
Welcome to Data Carpentry Genomics!

Important Links:

Short link for this Etherpad: https://go.wisc.edu/3ge5gt
Workshop Website: https://uw-madison-datascience.github.io/2024-08-05-uwmadison-dc/
Curriculum: https://datacarpentry.org/genomics-workshop/
Pre-workshop survey: https://carpentries.typeform.com/to/wi32rS?slug=2024-08-05-uwmadison-dc
AWS Instances: https://pad.carpentries.org/2024-08-05-uwmadison-dc-instances
Intro slides:
https://docs.google.com/presentation/d/1Z95pVjpFaQmdvOpmnPF_neumnuwQZ4T6lMJI5K7P-6w/edit#slide=id.g273d1143ff7_0_170
Feedback form: https://forms.gle/MLh8gUWiukCodNtf7
Wrap-up Slides: https://docs.google.com/presentation/d/1YgFjVi7pTnsg97uNJv-5hyTlmTyDAPkPus4jgSr8yBI/edit?usp=sharing

Day 1

Sign-in (Name, Dept, Describe your research/work in a couple of sentences)

- Sarah Stevens (she/her/hers), Data Science Hub, I help researchers learn and apply data sicence and computational tools to their work.
- Heather Shimon (she/her), Science & Engineering Libraries, I support research data management
- Trisha Adamus (she/her), Ebling Library - Instructor
- Zekai Otles (He/him) DoIT- Research Cyberinfrastructure consultant
- Anita Bhattacharyya, Waisman Center, SMPH, faculty researcher, I study human brain development using human stem cells, I want to learn how to analyze the multiome sequencing data generated by my lab.
- Heidy Elkhaligy , Biophysics graduate program , working on CryoEM method development
- Rachel Kirchner (she/her), Dept of Medicine, wanting to learn how to analyze genomic data to

conduct family-based (and individual) variant analyses for identification of novel causative loci for disease (I specifically study telomere biology disorders). also learn about general genetic data analysis!
- Balendra Sah (Wisconsin Enbery Institute) Interested in analysing NSG data using R
- Hailey Hessler, Biotechnology Center, analyzing WGS data through R.
- Celeste Huff (she/her) , Department of Entomology/ Plant Pathology, I am researching fungal movement (including yeasts) in cranberry flowers, bees, and flies
- Lukas Matthews, (he/him), Information School Master's program. I am a student looking for new research projects.
- Avelina Gaston (she/her), Plant and Agroecosystem Sciences Dept, graduate student in Plant Breeding and Genetics studying carrot texture, looking to learn about genomics data management
- Lisa Abler (she/her), Science & Engineering Libraries, background in the sciences (Cell and Molecular, Developmental BIology), here to observe to learn about the Carpentries
- Madeleine Lodes (she/her), Bacteriology Master's Program, working on Tuberculosis and SARS-CoV-2. Hoping to learn how to use coding language to work through large data sets.
- Adel Talaat, Pathobiological Sciences, interested to learn more about R language and coding in general.
- Erwin Lares - Research Cyberinfrastructure - I lead the Data Science Platform
- Emilie Greene (she/her), Cancer Biology graduate student, investigating EBV assocaited cancers and changes in the transcriptome
- Kaustubh Amrikar (he/him), Biophysics graduate student, exploring the evolution of RuBisCO enzyme

Notes

To start AWS instance

- open gitbash (windows), or your terminal (mac)
- AWS Instances: https://pad.carpentries.org/2024-08-05-uwmadison-dc-instances
  - write your name next to the instance that you will use
- Login via the terminal (“ssh dcuser@PASTE_AWS_INSTANCE HERE”, use right click to paste for windows)
- press enter
- For question "Are you sure you want to continue connecting?" type: yes and press enter
- Username: dcuser , Password: data4Carp (password won’t show up as you type it) and press enter
- Log out (type “exit”)
- After the workshop
  - instances will go away
  - you can still use the unix shell from your terminal/gitbash
  - you can use RStudio on your local computer
  - other software can be installed as needed

Helpful tip: you can start over in the terminal by pressing control + c

Project Management:
Slides:
https://docs.google.com/presentation/d/1yLIG8kM4njehBCZEL6nsRB0v12JuV4I3ODTypOzLq_0/edit?

usp=sharing

Experimental setup
12 populations
24 samples
x 11 time points =264 samples!

For each sample, sent to NGS and received 2 FASTQ files
For each FASTQ file, using breseq against reference genome to get mutation counts > need to document process

Project organization is important!
Book: the life-changing magic of tidying data, by dr. tracy teal

Data Tidiness

- documentation
- spreadsheet rules
- machine readability
- file/variable naming

Discussion: What kinds of data and information have you generated before you send your DNA/RNA off for sequencing?

- DNA quantity (ul/ng)
- DNA/RNA initial quality information (determine if high enough quality to submit for sequencing), concentration of DNA/RNA, your experimental set-up (ie. how many samples, organism, hypothesis, etc.)
- age, gender of sample; cell type; experimental condition (e.g. disease/ control), some reference to project
- DNA purity, 260/280 ratios, 260/230
- name of the researchers and date the experiment took place
- plot numbers, pedigrees, seed sources, phenotype
- Tissue type (species, organ)
- Name sample, type of sample (PCR or plasmid) detail of oligos and amount of DNA, expected size of sequencing read
- Sample ID, treatments and the number of replica used
- lab notebooks notes that informs the way the experiment was conducted

Recording experimental info

- Where to record
    - in a paper notebook
    - in an electronic lab notebook (ELN)

What to record

- Dates (also on samples and records)
- creates unique identifiers

Data about the experiment

- collected in spreadsheets
- content depends on the type of experiment
- metadata standards (Genomics Standards Consortium)
- look for standards for your own data types

Structuring data in spreadsheets

- makes it easier to analyze later
- making it human readable (using color coding, etc.) makes it harder to code
- think like a computer - how to format for the machine to process

Using spreadsheets wisely

- save a copy of the raw data
- one row per oberservation (e.g., sample)
    - one colun per variable (e.g., strain)
    - use good column names (e.g., DNA-concentration)
- dont' combine related variables (e.g., E. coli K12)
- export the cleaned data to a text-based file (e.g., .csv)

Tidy data: http://r4ds.hadley.nz/
More reading on tidy data: https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html

Good naming conventions

- be descriptive and meaningful, pick 2-3 key things
- be brief (fewer than 25 characters if possible)
- no spaces or special characters \ / : * < > [ ] & $
- format the data as YYYYMMDD or YYYY-MM-DD

Discuss: How could the spreadsheet be improved?

- "Clade" column includes special characters
- Spaces in some of the strain names
- each row isn't its own observation with the multiple tables
- some rows say "none" and some are left blank
- data entries are not consistent; e.g. Population column
- Not saved as CSV
- 1 row, 1 sample

More information on data structure in the DC Ecology Spreadsheet lesson:
https://datacarpentry.org/spreadsheet-ecology-lesson/02-common-mistakes

OpenRefine is a great tool for data cleaning: https://openrefine.org/

Discussion: What challenges do you think you’ll face (or have already faced) in working with a large sequence dataset?

    - What is your strategy for saving and sharing your sequence files?
    - How can you be sure that your raw data have not been unintentionally corrupted?
    - Where/how will you (did you) analyze your data - what software, what
- Inconsistent data management and metadata documentation by different people in lab; need to spend time and effort to decipher
- Struggling with data sets that take hours to download/upload etc. with my computer's processing/memory power. I have been storing data on a large external hardrive. I have been using CLC genomics workbench, QluCore, Galaxy.
- maintaining HIPAA compliance (certain versions with patient identifiers/names and certain versions without actual names to share/analyze)
- Store data on multiple devices. Check files once received. Will try to use cloud computing.
- One concept for checking for data corruption I'd love to learn how to improve or automate: Checking filter values. This works well for categorical textual data, but I'm less certain how to review for errors in quantitative data.

Exercise:

- What are some errors you can spot in the data? Typos, missing data, inconsistencies?
- What improvements could be made to the choices in naming?
- What are some errors in the spreadsheet that would be difficult to spot? Is there any way you can test this?

Sample submission sheet: https://datacarpentry.org/organization-genomics/files/sample_submission.txt
**

- Download the file using right-click (PC)/command-click (Mac). If no download option, choose "Save as" and put it into your downloads
- This is a tab-delimited text file. Try opening it with Excel or another spreadsheet program.

- date formating is day first
- "wild type" has a space
- only one plate-is barcode needed?
- date should be in YYYY-MM-DD format, or at least include the year
- The replicate column has both lower and upper case letters
- Use greek letter on some headings

Store data properly

- accessiblity - can you and your PI access the data and open it?
- backup - 3 copies, 2 media types, 1 which is offsite
- leave the raw data raw - can always go badk to step 1 if you've made a mistake

Finding NGS data - SRA database

Exercise: Find the data

- access the Tenaillon dataset from the provided link: https://trace.ncbi.nlm.nih.gov/Traces/study/?acc=SRP064605

- “Common Fields” “Select”, and “Found 312 Items”
- In “Found 312 Items”, click on the first Run Number (Column “Run” Row “1”)
- “Select” has the Total row showing you the current number of “Runs”, “Bytes”, and “Bases” in the dataset to date
- Click on the “Metadata” button to download the data for this lesson
- Open excel and open the data file, delimited by comma. You may need to adjust the column or wrap the text to read and understand the data

Questions

- What strain of *E. coli* was used in this experiment?

Escherichia coli B str. REL606 shown under the “organism” column. This is a tricky question because the column labeled “strain” actually has sample names

- What was the sequencing platform used for this experiment?

The Illumina sequencing platform was used shown in the column “Platform”. But notice they used multiple instrument types listed under “Instrument”

- What samples in the experiment contain paired and sequencing data?

Sort by LibraryLayout and the column “DATASTORE_filetype” shows that “realign,sra,wgmlst_sig” were used for paired-end data, while “fastq,sra” were used for all single-end reads. (Also notice the Illumina Genome Analyzer IIx was never used for paired-end sequencing)

- What other kind of data is available?

There are several columns including: megabases of sequence per sample, Assay type, BioSample Model, and more.

- Why are you collecting this kind of information about your sequencing runs?

