grep -v NA$ $currentFile > $(basename $currentFile .csv)_removedNA.csv;

  done
- <ctrl>O to save, then his enter when it asks for the file name
- <ctrl>X to exit.

Scripts: running them

- sh removeNAEntrys.sh

- Errors: Read Them.

Comments

- # at the start of the line
- Doesn’t do anything to the code, does help you remember what it does!

- Add a comment to the script.
  - Open it with nano
  - Add

    #for each of the .csv files in the current folder, this script creates a new file that has no lines that end with NA

More complex scripts

- What we wrote only works for the current folder
- Arguements: same as we're passing to other programs
    - cd <folder>, grep <search for> <file>
- Use $1, $2, $3… to access the first, second, third etc argument passed

- targetFolder=$1

  for currentFile in $(ls $1/*.csv);

  do grep -v NA$ $currentFile > $(dirname $currentFile)/$(basename $currentFile .csv)_removedNA.csv;

  done
- sh removeNAEntrys.sh args/
- What happens if we run it without an argument now?

Replacing with sed

- Removing lines with NA isn't the best solution
  - (In fact, I'd argue it's a bad solution).
- It's adjusted p-value, so replacing it with 1 will be a much better solution (it'll get filtered out in the analysis).

- cd ~/unixCourse/sed/

sed

- **S**tream **ed**itor.
  - Very powerful, very versitile, very baffling.
- sed ‘s/NA$/1/g’ file.csv > outputFile.csv
  - Substitute NA at the end of the line with 1, globally (not just the first time)
- sed -i ‘s/NA$/1/g’ file.csv
  - Edits the file inline (directly). You make a mistake here, there’s no going back.
- sed ‘/NA$/d’ file.csv > outputFile.csv
  - Find lines with NA at the end, delete them.

Sed – lets try it...

- for currentFile in $(ls *.csv); do sed ‘s/NA$/1/g’ $currentFile > $(basename $currentFile .csv)_NAreplaced.csv; done
  - Script or command line, your call.
  - Don’t just copy it. Think about what each bit is doing.

- Check what we’ve done with ls, open the files and search for what we replaced (/NA) and make sure it did what we expected.

cut

- Sed/grep can do rows, but what about columns?
  - Again, semi-real situation.
  - We only want columns 1 (gene name), 4 (log2foldchange) and padj (6) for DESeq files
- cut -d , -f 1,4,6 DESeq2_cond1_dark6.csv
  - -d , means “columns are split by ,”
  - -f 1,4,6 means we need columns 1, 4 and 6
- Use head to check what columns we want from edgeR_cond1_dark6.csv, then try cut on that.

Dos2Unix & Mac2Unix

- Something to be aware of!
- Dos and *old* Macs use different line endings
- If you get weird errors that say thinks like “character encoding” or “unicode not found for...” run dos2unix or mac2unix on the file

Keep Practicing

- Mac: Unix based already! Open Terminal
- Windows: Harder. Download gitBash
  - https://git-for-windows.github.io/
  - Windows10 can get a proper unix command line https://www.howtogeek.com/249966/how-to-install-and-use-the-linux-bash-shell-on-windows-10/

- Get stuck? Google.

More resources

- http://rik.smith-unna.com/command_line_bootcamp/?id=yuw06k9pw3o
  - Online resource for learning command line, including brower based command line
- https://www.ed.ac.uk/information-services/help-consultancy/is-skills/catalogue/program-op-sys-catalogue/unix1
  - Edinburgh’s Introduction to Unix course materials

Feedback forms

- Please fill them out! We want to improve!