Examples that you will need to send to public databases to meet grant funding requirements

Unix Shell

- Many programs do not have a GUI (graphical user interface), so getting used to command line is useful
- good at repetitive tasks
- can work with many samples at once
- when a cloud instance for large datasets is needed, the command line is how to communicate across platforms

# Accessing AWS instances
1. Open Git Bash (windows) or terminal (Mac)
2. ssh dcuser@(copyAndPasteYourInstance), e.g., ssh dcuser@ec2-3-236-71-133.compute-1.amazonaws.com

- - Your instance will start with ec2 and end with amazonaws.com. Grab the instance from here: https://pad.carpentries.org/2024-08-c05-uwmadison-dc-instances

3. You'll get a message saying, "Are you sure you want to continue connecting? Enter yes followed by enter
4. Enter in your password (same for all of us): data4Carp

```
# clear screen
clear

# change prompt info
PS1='$ '

# find where you are with pwd (print working directory)
pwd

# output is /home/dcuser

# often when you get an error, it might mean that you are in the wrong directory, try pwd to find where
you are as the first troubleshooting technique

# list all of the files in the working directory that you are in
ls

# output is R r_data shell_data

# change directory with cd and directory that you want to go to
cd shell_data

# see that you've changed directories
pwd

# output is /home/dcuser/shell_data (we have gone down a level into the directory)

# return to home directory
cd

# use tab complete to return to shell_data folder
cd sh   # tab button to complete the name of the directory

pwd   # we are back in shell_data folder

# see what is in the shell_data folder
ls

# Can use up arrow to call past commands
# case matters

# use the manual to get information on a command -- man + command
man ls

# scroll with down arrow
# space bar to jump forward
# back pages with B key
# q to quit
```

-F   # classify with F flag, add this to your command

ls -F

# F flag gives more information about what appears in your list - if it is a file, program*, directory/
# or can use
ls --classify

Challenge
Use the -l option for the ls command to display more information for each item in the directory. What is one piece of additional information this long format gives you that you don’t see with the bare ls command?l

ls -l

# output
```
total 8
drwxr-x--- 2 dcuser dcuser 4096 Jul 30  2015 sra_metadata
drwxr-xr-x 2 dcuser dcuser 4096 Nov 15  2017 untrimmed_fastq
```

# we are getting more information about the directories- date last modified, owner of the file, permission to read/write

# more about permissions: https://www.redhat.com/sysadmin/linux-file-permissions-explained

# change directories
cd untrimmed_fastq/

# pwd to make sure we are in the correct directory
# output is /home/dcuser/shell_data/untrimmed_fastq

ls -F

# output is SRR097977.fastq SRR098026.fastq
# we can see that they are fastq files

# return to home directory (can use cd without directory name)
cd

pwd
# output is /home/dcuser

# return to shell_data directory
cd shell_data

cd untrimmed_fastq/

# we stepped down by directory

```
# but can do this in one

# return to home directory
cd

# jump more than one directory at a time
cd shell_data/untrimmed_fastq/

ls

# output is two files SRR097977.fastq  SRR098026.fastq

# want a list of what is in the files with SRR
ls SRR09 -- tab twice -- then hit 7 and tab complete

# output is SRR097977.fastq

ls SRR098    # tab complete

# output is SRR098026.fastq

# find command names with tab complete
pw     # double tab to get options for commands

# back to home directory
cd

pwd   # to check

cd shell_data

cd untrimmed_fastq

# return to shell_data but can't tab complete
# can only cd to a directory that is inside the directory where you are
# to go back a directory cd ..

cd ..

pwd
# output is /home/dcuser/shell_data

ls
# output is: sra_metadata  untrimmed_fastq

# see everything in the shell_data directory
ls ..

# output is: R  r_data  shell_data
```

ls ../..   # to see the directory above the directory where I am, output is: dcuser ubuntu

Challenge: Finding the Hidden Directories
First navigate to the shell_data directory. There is a hidden directory within this directory.
Explore the options for ls to find out how to see hidden directories.
List the contents of the directory and identify the name of the text file in that directory.

Hint: hidden files and folders in Unix start with ., for example .my_hidden_directory

# we are in the shell_data directory
# if not there

cd
cd shell_data

# looking for hidden directories
# how to explore options for ls to find hidden directories

man ls

# look at options in manual
-a , --all (do not ignore entries starting with .

ls -a

# output is . .. .hidden sra_metadata untrimmed_fastq

# identify the name of the text file in the directory

cd .hidden

pwd   # in hidden folder

ls

# output is youfoundit.txt

# can combine flag

cd..
# back to shell_data

ls -Fa
# now I can see what is a diretory or file even with hidden stuff

# return to .hidden folder
cd .hidden/

# return to home
cd

pwd
# back here /home/dcuser

# just a / is your root directory, not your home directory, which is home/dcuser

# to get give absoulte path from root directory

cd /home/dcuser/shell_data/.hidden/

# or can do relative path
# cd back to home

cd

# from home directory, tell the path from there

cd shell_data/.hidden/

pwd   # output is /home/dcuser/shell_data/.hidden

Challenge: Relative Path Resolution
Using the filesystem diagram below, if pwd displays /Users/thing, what will ls ../backup display?
https://datacarpentry.org/shell-genomics/fig/filesystem-challenge.svg

1. ../backup: No such file or directory
2. 2012-12-01 2013-01-08 2013-01-27
3. 2012-12-01/ 2013-01-08/ 2013-01-27/
4. original pnas_final pnas_sub

# Think through this
pwd   # output is /Users/thing

ls ../backup   # go up one level to Users, then to backup folder in Users folder
it will display: original pnas_final pnas_sub # answer is #4

# return to home directory
cd

# do not recommend to cd to root directory /

# even if you are in the root directory, you can use cd~ to get to home directory
cd ~

# more thought exercises on absolute and relative paths! https://swcarpentry.github.io/shell-novice/02-filedir.html#absolute-vs-relative-paths

# Question: are there hidden directory in real world datasets?
# usually things in hidden folders are there so you don't access them, these are usually configuration files
# .gitignore is a good example of a hidden file

# we will cover how to see file content tomorrow!

# Please share your feedback for today! https://forms.gle/MLh8gUWiukCodNtf7

# we will address questions in the feedback tomorrow

exit   # to leave AWS instance

Day 2 - More Unix Shell

Sign-in

- Sarah Stevens (she/her/hers)
- Heather Shimon (she/her) Science & Engineering Libraries
- Zekai Otles (He/him) DoIT-Research Cyberinfrastructure consultant
- Trisha Adamus (she/her) - Ebling Library - Helper
- Celeste Huff (she/her) - Dept. of Plant Pathology/ Entomology
- Erwin Lares - he pronouns - RCI
- Rachel Kirchner (she/her), Dept of Medicine
- Madeleine Lodes (she/her) Bacteriology
- Lisa Abler (she/her) - Science & Engineering Librarian
- Adel Talaat, Pathobiological Sciences.
- Hailey Hessler (she/her), Biotech Center
- Balendra Sah (He/Him), Wisconsin Energy Institute
- Avelina Gaston (she/her)
- Lukas Matthews (he/him), iSchool grad student.
- Anita Bhattacharyya
- Emilie Greene (she/her)

Feedback Reponse:

Things you liked learning/found useful (summarized)

- navigating directories x3
- going slowly through commands x3
- challenges
- data organization
- 3-2-1 rule for back-ups

Questions and confusions (summarized/paraphrased and with replies)

- Get stuck navigating with onedrive on my home computer. Also is this the same for navigating with a shared server/reserach drive?
  - A: One drive takes over several of your folders - for example your Desktop - and puts them inside its

own directory. Usually that folder is in your home folder so it would be ~/OneDrive/Desktop.
  - A: Navigating in a shared server is similar, each user has a home folder. Then there are sometimes shared locations maybe in something like /Shared/data/ or /scratch/. For something like research drive - you may have a home folder on the drive and other shared files if your group has setup some structure already. These can be mounted to your computer or other server, on a mac it would be in the /Volumes/ location typically.

- Is there a way to change how ls displays, for example always doing ls -F?
  - A: Yes! You can make what is called an alias. I used to have aliases to ssh to all the servers I connected to so I wouldn't have to remember/type all the addresses. Here is a link that explains it - https://www.geeksforgeeks.org/alias-command-in-linux-with-examples/

- There are lots of commands! How does one remember/learn them all?

- - A: You start to get used to the ones you use all the time - ls, pwd, cd, etc - others I usually look up by googling. Here are some useful cheatsheets - https://devhints.io/bash and https://www.alexji.com/UNIXCheatSheet.pdf

- Would love to learn more about the SRA

- - Commonly used to deposit data when you publish. Sometimes people use it for meta-analysis where they use a bunch of public data sets. Can't deposit sensitive/ personally idnetifiable data in it.

- Are we going to acess and modify files?

- - A: Yes! we will learn to do that today!

- accessing the amazon server

- - A: this will get easier, we will continue to practice it each day. Please let us know if you aren't able to get connected and we will help walk you through it.

- Identifying paired vs unpaired sequencing data

- - A: Usually you know this becuase it is your data and you submitted it to be sequenced in a particular way. But often times you can tell by the file names paired data usually have names that indicate it, for example ending in _1 and _2 or something along those lines. Also the paired read names have the same ID and then 1/2 or F/R or something. If you are interested in talking about which type of data might be better for your work, the biotech center can help or you can make an appointment with me

Notes:

# Accessing AWS instances
1. Open Git Bash (windows) or terminal (Mac)
2. ssh dcuser@(copyAndPasteYourInstance), e.g., ssh dcuser@ec2-3-236-71-133.compute-1.amazonaws.com

- - Your instance will start with ec2 and end with amazonaws.com. Grab the instance from here: https://pad.carpentries.org/2024-08-05-uwmadison-dc-instances

3. Enter in your password (same for all of us): data4Carp

# change prompt display
PS1='$ '

# cheatsheet for commands: https://devhints.io/bash
# also https://www.alexji.com/hidirectory
.pdf

# reminder that we are using AWS in order to use a large dataset, as if you are working in a lab, but you can use the commands that we are using on your local computer

# cd to untrimmed_fastq directory
cd ~/shell_data/untrimmed_fastq/

# use wildcard * to call many files at once, examples
# asterisk = any character including none

ls *.fastq   #   *.fastq means any filename that ends in .fastq

ls *977*

ls *977.fastq

ls *.fastq

# You can also use it to specify you want your program to run on multiple files
# Maybe you want it to run on all of the fastq files and you have 100, typing each one out
# instead you can use a wildcard

# reminder of absoulute vs relative paths: https://www.redhat.com/sysadmin/linux-path-absolute-relative

# ls /usr/bin/*.sh
# output is: /usr/bin/amuFormat.sh /usr/bin/gettext.sh /usr/bin/gvmap.sh

Exercise:
Do each of the following tasks from your current directory using a single ls (list) command for each:

1. List all of the files in /usr/bin that start with the letter ‘c’   # ls /usr/bin/c
2. List all of the files in /usr/bin that contain the letter ‘a’   # ls /usr/bin/a*
3. List all of the files in /usr/bin that end with the letter ‘o’   #ls /usr/bin/o

Bonus: List all of the files in /usr/bin that contain the letter ‘a’ or the letter ‘c’   # ls /usr/bin/*[ac]*

Hint: The bonus question requires a Unix wildcard [ ] that we haven’t talked about yet. Try searching the internet for information about Unix wildcards to find what you need to solve the bonus problem.

# echo to understand the command - see how the wildcard character is interpreted by the shell

echo *.fastq

# output is: SRR097977.fastq SRR098026.fastq

# command history
history

# will show line numbers next to commands
# can repeat the line with ! and its number: !LineNumber

!260

Exercise
Find the line number in your history for the command that listed all the .sh files in /usr/bin/c*.
clera Rerun that command.

# Solution: First type history. Then use ! followed by the line number to rerun that command.
mine !509

# Examining Files

pwd

# in the shell_data/untrimmed_fastq directory
# print all of the contents of a file using cat and file name

cat SRR098026.fastq

man cat   # to look at cat options

# less opens the file as read only, and lets you navigate through it
# less and file name

less SRR097977.fastq

# can navigate around, b to go back, shift + g to go to end of file

# head to see first 10 lines
# head and file name

head SRR098026.fastq

# tail for last 10 lines
tail SRR098026.fastq

# can add number of lines to head or tail with -n #of lines

```
head -n 20 SRR098026.fastq
```

```
tail -n 1 SRR098026.fastq
```

# press ctrl + c if you get stuck

# FASTQ format: more information here: https://datacarpentry.org/shell-genomics/03-working-with-files.html#details-on-the-fastq-format

Line	Description
1	Always begins with '@' and then information about the read
2	The actual DNA sequence
3	Always begins with a '+' and sometimes the same info in line 1
4	Has a string of characters which represent the quality scores; must have same number of characters as line 2

# view the first complete read in one of the files by using head to look at the first four lines

```
head -n 4 SRR098026.fastq
```

# output is

```
@SRR098026.1 HWUSI-EAS1599_1:2:1:0:968 length=35
NNNNNNNNNNNNNNNNCNNNNNNNNNNNNNNNNNN
+SRR098026.1 HWUSI-EAS1599_1:2:1:0:968 length=35
!!!!!!!!!!!!!!!!#!!!!!!!!!!!!!!!!!!
```

# The # character and each of the ! characters represent the encoded quality for an individual nucleotide.

```
# Quality encoding: !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJK
                    |         |         |         |         |
Quality score:      0........ 10........ 20........ 30........ 40..
```

# the quality of each of the Ns is 0 and the quality of the only nucleotide call (C) is also very poor (# = a quality score of 2). This is indeed a very bad read.

# Copying Files with cp
# cp OriginalFileName NewFileName

```
cp SRR098026.fastq SRR098026-copy.fastq
```

# check our files

```
ls -F
```

# output is: SRR097977.fastq SRR098026-copy.fastq SRR098026.fastq

# creating directories with mkdir and directoryName

```
mkdir backup
```

# check

ls -F

# output is: backup/ SRR097977.fastq SRR098026-copy.fastq SRR098026.fastq

# moving with mv

mv SRR098026-copy.fastq backup

# check with ls backup

ls backup

# output is: SRR098026-copy.fastq

# File permissions: recap: https://www.redhat.com/sysadmin/linux-file-permissions-explained
# in lesson: https://datacarpentry.org/shell-genomics/03-working-with-files.html#file-permissions

ls -l backup

# to see file permissions: -rw-r--r-- 1 dcuser dcuser 43332 Aug 6 15:48 SRR098026-copy.fastq

# change the permission on the file to protect it with chmod (change mode) command and subtracting (-) the write permission -w
# Changing the permissions can be helpful for protecting your data files so you can't accidentally delete them or modify them
# Also sometimes you will need to tell it that a program is something you should be allowed to be run (executed)

chmod -w SRR098026-backup.fastq

# check

ls -l

# output is now: -r--r--r-- 1 dcuser dcuser 43332 Nov 15 23:02 SRR098026-backup.fastq

# removing files with rm

rm SRR098026-copy.fastq

# You'll be asked if you want to override your file permissions

y

ls   # to see that file is gone

# Note! cannot undelete files!!!!!!

# be very careful when editing or removing files - especially raw data files
# kind of permission is important - read, write, etc.

Exercise: solutions: https://datacarpentry.org/shell-genomics/03-working-with-files.html#exercise-5
Starting in the shell_data/untrimmed_fastq/ directory, do the following:

1. Make sure that you have deleted your backup directory and all files it contains.
2. Create a backup of each of your FASTQ files using cp. (Note: You'll need to do this individually for each of the two FASTQ files. We haven't learned yet how to do this with a wildcard.)
3. Use a wildcard to move all of your backup files to a new backup directory.
4. Change the permissions on all of your backup files to be write-protected.

```
# searching files with grep (global regular expression)

# still in /shell_data/untrimmed_fastq

cd ~/shell_data/untrimmed_fastq       # if not there

# search for the string NNNNNNNNNN in the SRR098026 file

grep NNNNNNNNNN SRR098026.fastq

# looking for lines before and after the string NNNNNNNNNN

grep -B1 -A2 NNNNNNNNNN SRR098026.fastq

# redirecting the output into another file with > and outputFileName

grep -B1 -A2 NNNNNNNNNN SRR098026.fastq > bad_reads.txt

# word count wc to get line numbers

wc bad_reads.txt

# output is: 802    1338   24012 bad_reads.txt

#  to get number of lines use -l flag

wc -l bad_reads.txt

# output is: 802 bad_reads.txt

# search multiple FASTQ files for sequences

grep -B1 -A2 NNNNNNNNNN SRR097977.fastq > bad_reads.txt
```

wc -l bad_reads.txt

# output is: 0 bad_reads.txt

# avoid overwriting our files by using the command >>

grep -B1 -A2 NNNNNNNNNN SRR097977.fastq >> bad_reads.txt

# pipe command |
# the pipe takes the output from one command and uses that output as input to another command

grep -B1 -A2 NNNNNNNNNN SRR098026.fastq | less

# Note: The up arrow will take you back through your previous commands

# don't want to create a file before counting lines of output from grep search, can directly pipe the output of the grep search to the command wc -l

grep -B1 -A2 NNNNNNNNNN SRR098026.fastq | wc -l

# changing the number of NNNN will affect the number of counts

grep -B1 -A2 NNNNNNNNNN SRR098026.fastq | wc -l

# output: 537

grep -B1 -A2 NNNN SRR098026.fastq | wc -l

# output: 996

# showing matches per line - more reads that have NNNN or more, than NNNNNNNNNN

# loops to run commands on multiple files finding search patterns
# define a variable with foo

foo=test
echo $foo

# for loops

```
for filename in *.fastq      # enter
> do
> head -n 2 ${filename}
> done
```

```
for filename in *.fastq
> do
> grep -B1 -A2 NNNNNNNNN ${filename}
> done
```

```
# *.fastq matches two files in the current folder
# each time through the loop it assigns one of the files to the variable called filename
# then it substitutes in that filename into the ${filename} part of the body of the for loop and runs the command on that file
# then it repeats for every file that matches the pattern we gave it

# Basename for removing a uniform part of a name from a list of files

basename SRR097977.fastq .fastq

# output is: SRR097977

for filename in *.fastq
> do
> name=$(basename $filename .fastq) # this line runs the basename command and separates the .fastq extension off of the filename and assigns it to a variable called name
> echo $name $ this prints out the resulting filename
> done

# Think we are missing what generated the _2024.txt files here

for filename in *.txt
> do
> name=$(basename $filename _2024.txt)
> mv $filename ${name}.fastq
> done

# writing scripts in nano

# create nano bad-reads-script

nano bad-reads-script.sh

# enter text
grep -B1 -A2 -h NNNNNNNNNN *.fastq | grep -v '^--' > scripted_bad_reads.txt

# to exit nano ctrl + X > then Y for yes > then enter to close

# look at permissions

ls -l bad-reads-script.sh

# change permissions

chmod +x bad-reads-script.sh

# now: -rwxr-xr-x 1 dcuser dcuser 0 Oct 25 21:46 bad-reads-script.sh
```

Day 3 - Unix shell continued / Data Wrangling

Sign-in

- Sarah Stevens (she/her/hers)
- Heather Shimon (she/her) Science & Engineering Libraries
- Trisha Adamus (she/her), Ebling Library
- Emilie Greene (she/her)
- Celeste Huff (she/her) - Dept. Plant Pathology and Entomology
- Erwin Lares - he - Research Cyberinfrastructure
- Hailey Hessler (she/her)
- Rachel Kirchner (she/her), Dept of Medicine
- Madeleine Lodes (she/her), Bacteriology
- Anita Bhattacharyya
- Balendra (He/him)
- Avelina (She/her)
- Adel Talaat, SVM
- Lisa Abler (she/her)
- Lukas Matthews (he/him)

Feedback Review:

Things you learned/found useful

- unix shell commands
- searching history
- read quality
- loops
- the curriculum notes x2
- learning about fastq files x2
- the up arrow to get back previous commands
- viewing contents of files
- code along model
- helpers answering questions

Things you found confusing

- Scripts were challenging
    - Scripts are a challenging concept and are super useful! Everything we've done so far we entered in the console, while it is recorded in our history - that history can disappear and then we would have no record of what we did. It is also not reuseable on another computer - unless we move the commands into a script. Then we can rerun it whenever we need to and make it more flexible (with variables) to run on different files/samples.
- nano x4
    - Nano is a text editor. It is one that opens up in our commandline directly. There are others that open

up in graphical interfaces (vscode, bbedit, notepad++, etc) but since we are using this "headless" remote computer, if it popped open, we wouldn't be able to see it. In-line text editors like vi(m), emacs, and nano allow us to write/edit files directly on the remote machine directly in the command line. Nano is the simplest of the cmdline text editors, it doesn't have as many features but it gives some hints about how to use it on the screen.
    - We used nano to write a README file and to write a script that contained a command we wanted to be able to rerun.
- just playing around with commands? how do they fit together?
    - We are learning the foundational commands for the unix shell, and how to navigate around the directories/folders in this first lesson. In the next lesson, today and tomorrow, we will work with bioinformatics software to create a variant calling pipeline and build upon the skills we learned in a realistic example.
- grep x2
    - Grep is great for searching for patterns in files. This can be useful if you want to quickly check found for paritucIar motifs, find specific lines or the inverse of those specific lines, and other text manipulations.
- less
    - less is "pager", aka text file viewer. It allows us to look at a text file without editing it. Unlike cat it only shows a portion of the file at a time and then closes it when we exit less. Unlike head and tail, we can page through the file instead of seeing only the top or bottom.

Notes

# Accessing AWS instances
1. Open Git Bash (windows) or terminal (Mac)
2. ssh dcuser@(copyAndPasteYourInstance), e.g., ssh dcuser@ec2-3-236-71-133.compute-1.amazonaws.com

- - Your instance will start with ec2 and end with amazonaws.com. Grab the instance from here: https://pad.carpentries.org/2024-08-05-uwmadison-dc-instances

3. Enter in your password (same for all of us): data4Carp

```
# change prompt display if you'd like
PS1='$ '

# for loops
# structure all start with for
# then provide the variable name
# then in

for num in 1 2 3 4
> do
> echo the number this time is $num
> done

# output:
the number this time is 1
the number this time is 2
the number this time is 3
the number this time is 4
```

```
# can also add { } around the variable
for num in 1 2 3 4; do echo the number this time is ${num}; done

# for num in 1 2 3 4; do echo the number this time is $num_2024; done      # doesn't work because there is no variable called num_2024

# add { } around the variable name to add _2024 at the end of the output
for num in 1 2 3 4; do echo the number this time is ${num}_2024; done

# output:
the number this time is 1_2024
the number this time is 2_2024
the number this time is 3_2024
the number this time

# for loop can change a bunch of files at one time

# basename command to remove extra characters
# basename [name of file] [what to remove]

basename myfavfile.txt .txt
# output is myfavfile

basename 1_2024 _2024
# output is 1

# another example
$ foo=myfavfile.txt
$ echo
$ echo $foo
myfavfile.txt
$ echo ${foo%.txt}
myfavfile

# help with commands: https://devhints.io/bash

# cd to shell_data

cd shell_data

# cd to untrimmed_fastq

cd untrimmed_fastq

# create/open nano file called READme.txt
# note that letter case matters

nano READme.txt
```

```
# add text
This is an example project to learn the unix shell
It was created by [name]
email: [your email]
date: 2024-08-07

# to exit > ctrl + X > y > enter to save and exit

# resources for making a readme: https://www.makeareadme.com/

# cat to see what is in the file

cat READme.txt

# see what is in the bad-reads-scripts

cat bad-reads-scripts.sh

# output
grep -B1 -A2 -h NNNNNNNNNN *.fastq | grep -v '^--' > scripted_bad_reads.txt

# open nano bad-reads-scripts.sh

nano bad-reads-scripts.sh

# enter additional text

mv scripted_bad_reads.txt scripted_bad_reads.fastq

$ cat bad-reads-scripts.sh
grep -B1 -A2 -h NNNNNNNNNN *.fastq | grep -v '^--' > scripted_bad_reads.txt
mv scripted_bad_reads.txt scripted_bad_reads.fastq

# change permissions to bad-reads-scripts.sh

bash bad-reads-scripts.sh

# look at bad-reads-scripts.sh again

cat bad-reads-scripts.sh

# output
grep -B1 -A2 -h NNNNNNNNNN *.fastq | grep -v '^--' > scripted_bad_reads.txt
mv scripted_bad_reads.txt scripted_bad_reads.fastq

# transferring data from computers with curl or wget

# check on what software is accessible on the machine with which
```

```
which curl
which gwet

# we have both installed it and it prints back the address of where the program is installed
# if not installed the output will be empty

# to get a word count of how many lines in a file with wc and -l flag

wc -l species_EnsemblBacteria.txt

# output
44049 species_EnsemblBacteria.txt

# using a pipe | to count files

ls *.fastq | wc -l

# output is 3

# comment with #
# I can write whatever I want and it will not run it as a command

# scp file_I_want_to_move where_I_want_to_move_it

ls   # to see files

# confirm file exists and get full file structure using tab complete to make sure it is there

ls ~/shell_data/untrimmed_fastq/scripted_bad_reads.fastq

# output
/home/dcuser/shell_data/untrimmed_fastq/scripted_bad_reads.fastq

exit  # to return to your computer

# cd to Desktop if you'd like to put the file there

scp dcuser@(your instance):/home/dcuser/shell_data/untrimmed_fastq/scripted_bad_reads.fastq .

# enter

# password data4Carp

ls    # to see file on local computer

# transfer file from local computer to AWS instance

# touch will create a pretend file
```

touch myfavfile.txt

scp myfavfile.txt dcuser@PASTEYOURINSTANCE:~/shell_data/untrimmed_fastq

# return to AWS instance
# cd to shell_data/untrimmed_fastq/

cd shell_data/untrimmed_fastq/

# confirm that the scripted_bad_reads.fastq and myfavfile.txt exist

ls

# if you are missing .fastq files and need to move them

```
for filename in
*.fastq
> do
> name= $(basename ${filename} _2024.txt)
> mv $filename ${name}.fastq
> done
```

# data wrangling
# project organization first

# make folders

pwd

# move to cd to return to home directory /home/dcuser

EXERCISE

in /home/dcuser directory, use the mkdir command to make the following directories:

- dc_workshop
- dc_workshop/docs
- dc_workshop/data
- dc_workshop/results

# Solution

mkdir dc_workshop

# cd into dc_workshop

# now that we are in dc_workshop make more than one directory at a time

```
mkdir docs data results scripts
```

# good example of a project folder, might also want to have a folder for reports and figures

# check the folders

```
ls -R
```

# output
```
.:
data  docs  results  scripts

./data:

./docs:

./results:

./scripts:
```

# Bioinformatic workflows: https://datacarpentry.org/wrangling-genomics/02-quality-control.html#bioinformatic-workflows

# Sequence reads > Quality control > Alignment to genome > Alignment cleanup (BAM ready for variant calling) > Variant calling

1. Quality control - Assessing quality using FastQC
2. Quality control - Trimming and/or filtering reads (if necessary)
3. Align reads to reference genome
4. Perform post-alignment clean-up
5. Variant calling

# there are many tools for the different steps, check what tools your lab/collaborators are using
# this is practice of stitching together different tools in a workflow

# need folder for raw files

```
mkdir data/untrimmed_fastq

cd data/untrimmed_fastq/

pwd
```

# output
```
/home/dcuser/dc_workshop/data/untrimmed_fastq
```

# upload file

```
curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR258/004/SRR2589044/SRR2589044_1.fastq.gz

# it's a big file!

ls    # to see file SRR2589044_1.fastq.gz

# copy the files
cp ~/.backup/untrimmed_fastq/*.fastq.gz .

ls

# output
SRR2584863_1.fastq.gz  SRR2584863_2.fastq.gz  SRR2584866_1.fastq.gz  SRR2584866_2.fastq.gz
SRR2589044_1.fastq.gz  SRR2589044_2.fastq.gz

# Note we will be calling the files by the last 2 numbers

# gz ending indicates that they are compressed or zipped files

# can see the sizes of the files with -lh

ls -lh

# output
total 1.2G
-rw-rw-r-- 1 dcuser dcuser 175M Aug  7 16:00 SRR2584863_1.fastq.gz
-rw-rw-r-- 1 dcuser dcuser 183M Aug  7 16:00 SRR2584863_2.fastq.gz
-rw-rw-r-- 1 dcuser dcuser 309M Aug  7 16:00 SRR2584866_1.fastq.gz
-rw-rw-r-- 1 dcuser dcuser 296M Aug  7 16:00 SRR2584866_2.fastq.gz
-rw-rw-r-- 1 dcuser dcuser 124M Aug  7 16:00 SRR2589044_1.fastq.gz
-rw-rw-r-- 1 dcuser dcuser 128M Aug  7 16:00 SRR2589044_2.fastq.gz

# Note forward reads will be 1 and reverse reads will be 2

# unzip a file and see how big they get

gunzip SRR2584863_1.fastq.gz

# check file

ls -lh

# output:
total 1.6G
-rw-rw-r-- 1 dcuser dcuser 545M Aug  7 16:00 SRR2584863_1.fastq
-rw-rw-r-- 1 dcuser dcuser 183M Aug  7 16:00 SRR2584863_2.fastq.gz
-rw-rw-r-- 1 dcuser dcuser 309M Aug  7 16:00 SRR2584866_1.fastq.gz
-rw-rw-r-- 1 dcuser dcuser 296M Aug  7 16:00 SRR2584866_2.fastq.gz
-rw-rw-r-- 1 dcuser dcuser 124M Aug  7 16:00 SRR2589044_1.fastq.gz
```

```
-rw-rw-r-- 1 dcuser dcuser 128M Aug  7 16:00 SRR2589044_2.fastq.gz

# Note some tools will require unzipping files, but some can use zipped files

# check first four lines of the file

ls



# output
@SRR2584863.1 HWI-ST957:244:H73TDADXX:1:1101:4712:2181/1
TTCACATCCTGACCATTCAGTTGAGCAAAATAGTTCTTCAGTGCCTGTTTAACCGAGTCACGC
AGGGGTTTTTGGGTTACCTGATCCTGAGAGTTAACGGTAGAAACGGTCAGTACGTCAGAATT
TACGCGTTGTTCGAACATAGTTCTG
+
CCCFFFFFGHHHHJIJJJJIJJJIIJJJJIIIJJGFIIIJEDDFEGGJIFHHJIJJDECCGGEGIIJFHFFFACD:BBBD
DACCCCAA@@CA@C>C3>@5(8&>C:9?8+89<4(:83825C(:A#########################
```

EXERCISE

What is the last read in the SRR2584863_1.fastq file? How confident are you in this read?

```
# solution
tail -4 SRR2584863_1.fastq

# better quality at the beginning, gets worse, but not as bad as the first one that we looked at

# pipeline
# run the software in the terminal, and then move it to a script to save it

# write first script notes
# qulity control script and variant calling script
# want scripts in the scripts folder

pwd

# output
/home/dcuser/dc_workshop/data/untrimmed_fastq

# need to go 2 folders up

nano ../../scripts/read_qc.sh

# in nano create outline

set -e
cd ~/dc_workshop/data/untrimmed_fastq/
```

# Run the quality assessment

# make a folder for the output files and move the resulting QA files there

# put together all the QA results into one files

# run the trimming software

# Run the QA again

## exit with ctrl + X > Y > enter

# see the file with cat

```
cat ../../scripts/read_qc.sh
```

# pwd to make sure you are in the correct directory: /home/dcuser/dc_workshop/data/untrimmed_fastq

# run quality assessment with fastqc software

# get help

```
fastqc -h
```

```
SYNOPSIS
        fastqc seqfile1 seqfile2 .. seqfileN
    fastqc [-o output dir] [--(no)extract] [-f fastq|bam|sam]
           [-c contaminant file] seqfile1 .. seqfileN
```

# we will run it with the default
# run it on 1 to start with

```
 fastqc SRR2584863_1.fastq
```

# will start analysis on the file and print every 5% that it gets through

```
ls
```

# new outputs
```
SRR2584863_1_fastqc.html
SRR2584863_1_fastqc.zip
```

# run it on all of the files
# will run on compressed files as well, so adding * after fastq to include fastq.gz files

```
fastqc *.fastq*
```

# running analysis on all files in the folder

```
# looking at the quality scores of the reads and giving us a view of them - not changing the files

ls

# now have .html and .zip files for all of the reads
# document in script

nano ../../scripts/read_qc.sh

# update script to:

set -e
cd ~/dc_workshop/data/untrimmed_fastq/

# Run the quality assessment
echo "Running FastQC..."
fastqc *.fastq*

# make a folder for the output files and move the resulting QA files there

# put together all the QA results into one files

# run the trimming software

# Run the QA again

ls

# have output of fastq files and the qc files
# lets move qc folders
# make directory to move them to

 mkdir ~/dc_workshop/results/fastqc_untrimmed_reads

# move all the .zip and .html files to new directory

mv *.zip *.html ~/dc_workshop/results/fastqc_untrimmed_reads/

or could do it with *fastqc*

mv *fastqc* ~/dc_workshop/results/fastqc_untrimmed_reads/

ls

ls ~/dc_workshop/results/fastqc_untrimmed_reads/

# shows the .html and .zip files
# update script

nano ../../scripts/read_qc.sh

# add text

# make a folder for the output files and move the resulting QA file>mkdir
~/dc_workshop/results/fastqc_untrimmed_reads
mv *fastqc* ~/dc_workshop/results/fastqc_untrimmed_reads/

cd ../../results/fastqc_untrimmed_reads/

pwd

# output
/home/dcuser/dc_workshop/results/fastqc_untrimmed_reads

ls

# ouput
SRR2584863_1_fastqc.html  SRR2584866_2_fastqc.html
SRR2584863_1_fastqc.zip   SRR2584866_2_fastqc.zip
SRR2584863_2_fastqc.html  SRR2589044_1_fastqc.html
SRR2584863_2_fastqc.zip   SRR2589044_1_fastqc.zip
SRR2584866_1_fastqc.html  SRR2589044_2_fastqc.html
SRR2584866_1_fastqc.zip   SRR2589044_2_fastqc.zi

# unzip files
# unzip *.zip will only do one file at a time
# need a for loop

for filename in *.zip
> do
> unzip $filename
> done

ls -F

# now for every zip file, there is a folder

# output
fastqc_data.txt  fastqc_report.html  Images
fastqc.fo        Icons               summary.txt

# look at the summary file
cat SRR2584863_1_fastqc/summary.txt

# ouput summary is pass or warning
PASS    Basic Statistics        SRR2584863_1.fastq
PASS    Per base sequence quality       SRR2584863_1.fastq
PASS    Per tile sequence quality       SRR2584863_1.fastq
PASS    Per sequence quality scores     SRR2584863_1.fastq
WARN    Per base sequence content       SRR2584863_1.fastq
WARN    Per sequence GC content SRR2584863_1.fastq
PASS    Per base N content      SRR2584863_1.fastq
PASS    Sequence Length Distribution    SRR2584863_1.fastq
PASS    Sequence Duplication Levels     SRR2584863_1.fastq
PASS    Overrepresented sequences       SRR2584863_1.fastq
WARN    Adapter Content SRR2584863_1.fastq
```

# transfer file
# list full path so we can copy it

ls

# copy first .html file: SRR2584863_1_fastqc.html
# ls to get full file path name to use later on your local computer

ls ~/dc_workshop/results/fastqc_untrimmed_reads/SRR2584863_1_fastqc.html

# output
/home/dcuser/dc_workshop/results/fastqc_untrimmed_reads/SRR2584863_1_fastqc.html

# exit to return to local computer

# copy file to local computer using scp and the file path where the file is, and then where you want it to go (use a dot .)

scp dcuser@(your instance):/home/dcuser/dc_workshop/results/fastqc_untrimmed_reads/
SRR2584863_1_fastqc.html .

enter password - data4Carp

# look for SRR2584863_1_fastqc.html on your local computer directory to open file in a browser

Day 4 - Data Wrangling

Sign-in

- Sarah Stevens (she/her/hers)
- Anita
- Adel
- pwd
- Hailey Hessler
- Rachel Kirchner
- Heather Shimon (she/her) Science & Engineering Libraries

- Avelina (she/her)
- Balendra
- Madeleine
- Celeste
- Lisa (she/her)
- Emilie (she/her)

Feedback Review

Things you learned/found useful

- using nano
- navigating personal and remote computer
- practice using earlier concepts like mkdir, mv, for loops x2
- explanations of the tools/commands x2
- Downloading data x2
- scripts x2

Questions/confusions

- Is fastqc a standard program? or specific to the AWS instance?
    - fastqc is a bioinformatics program with an open license (so you can use it for free). It is installed on the AWS instance. If you try to run fastqc on your computer you will get an error that it isn't installed. You can install it on your computer, though installation is easier to understand once you are a bit familiar with navigating the unix shell and using commands in it, so we start with those skills. There are directions to help you install the tools in this workshop on your own computer in the setup page of the workshop (https://datacarpentry.org/genomics-workshop/#option-b-using-the-lessons-on-your-local-machine). If you want help installing it on a different computer, please come to Coding Meetup (https://hub.datascience.wisc.edu/consultation/).
- How can we automate more things? Can we automate data transfers?
    - You can automate data transfers! For downloading from a public site, you can put the wget/curl commands into a script that then downloads the data when you run it. This is a great way to organize your files and works well with formal version control (git), which we won't learn today but is really useful for keep track of different versions of your code/project files.
    - For transferring from you local computer to a server and vice versa, you will probably want to write the script to be run from your local computer since the remote server will more likely be setup to allow you to connect to it. Formal version control will also be helpful for keeping your code in sych in more than one location so you could have your project folder and scripts on your local computer and on the remote server and keep them sychronized. It is also possible to set up your computer to have ssh access but you will need security setup help from your departmental IT.
- Where is the software we are using located? (curl, fastqc, etc)
    - These tools are installed in the AWS instance. Some of them are utilities that come with the linux OS we are using (curl, ls, mkdir, etc) , others are bioinformatics software tools (fastqc, trimmomatic, samtools, etc) that were installed in this instance. There are setup

directions in the setup page of the curriculum webpage (https://datacarpentry.org/genomics-workshop/#option-b-using-the-lessons-on-your-local-machine) if you want to install it on your own computer. You can find where the software is installed using the `which` command, e.g. `which curl` will print out where the curl program is installed. I'd bet it is in the /usr/bin location we explored eariler as a lot of system utilities are installed there.

    - A little extra to know is that not all software/programs/scripts on a computer are available to run from anywhere on the computer. When you try to run a program, `myfavprogram input1 input2`, it will first look in your current folder for a program called myfavprogram, then it looks in what is called the "PATH" variable. The PATH variable is a list of locations on your computer where tools you want to be able to use anywhere are installed. You can also add locations of your own scripts or other tools you install to the PATH variable so you can run them from anywhere on the machine. This is a bit more abstract so it is one of the things that can make installation difficult for beginners. Once you are more familiar with navigating the unix shell and using variables, which we teach in this workshop, it is a bit easier to do.
- Filenames are a bit confusing - lots of fastq or fastqc
    - The fileanmes in this lesson are a bit confusing. They are somewhat representive of real filenames in a pipeline though. Often times researchers add on more info to the base sample name about how the data has been processed as they work through a pipeline. This is a good lesson though, think carefully about what you want to call your filenames. One I don't love about the names coming up is their use of "un", it looks like it is saying untrimmed but it is really saying unpaired.
- How do I get access to a bigger computer/computing cluster after the workshop?
    - We will talk more about this in the wrap up slides, there are a few options but one you might want to check out the Center for High Throughput Computing (CHTC). They have a computnig cluster that allows you to do a lot of computing. You do need to know the Unix shell to use their system, so you are on your way to being able to use it.
    - Center for High Throughput Computing: https://chtc.cs.wisc.edu/uw-research-computing/get-started

Notes

```
# trimming and filtering
# how to get rid of sequence data that doesn't meet our quality standards?
# will use Trimmomatic, lots of options
https://datacarpentry.org/wrangling-genomics/aio.html#trimmomatic-options

# example of Trimmomatic command
trimmomatic PE -threads 4 SRR_1056_1.fastq SRR_1056_2.fastq  \
        SRR_1056_1.trimmed.fastq SRR_1056_1un.trimmed.fastq \
        SRR_1056_2.trimmed.fastq SRR_1056_2un.trimmed.fastq \
        ILLUMINACLIP:SRR_adapters.fa SLIDINGWINDOW:4:20
```

code = meaning

PE = that it will be taking a paired end file as input
-threads 4 = to use four computing threads to run (this will speed up our run)
SRR_1056_1.fastq = the first input file name
SRR_1056_2.fastq = the second input file name
SRR_1056_1.trimmed.fastq = the output file for surviving pairs from the _1 file
SRR_1056_1un.trimmed.fastq = the output file for orphaned reads from the _1 file
SRR_1056_2.trimmed.fastq = the output file for surviving pairs from the _2 file
SRR_1056_2un.trimmed.fastq = the output file for orphaned reads from the _2 file
ILLUMINACLIP:SRR_adapters.fa = to clip the Illumina adapters from the input file using the adapter sequences listed in SRR_adapters.fa
SLIDINGWINDOW:4:20 = to use a sliding window of size 4 that will remove bases if their phred score is below 20

```
# change to directory

 cd ~/dc_workshop/data/untrimmed_fastq/

# copy file

cp ~/.miniconda3/pkgs/trimmomatic-0.38-0/share/trimmomatic-0.38-0/adapters/NexteraPE-PE.fa .

ls

# output
NexteraPE-PE.fa        SRR2584866_1.fastq.gz  SRR2589044_2.fastq.gz
SRR2584863_1.fastq     SRR2584866_2.fastq.gz
SRR2584863_2.fastq.gz  SRR2589044_1.fastq.gz

# trim the files
# can only run on one sample as a time
# for order, PE wants the forward read first
# So you do need them in order, which might make the wildcards not work (though I think with the alpha sorting of the name it will)

trimmomatic PE SRR2589044_1.fastq.gz SRR2589044_2.fastq.gz SRR2589044_1.trimmed.fastq
SRR2589044_1.un.trimmed.fastq SRR2589044_2.trimmed.fastq SRR2589044_2.un.trimmed.fastq
ILLUMINACLIP:NexteraPE-PE.fa:2:40:15 SLIDINGWINDOW:4:20 MINLEN:25

# Will take a few moments to run then says: TrimmomaticPE: Completed successfully

ls

# output
NexteraPE-PE.fa        SRR2589044_1.trimmed.fastq
SRR2584863_1.fastq     SRR2589044_1.un.trimmed.fastq
SRR2584863_2.fastq.gz  SRR2589044_2.fastq.gz
SRR2584866_1.fastq.gz  SRR2589044_2.trimmed.fastq
SRR2584866_2.fastq.gz  SRR2589044_2.un.trimmed.fastq
SRR2589044_1.fastq.gz
```

Exercise

- What percent of reads did we discard from our sample?

0.29%

- What percent of reads did we keep both pairs?

79.26%

```
ls SRR2589099*

# output
SRR2589044_1.fastq.gz          SRR2589044_2.fastq.gz
SRR2589044_1.trimmed.fastq     SRR2589044_2.trimmed.fastq
SRR2589044_1.un.trimmed.fastq  SRR2589044_2.un.trimmed.fastq

 ls -lah SRR2589044*

#output
-rw-rw-r-- 1 dcuser dcuser 124M Aug  7 16:00 SRR2589044_1.fastq.gz
-rw-rw-r-- 1 dcuser dcuser 295M Aug  8 14:44 SRR2589044_1.trimmed.fastq
-rw-rw-r-- 1 dcuser dcuser  55M Aug  8 14:44 SRR2589044_1.un.trimmed.fastq
-rw-rw-r-- 1 dcuser dcuser 128M Aug  7 16:00 SRR2589044_2.fastq.gz
-rw-rw-r-- 1 dcuser dcuser 268M Aug  8 14:44 SRR2589044_2.trimmed.fastq
-rw-rw-r-- 1 dcuser dcuser 596K Aug  8 14:44 SRR2589044_2.un.trimmed.fastq

# compress the file

gzip SRR2584863_1.fastq

ls

#output
NexteraPE-PE.fa        SRR2589044_1.trimmed.fastq
SRR2584863_1.fastq.gz  SRR2589044_1.un.trimmed.fastq
SRR2584863_2.fastq.gz  SRR2589044_2.fastq.gz
SRR2584866_1.fastq.gz  SRR2589044_2.trimmed.fastq
SRR2584866_2.fastq.gz  SRR2589044_2.un.trimmed.fastq
SRR2589044_1.fastq.gz

# make two loops
for infile in *_1.fastq.gz
do
  base=$(basename ${infile} _1.fastq.gz)
  trimmomatic PE ${infile} ${base}_2.fastq.gz \
        ${base}_1.trim.fastq.gz ${base}_1un.trim.fastq.gz \
        ${base}_2.trim.fastq.gz ${base}_2un.trim.fastq.gz \
        SLIDINGWINDOW:4:20 MINLEN:25 ILLUMINACLIP:NexteraPE-PE.fa:2:40:15
done
```

```
ls

# output
NexteraPE-PE.fa               SRR2584866_2un.trim.fastq.gz
SRR2584863_1.fastq.gz         SRR2589044_1.fastq.gz
SRR2584863_1.trim.fastq.gz    SRR2589044_1.trim.fastq.gz
SRR2584863_1un.trim.fastq.gz  SRR2589044_1.trimmed.fastq
SRR2584863_2.fastq.gz         SRR2589044_1un.trim.fastq.gz
SRR2584863_2.trim.fastq.gz    SRR2589044_1.un.trimmed.fastq
SRR2584863_2un.trim.fastq.gz  SRR2589044_2.fastq.gz
SRR2584866_1.fastq.gz         SRR2589044_2.trim.fastq.gz
SRR2584866_1.trim.fastq.gz    SRR2589044_2.trimmed.fastq
SRR2584866_1un.trim.fastq.gz  SRR2589044_2un.trim.fastq.gz
SRR2584866_2.fastq.gz         SRR2589044_2.un.trimmed.fastq
SRR2584866_2.trim.fastq.gz
```

# EXERCISE
We trimmed our fastq files with Nextera adapters, but there are other adapters that are commonly used. What other adapter files came with Trimmomatic?

```
ls ~/miniconda3/pkgs/trimmomatic-0.38-0/share/trimmomatic-0.38-0/adapters/

output#
NexteraPE-PE.fa  TruSeq2-SE.fa    TruSeq3-PE.fa
TruSeq2-PE.fa    TruSeq3-PE-2.fa  TruSeq3-SE.fa
```

```
# to remove trimmed files (if you'd like to, don't have to)

rm *trimmed*

ls

# output
NexteraPE-PE.fa               SRR2584866_2.fastq.gz
SRR2584863_1.fastq.gz         SRR2584866_2.trim.fastq.gz
SRR2584863_1.trim.fastq.gz    SRR2584866_2un.trim.fastq.gz
SRR2584863_1un.trim.fastq.gz  SRR2589044_1.fastq.gz
SRR2584863_2.fastq.gz         SRR2589044_1.trim.fastq.gz
SRR2584863_2.trim.fastq.gz    SRR2589044_1un.trim.fastq.gz
SRR2584863_2un.trim.fastq.gz  SRR2589044_2.fastq.gz
SRR2584866_1.fastq.gz         SRR2589044_2.trim.fastq.gz
SRR2584866_1.trim.fastq.gz    SRR2589044_2un.trim.fastq.gz
SRR2584866_1un.trim.fastq.gz
```

# alignment to a reference genome
# The alignment process consists of two steps:

1. Indexing the reference genome
2. Aligning the reads to the reference genome

```
cd ~/dc_workshop/

# make directory
mkdir -p data/ref_genome

curl -L -o data/ref_genome/ecoli_rel606.fasta.gz
ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/000/017/985/GCA_000017985.1_ASM1798v1/
GCA_000017985.1_ASM1798v1_genomic.fna.gz

ls data/ref_genome/ecoli_rel606.fasta.gz

# unzip the file
gunzip data/ref_genome/ecoli_rel606.fasta.gz

# output
data/ref_genome/ecoli_rel606.fasta.gz

# exercise
We saved this file as data/ref_genome/ecoli_rel606.fasta.gz and then decompressed it. What is the real
name of the genome?

# run
head data/ref_genome/ecoli_rel606.fasta

# solution
CP000819.1 Escherichia coli B str. REL606

curl -L -o sub.tar.gz https://ndownloader.figshare.com/files/14418248
tar xvf sub.tar.gz

# move to data folder
mv sub/ ~/dc_workshop/data/trimmed_fastq_small

ls data/trimmed_fastq_small/

# output
SRR2584863_1.trim.sub.fastq  SRR2584866_2.trim.sub.fastq
SRR2584863_2.trim.sub.fastq  SRR2589044_1.trim.sub.fastq
SRR2584866_1.trim.sub.fastq  SRR2589044_2.trim.sub.fastq

# make a results directory
mkdir -p results/sam results/bam results/bcf results/vcf

# index the reference genome with BWA

bwa index data/ref_genome/ecoli_rel606.fasta

# output
```

```
[bwa_index] Pack FASTA... 0.04 sec
[bwa_index] Construct BWT for the packed sequence...
[bwa_index] 1.46 seconds elapse.
[bwa_index] Update BWT... 0.03 sec
[bwa_index] Pack forward-only FASTA... 0.02 sec
[bwa_index] Construct SA from BWT and Occ... 0.58 sec
[main] Version: 0.7.17-r1188
[main] CMD: bwa index data/ref_genome/ecoli_rel606.fasta
[main] Real time: 2.219 sec; CPU: 2.147 sec

# aligning the reads from just one of the samples in our dataset (SRR2584866)
bwa mem data/ref_genome/ecoli_rel606.fasta data/trimmed_fastq_small/SRR2584866_1.trim.sub.fastq
data/trimmed_fastq_small/SRR2584866_2.trim.sub.fastq > results/sam/SRR2584866.aligned.sam

head -5 results/sam/SRR2584866.aligned.sam

# output
@SQ     SN:CP000819.1   LN:4629812
@PG     ID:bwa  PN:bwa  VN:0.7.17-r1188 CL:bwa mem data/ref_genome/ecoli_rel606.fasta
data/trimmed_fastq_small/SRR2584866_1.trim.sub.fastq
data/trimmed_fastq_small/SRR2584866_2.trim.sub.fastq
SRR2584866.7    83      CP000819.1      3585268 60      26M     =  3584643  -651
GCTTCCCATGCCAATTAATACATGTG      IJJJJJJJJJIJJHHHHHFEFFFCC@  NM:i:0
MD:Z:26 MC:Z:150M       AS:i:26 XS:i:0
SRR2584866.7    163     CP000819.1      3584643 60      150M    =  3585268  651
TCCTCAATCTTAAGCGGATCAATGAGCTGGTACGCCATCAGCATATTGATTATCTGGTGTGA
ATTTCAGGCTTACGGTGAGTCTGGCTACGCTGCCACACAGATTAGCTAATTGAAACGCCTTT
CACCCCTGCCATACCTTTTAATAATC
C@CFFFFFGHGGGGGJJJIJJJJIJJJIJIHIIJJJJJJJJJJJJEIIIJIJJJJJJIIJJJHHHHHHHFFFFFDDDDDDDD
DDDDDDDDBBDDDDDDDDDBDDDDDDDDEDDEDDDDDDDDBDDCDDDDDBDDBDDDDDDDD
DDDDEDED  NM:i:0  MD:Z:150   MC:Z:26M AS:i:150        XS:i:0
SRR2584866.8    99      CP000819.1      2861836 60      26M     =  2862333  647
GCTGATATTCTTGCAGCAGTACCGGC      @@@DBDDDHHFFHIFFGBH<FHCGDG  NM:i:0
MD:Z:26 MC:Z:150M       AS:i:26 XS:i:0

# use sam tools (or other tools) to look at information instead of looking at the sam files directly for easier
reading

# SAM/BAM format

samtools view -S -b results/sam/SRR2584866.aligned.sam > results/bam/SRR2584866.aligned.bam

ls results/bam/SRR2584866.aligned.bam

#output
results/bam/SRR2584866.aligned.bam

ls results/sam/SRR2584866.aligned.sam
```

```
#output
results/sam/SRR2584866.aligned.sam
```

# sort the BAM file using the sort command from samtools. -o tells the command where to write the output

```
samtools sort -o results/bam/SRR2584866.aligned.sorted.bam results/bam/SRR2584866.aligned.bam

ls -lah results/bam

#output
total 73M
drwxrwxr-x 2 dcuser dcuser 4.0K Aug  8 16:01 .
drwxrwxr-x 7 dcuser dcuser 4.0K Aug  8 15:43 ..
-rw-rw-r-- 1 dcuser dcuser  42M Aug  8 16:00 SRR2584866.aligned.bam
-rw-rw-r-- 1 dcuser dcuser  31M Aug  8 16:01 SRR2584866.aligned.sort

# learn more about the file
samtools flagstat results/bam/SRR2584866.aligned.sorted.bam

# output
351169 + 0 in total (QC-passed reads + QC-failed reads)
0 + 0 secondary
1169 + 0 supplementary
0 + 0 duplicates
351103 + 0 mapped (99.98% : N/A)
350000 + 0 paired in sequencing
175000 + 0 read1
175000 + 0 read2
346688 + 0 properly paired (99.05% : N/A)
349876 + 0 with itself and mate mapped
58 + 0 singletons (0.02% : N/A)
0 + 0 with mate mapped to a different chr
0 + 0 with mate mapped to a different chr (mapQ>=5)
```

# variant calling with bcftools: https://datacarpentry.org/wrangling-genomics/04-variant_calling.html#variant-calling
# need to to some prep work before calling the variants
# step 1: Calculate the read coverage of positions in the genome
# in bcftools, using the command mpileup.
# flag -O b tells bcftools to generate a bcf format output file, -o specifies where to write the output file, and -f flags the path to the reference genome

```
bcftools mpileup -O b -o results/bcf/SRR2584866_raw.bcf -f data/ref_genome/ecoli_rel606.fasta results/bam/SRR2584866.aligned.sorted.bam

# output
[mpileup] 1 samples in 1 input files
```

```
# if you need to rename files
mv results/bam/SRR2584866.sorted.bam results/bam/SRR2584866.aligned.sorted.bam

# Step 2: Detect the single nucleotide variants (SNVs)
# Identify SNVs using bcftools call
```

# specify ploidy with the flag --ploidy, which is one for the haploid E. coli. -m allows for multiallelic and rare-variant calling, -v tells the program to output variant sites only (not every site in the genome), and -o specifies where to write the output file:

```
bcftools call --ploidy 1 -m -v -o results/vcf/SRR2584866_variants.vcf results/bcf/SRR2584866_raw.bcf

# Step 3: Filter and report the SNV variants in variant calling format (VCF)
# Filter the SNVs for the final output in VCF format, using vcfutils.pl

vcfutils.pl varFilter results/vcf/SRR2584866_variants.vcf  > results/vcf/SRR2584866_final_variants.vcf

# filtering
```

# The vcfutils.pl varFilter call filters out variants that do not meet minimum quality default criteria, which can be changed through its options. Using bcftools we can verify that the quality of the variant call set has improved after this filtering step by calculating the ratio of transitions(TS) to transversions (TV) ratio (TS/TV), where transitions should be more likely to occur than transversions:

```
bcftools stats results/vcf/SRR2584866_variants.vcf | grep TSTV

# output
# TSTV, transitions/transversions:
# TSTV  [2]id   [3]ts   [4]tv   [5]ts/tv       [6]ts (1st ALT) [7]tv (1st ALT)     [8]ts/tv (1st ALT)
TSTV    0       628     58      10.83   628     58      10.83

bcftools stats results/vcf/SRR2584866_final_variants.vcf | grep TSTV

# output
# TSTV, transitions/transversions:
# TSTV  [2]id   [3]ts   [4]tv   [5]ts/tv       [6]ts (1st ALT) [7]tv (1st ALT)     [8]ts/tv (1st ALT)
TSTV    0       621     54      11.50   621     54      11.50
```

# Explore the VCF format: https://datacarpentry.org/wrangling-genomics/04-variant_calling.html#explore-the-vcf-format

```
less -S results/vcf/SRR2584866_final_variants.vcf

# Assess the alignment (visualization)
#  to visualize the alignment files, we will need to index the BAM file using samtools:

samtools index results/bam/SRR2584866.aligned.sorted.bam

# Viewing with tview

samtools tview results/bam/SRR2584866.aligned.sorted.bam data/ref_genome/ecoli_rel606.fasta
```

# For viewer the . Means it matches in the forward strand, the , means it matches in the revers
# The first line of output shows the genome coordinates in our reference genome.
# The second line shows the reference genome sequence.
# The third line shows the consensus sequence determined from the sequence reads. A . indicates a match to the reference sequence, so we can see that the consensus from our sample matches the reference in most locations. # That is good! If that was not the case, we should probably reconsider our choice of reference.

# Below the horizontal line, we can see all of the reads in our sample aligned with the reference genome. Only positions where the called base differs from the reference are shown.
# You can use the arrow keys on your keyboard to scroll or type ? for a help menu.
# To navigate to a specific position, type g. A dialogue box will appear. In this box, type the name of the "chromosome" followed by a colon and the position of the variant you would like to view (e.g. for this sample, type CP000819.1:50 to view the 50th base.
# Type Ctrl^C or q to exit tview.

# Automating a Variant Calling Workflow
# make it more efficient and minimize errors

# previous script:
# grep -B1 -A2 NNNNNNNNNN *.fastq > scripted_bad_reads.txt
# echo "Script finished!"

# script we made for trimmomatic for loop

```
for infile in *_1.fastq.gz
> do
>   base=$(basename ${infile} _1.fastq.gz)
>   trimmomatic PE ${infile} ${base}_2.fastq.gz \
>               ${base}_1.trim.fastq.gz ${base}_1un.trim.fastq.gz \
>               ${base}_2.trim.fastq.gz ${base}_2un.trim.fastq.gz \
>               SLIDINGWINDOW:4:20 MINLEN:25 ILLUMINACLIP:NexteraPE-PE.fa:2:40:15
> done
```

# cd to scripts folder

```
cd ~/dc_workshop/scripts

nano read_qc.sh
```

# add text and comment out earlier commands (temporarily for testing, you will want to go uncomment the lines later so the script will run the full pipeline)
# full nano text

```
set -e
cd ~/dc_workshop/data/untrimmed_fastq/

# Run quality assessment
echo "Running FastQC..."
#fastqc *.fastq*
```

```
# make a folder for the output files and move the result QA files there
#mkdir ~/dc_workshop/results/fastqc_untrimmed_reads
#mv *fastqc* ~/dc_workshop/results/fastqc_untrimmed_reads

# put together all the QA results into one file

# run trimming software

for infile in *_1.fastq.gz
do
	base=$(basename ${infile} _1.fastq.gz)
	trimmomatic PE ${infile} ${base}_2.fastq.gz \
		${base}_1.trim.fastq.gz ${base}_1un.trim.fastq.gz \
		${base}_2.trim.fastq.gz ${base}_2un.trim.fastq.gz \
		SLIDINGWINDOW:4:20 MINLEN:25 ILLUMINACLIP:NexteraPE-PE.fa:2:40:15

done

# Run QA again

# exit nano:  ctrl + X  >  y  > enter
```

# Automating the rest of our variant calling workflow: https://datacarpentry.org/wrangling-genomics/05-automation.html#automating-the-rest-of-our-variant-calling-workflow

```
curl -O https://datacarpentry.org/wrangling-genomics/files/run_variant_calling.sh

ls -la

#output
total 16
drwxrwxr-x 2 dcuser dcuser 4096 Aug  8 17:24 .
drwxrwxr-x 6 dcuser dcuser 4096 Aug  8 15:42 ..
-rw-rw-r-- 1 dcuser dcuser  707 Aug  8 17:22 read_qc.sh
-rw-rw-r-- 1 dcuser dcuser 1158 Aug  8 17:24 run_variant_calling.sh
```

# Our variant calling workflow has the following steps:

1. Index the reference genome for use by bwa and samtools.
2. Align reads to reference genome.
3. Convert the format of the alignment to sorted BAM, with some intermediate steps.
4. Calculate the read coverage of positions in the genome.
5. Detect the single nucleotide variants (SNVs).
6. Filter and report the SNVs in VCF (variant calling format).

# go through the script

```
cd ~/dc_workshops/scripts

# script should look like this

set -e
cd ~/dc_workshop/results

genome=~/dc_workshop/data/ref_genome/ecoli_rel606.fasta

bwa index $genome

mkdir -p sam bam bcf vcf

for fq1 in ~/dc_workshop/data/trimmed_fastq_small/*_1.trim.sub.fastq
    do
    echo "working with file $fq1"

    base=$(basename $fq1 _1.trim.sub.fastq)
    echo "base name is $base"

    fq1=~/dc_workshop/data/trimmed_fastq_small/${base}_1.trim.sub.fastq
    fq2=~/dc_workshop/data/trimmed_fastq_small/${base}_2.trim.sub.fastq
    sam=~/dc_workshop/results/sam/${base}.aligned.sam
    bam=~/dc_workshop/results/bam/${base}.aligned.bam
    sorted_bam=~/dc_workshop/results/bam/${base}.aligned.sorted.bam
    raw_bcf=~/dc_workshop/results/bcf/${base}_raw.bcf
    variants=~/dc_workshop/results/bcf/${base}_variants.vcf
    final_variants=~/dc_workshop/results/vcf/${base}_final_variants.vcf

    bwa mem $genome $fq1 $fq2 > $sam
    samtools view -S -b $sam > $bam
    samtools sort -o $sorted_bam $bam
    samtools index $sorted_bam
    bcftools mpileup -O b -o $raw_bcf -f $genome $sorted_bam
    bcftools call --ploidy 1 -m -v -o $variants $raw_bcf
    vcfutils.pl varFilter $variants > $final_variants

    done

# q to quit less screen

# open nano

nano run_variant_calling.sh

# comment the script with #
# put the comment above the code or on the same line

# exit nano:  ctrl + X  >  y  > enter
# run script
nano run_variant_calling.sh

## Script with everyone's comments

#
set -e

#switches to the results directory
cd ~/dc_workshop/results

#variable forlocation of reference genome
genome=~/dc_workshop/data/ref_genome/ecoli_rel606.fasta

#indexed reference genome for BWA
bwa index $genome

#create directories
mkdir -p sam bam bcf vcf

#Run variant calling workflow for each of the fastQ files and prints to the screen

for fq1 in ~/dc_workshop/data/trimmed_fastq_small/*_1.trim.sub.fastq
   do
   echo "working with file $fq1"

#Pulls the sample name from each of the forward read files
   base=$(basename $fq1 _1.trim.sub.fastq)
   echo "base name is $base"

#creates variables for the output files
   fq1=~/dc_workshop/data/trimmed_fastq_small/${base}_1.trim.sub.fastq
   fq2=~/dc_workshop/data/trimmed_fastq_small/${base}_2.trim.sub.fastq
   sam=~/dc_workshop/results/sam/${base}.aligned.sam
   bam=~/dc_workshop/results/bam/${base}.aligned.bam
   sorted_bam=~/dc_workshop/results/bam/${base}.aligned.sorted.bam
   raw_bcf=~/dc_workshop/results/bcf/${base}_raw.bcf
   variants=~/dc_workshop/results/bcf/${base}_variants.vcf
   final_variants=~/dc_workshop/results/vcf/${base}_final_variants.vcf

#1. align the reads to the reference genome and output a .sam file:
   bwa mem $genome $fq1 $fq2 > $sam
#2. convert the SAM file to BAM format:
   samtools view -S -b $sam > $bam
#3. sort the BAM file
   samtools sort -o $sorted_bam $bam
#4. index the BAM file for display+functional purposes:
   samtools index $sorted_bam
```

```
#5. calculate the read coverage of positions in the genome:
    bcftools mpileup -O b -o $raw_bcf -f $genome $sorted_bam
#6. call SNVs with bcftools:
    bcftools call --ploidy 1 -m -v -o $variants $raw_bcf
#7. filter and report the SNVs in variant calling format (VCF)(for quality)
    vcfutils.pl varFilter $variants > $final_variants

    done
```

Day 5 - Intro R

Sign-in:

- Sarah Stevens (she/her/hers)
- Hailey Hessler
- Heather Shimon (she/her) Science & Engineering Libraries
- Zekai Otles (He/him)- DoIT Research Cyberinfrastructure Consultant
- Anita Bhattacharyya
- Madeleine Lodes (she/her)
- Emilie Greene (she/her)
- Rachel Kirchner (she,her) Dept of Medicine
- Balendra Sah (He/him)
- Annika Pratt (she/her) - Helper, Plant Pathology graduate student
- Celeste Huff (she/her)- Dept Plant Pathology and Entomology
- Lukas matthews (he/him)
- Avelina Gaston (she/her)
- Grace Cagle - Soil Sciences
- Kaustubh Amritkar
- Lisa Abler (she/her)

Feedback Review

Things you learned / found useful:

- That we used real data and biological Questions
- Using a real workflow
- Using nano x2
- scripts with comments x3
- explainations of the tools
- incremental knowledge building of the workshop
- practicing for loops
- using trimmomatic

Questions/Concerns

- Will we have access to the curriciulum after the course?
    - Yes! All the lesson materials are openly available on the web and will remain so after the workshop.
- How will I use this info after the workshop?
    - You may want to search for related tools that will help you answer the questions you are interested in for your analysis. You can come back and review the lesson materials as needed, or get help at Coding Meetup as you work to implement them in your work.
- What do some of the -letter options mean? e.g. samtools view -S -b results
    - - The -letter (often called options or flags) can be looked up in the manual. Here is the view manual page (https://www.htslib.org/doc/samtools-view.html).
    - The -S is outdated in the recent version of sam tools but used to specify that the input to the command is in sam format. Now it automatically detects the input format.
    - - The -b specifies that the output should be in bam format
- Using the path variable when we do our own installations
    - Come to Coding Meetup for help! This is a helpful summary (https://janelbrandon.medium.com/understanding-the-path-variable-6eae0936e976)
- meaning of bash command and the -la option
    - the bash command runs a bash script so `bash myfavscript.sh` runs the myfavscript.sh script
    - I'm guessing you are asking about the -la options for the ls command. -a says show me all the files, including those that are hidden (aka their names start with a .) -l says show me the long form list that includes more info like the date modified, permissions, size of files, etc.
- Syntax of bash vs R
    - Some useful info here - https://www.r-bloggers.com/2022/06/shell-vs-r-fundamentals-from-syntax-to-control-structures-with-zsh-bash/
- List of bioinformatics tools?
    - Wish there was one. I'll see if I can find a few that might be helpful and add them here. Folks are always making new ones so it can be hard to keep up a single list.
    - https://en.wikipedia.org/wiki/List_of_bioinformatics_software
    - https://biocontainers.pro/registry - this is a containers registry - containers are a way to make software portable (easy to install) across lots of computers. This registry is a nice place to find tools that are already packaged in containers. Take our Docker workshop if you want to learn more about portable software and learn to use containers.
    - Lots of stuff you can do with bioconductor too, which is a set of R packages for doing bioinformatics (https://www.bioconductor.org/)
- Confused on terminology - Indexing, reference genome, bwa
    - Indexing helps sort the file so it can be searched faster
    - reference genome - this is a known genome that is sequenced and you are using in your analysis. In this case it is the starting strain from the experiement, so all the mutations we find in the sequcing came from the experiment.
    - bwa - burrows-wheeler aligment tool, this is the name of the tool that aligns/maps the sequenced reads to the reference genome
- Are all the flags/options in the manuals? x2
    - Typically they are, see the above example of samtools. There was also a little confusing about where the flags go. They go between the name of the command and the arguments. This diagram of the anatomy of a unix shell command image may help (https://swcarpentry.github.io/shell-novice/fig/shell_command_syntax.svg)
- Is there a simple way to map how data needs to be converted, modified, relocated?
    - I'm not quite sure quite what this question is asking, let me know if you have follow-up

questions on my answer or I didn't quite understand it.

- You can find the map of the tools we are using in this pipeline in the bioinformatics workflows section of the lesson (https://datacarpentry.org/wrangling-genomics/02-quality-control.html#bioinformatic-workflows). The way the data needs to be maniuplated really depends on the pipeline you are running. Often times data is cleaned and then analyzed/manipulated into something like a tabular form for further analaysis. Then you may make tables and figures and stasticial analyses to answer the questions for your research. Location may be thinking careful about project structure so your project is organized for others and yourself in the future.

Notes

Login to AWS through browser instead of terminal today
http://xxxinstancename:8787
dcuser
data4Carp
don't worry about security notice (do not work on secure data here)
Tools > Global Options > Code > Soft-wrap R source files (makes it easier to see text)

R vs Python?
It's up to you! Ask your lab mates/collabprators what they use
Today: getting R setup and some data wrangling
R/RStudio

- R - programming language
- RStudio - interface (what makes R easier to understand and use!)
- install both if using on your own machine instead of AWS

You will make mistakes. We all still make mistakes (even after years and years of practice). Making mistakes is the best way to learn!

Upper right: Project(none)
We want to set it up so there is a project - good data management practice
Can move from project to project in upper right had corner dropdown

File > New Project > New Directory > New Project
Directory name: dc_genomics_r
leave tilda (~) in Create project as subdirectory of:
Create Project
Now you should see your project name in upper right hand corner

File > New File > R Script
Click on save icon
File name: 01_intro_r
Click on Home > dc_workshop > scripts
Save

Good practice: keep all scripts together
R does not automatically save!

top left pane: our script
lower left pane: console - we can type code here that will be run once and not saved

- broom icon in upper righ tof console - clears the text
- when we run code in our script, it will be outputted in the console

upper right panel: environment

- objects will show up there

lower right pane: files, plots, packages, help

- plots will show up there tomorrow
- can navigate filesystem
- can lookup help with specific commands
- We have a lot of packages installed already - thanks AWS!
    - If you use R + RStudio on your own machine (not AWS instance) you will have to install packeges yourself

Packages need to be installed once on your system, and loaded every time you use RStudio

R project saves your whole environment (you don't need to re-run your scripts)
What Heather usually does: save all your scripts and data in your RProject so it is all together. We're doing it a little different today, but setting things up like Heather is a great idea for your research projects!
The "RProject" is basically a folder. R scripts are basically just text files (you can edit them in any text editor).

Use # to comment in R

```
# author: Your Name
# email: me@wisc.edu
# date: 2024-08-12
# description: Learning to work with R

# functions
# getwd() similar to pwd in Unix Shell
# parentheses signify function
getwd()

# hit ctrl + enter to run line
# can also hit run button (green arrow in upper right hand corner of script pane)
# output shows up in console

# do not use setwd() - set up a Project instead

# in the console
3 + 5
# click enter to run
```

7/2

```r
# click enter to run
# can put spaces between numbers and math symbols, but it isn't necessary

# click save often when working on your R script
# the filename will be red with a * if it hasn't been saved in the current form

# creating objects
# give us a chunk of information to analyze
# variable name <- value
# alt - (Windows) option - (Mac) is the keyboard shortcut for <-
a <- 1
# type control enter or hit the run button to assign the value 1 to the object a
# an object in this case is basically a variable in algebra

# To check all the shortcuts in RStudio, you can click on the Help `Menu` and then `Keyboard Shortcuts Help`

# object with gene name
# use quotations for string - single or double quotes will work as long as you are consitent
gene_name <- 'tp53'
# type control enter to run the line
# do not have spaces or dashes in your object name

# remove object with rm()
rm(gene_name)
# type control enter to run that line of code
# your object should now dissappear from the environment pane (upper right)

# object types / data types
# find the object type mode()
mode(a)
# ctrl enter

# data types: numeric (number), logical (T/F), character (string)

# EXERCISE

# Create the following objects in R
# Then use the mode() function to verify their modes
# Try to guess what the mode will be before you look at the solution

chromosome_name <- 'chr02'
mode(chromosome_name) # do not need quotes around object name

od_600_value <- 0.47
mode(od_600_value)

chr_position <- '1001701'
```

mode(chr_position) # since we used quotes, it will be saved as a character instead of a number
as.numeric(chr_position) # changes the string from a character to a number

spock <- TRUE   # TRUE must be in all caps
mode(spock)

pilot <- Earhart # doesn't work
pilot <- "Earhart" # works
mode(pilot)

# can do math with objects if they are numeric
od_600_value / 3

# can't do math on characters
pilot + 2

BREAK
Come back at 10:27 CT

# help in RStudio
?function
ex ?round

# help will show up in lower right pane
# everything in parentheses are arguments
# arguments are separated by commas

# will divide our od_600_value by 3, then round that ouput to 2 decimal places
# R has a default order for arguments, you don't need to use the equals sign because R knows the first argument is x and the second argument is digits. If someone else is going to be reading your code, it's nice to add the variable names (ex. digits) so your code is more human readable
#You have options: you can follow the order and just pass arguments without names, you can ignored the order and name the arguments, OR you can mix and match

round(od_600_value / 3, digits = 2)

# tells you the arguments of the round funciton
args(round)

Click File > New FIle
Click the save icon
Make sure you are still in your scripts folder
File name: 02_dplyr
Save

Click broom icon in upper right panel to clear environment
Click transparent broom in lower left panel to clear the console

In our new script:

```
# name: Cool Name
# email imthebest@wisc.edu
# date: 2024-08-12
# description: learning about dplyr

# in help, you can search for the name of a function - just like doing ?function
# you can also look on google

# load packages
library(dplyr)
# ctrl enter to run

library(tidyr)
# ctrl enter

library(ggplot2)
# ctrl enter

library(readr)
# ctrl enter

# you should load all your packages at the top of your script
# must load packages each time you open your script
#There is an umbrella package called “tidyverse” that includes those packages and a few others

#The default repo for all R packages is called CRAN (https://cran.r-project.org/)

# read in the dataset with read_csv() - more tidy than read.csv()
# save as an object called variants
# using relative paths (second option) is more reproducible than absolute paths
variants <- read_csv("/home/dcuser/r_data/combined_tidy_vcf.csv")
variants <- read_csv("~/r_data/combined_tidy_vcf.csv")

# tibble - table with a New Zealander accent
# observations: rows
# variables: columns

View(variants) # can also just click on the name variants in the environment pane

# This is one of my fav features of the editor. If you highlight text and press parentheses or brackets or
quotes, RStudio will place both start and end symbols

# columns can also be called vectors

glimpse(variants)
#ctrl enter

# select() and filter()
# these are dplyr functions, they will not work if you do not have that library/package installed and loaded
```

# if we only want to use specific columns
select(variants, sample_id, REF, ALT, DP)

# select all but some columns
# use - to specify columns you don't want
select(variants, -CHROM, -ALT)

# only columns that end with b
select(variants, ends_with("b"))

# to select all numeric columns
select(variants, where(is.numeric))

start_with() is part of a subset of function called tidyselect
Here's a blurb: https://tidyselect.r-lib.org/reference/language.html
They are intended to help you pick out variables in a programmatic manner. They only work inside select() and summarize()

# EXERCISE
# Select the column POS and all of the columns with the letter "i"
# Assign the output to an object called variants_subset

variants_subset <- select(variants, POS, contains("i"))

# Then remove the columns Indiv and FILTER from variants_subset
# Assign that output to an object called variants_result

variants_result <- select(variants_subset, -Indiv, -FILTER)

# Hint: use the filter contains() as one of your arguments
# Note that contains() is not case sensitive

# could also do this all in one step by combining it into one command

# filter() rows with sample_id SRR2584863
filter(variants, sample_id == "SRR2584863")

# rows that have TRUE in column INDEL
filter(variants, INDEL == TRUE)

# rows that don't have missing data in the IDV column
# ! is the not operator
# is.na() is a function to find NAs (returns true or false for each value)
filter(variants, !is.na(IDV))

#is.na() is the name of a function. Many functions have underscores separating words (ex. read_csv) others use a period instead (ex. read.csv)

# filter the variants DR to only rows where the QUAL is greate than or equal to 100
filter(variants, QUAL >= 100)

# filter with more than one condition from different columns
filter(variants, sample_id == "SRR2584863", QUAL >= 100)

# you can save any of your tables as an object and export them to a table. We'll do this later :)
# don't copy and paste from the console

# filter on two conditions in different columns with |
# this symbol | is our OR operator
filter(variants, sample_id == "SRR2584863", (QUAL >= 100 | MQ >= 50))

BREAK
come back at 11:55

# Pipes!!!!!
# take whatever is before the pipe, and use it as the input for what's after the pipe
%>%
# shortcut - ctrl shift m or command shift m
# can also use |> in base R

# filter to only sample SRR2584863 and select the REF, ALT, DP columns
variants %>% filter(sample_id == "SRR2584863") %>% select(REF, ALT, DP)

# EXERCISE
# Starting with the variants data frame,
# use pipes to subset the data to include only:
# rows from SRR2584863 sample
# where the filtered depth (DP) is at least 10
# showing only columns REF, ALT, and POS

```
variants %>%
  filter(sample_id == "SRR2584863", DP>=10) %>%
  select(REF, ALT, POS)
```

# you can use line breaks in R to make things more human readable. As long as there is an open parantheses or a pipe so R knows something is coming next, it will look at the next line for the rest of the command

# mutate
# add a new column calculated based on existing columns
# Probability = 1 - 10 ^ -(QUAL/10)
# adding column called POLPROB based on above equation
variants %>% # I didn't need to move mutate to a new line here, it just looks nice

- mutate(POLPROB = 1 - 10 ^ - (QUAL/10)) %>%
- select(POLPROB)

# if you want to save your dataframe with the mutated column
new_df <- variants %>% # I didn't need to move mutate to a new line here, it just looks nice

- mutate(POLPROB = 1 - 10 ^ - (QUAL/10)) %>%

# split apply combine
# group_by() and summarize()
# count the number of SNP in each sample
variants %>%

- group_by(sample_id) %>%
- summarize(num_snps = n())

# shortcut for group_by and count number of rows is count()
# count() = group_by() + summarize() + n()

variants %>%

- count(sample_id, name = "num_snps")

# mean, median, min, max
# calculate the mean(), median(), min(), and max() depth DP for each sample

depth_summary <- variants %>%

- group_by(sample_id) %>%
- summarize(mean_DP = mean(DP),
  - median_DP = median(DP),
  - min_DP = min(DP),
  - max_DP = max(DP))

# first depth_summary (before the comma) - what you want to write
# second depth_summary (after the comma) - what you want to name the new file
write_csv(depth_summary, file = "/home/dcuser/dc_genomics_r/depth_summary")

# look at how to remove NAs
depth_summary <- variants %>%

- group_by(sample_id) %>%
- summarize(mean_DP = mean(DP, na.rm = TRUE),
- median_DP = median(DP),

min_DP = min(DP),
max_DP = max(DP))

# USE WITH CARE
My fav, quick and easy way to drop NAs from your tibble is drop.na()

Day 6 - Data Visualization with ggplot2

Sign-in

- Trisha Adamus (she/her), Ebling Library
- Annika Pratt (she/her), Plant Pathology

- Grace Cagle (she/her), Postdoc in Soil Sciences
- Hailey Hessler
- Emilie Greene (she/her)
- Balendra Sah (He/Him)
- Madeleine Lodes (she/her)
- Avelina Gaston (she/her)
- Rachel Kirchner (she/her), Dept of Medicine
- Lukas Matthews (he/him)
- Celeste Huff (she/her)

Feedback Review

Things you learned/found useful

- Learning multiple syntax versions (bash and R)
- Clean stepwise instructions and explainations
- Referesher on R x2
- using dplyr / making different data tables to pull out specific data
- learning the purpose of a function and when to use it
- Going slowly through the materials / learning the pipeline of data analysis in R
- Pipes!
- How approchable all the instructors are
- learning basic functions

Questions / Concerns

- Can objects be renamed/adjusted once they are saved? Do you have to remove it?
  - Objects are quite flexible! You can reassign the value of an object at anytime by using the <- to give it a new value. Adjust them whenever you need to. Renaming, you can fix it in your script so it will have the right name when you rerun it. Then you can run that line or the script again and it will be updated. You may want to remove the misnamed one from your environment so you don't get confused about it later though.
- Is it common for people to use the tidyverse? Are these niche functions or common in genomic analysis?
  - The tidyverse is **very** widely used, it isnt' specific to genomic data and works on any tabular/tidy data. You will likely see it in genomic analyses and any work you do with tables.
- Need more practice using separators - commas, quotations, parentheses?
  - commas - separate arguments in a function
  - quotations - make text into literal text, otherwise R will think the words you typed are objects it need to find and sub in
  - parentheses - these are used for functions (aka actions) they hold the input values if you are giving arugments/inputs to the function.
- Is there a good reason to use python instead of R for genomics work? Is there some benefit or is it personal preference?
  - In my experience, you can pretty much use either. I recommend learning one well and you

can do whatever you need in either. There are sometimes packages only available in one or the other but they have pretty good overlap at this point. I recommend using what your peers and colleagues are so you can review each other's code and learn from other's work. Python might be a little more widely used in computing generally but R is very heavily used in health/life sciences.
- In my experience, R is more user-friendly and easier to learn, but I have heard that python handles big data better.

Notes:

materials for today:
https://uw-madison-datascience.github.io/2024-08-05-uwmadison-dc/

copy and paste top of script from yesterday
# name: Best Researcher
# email: irock@wisc.edu
# date: 2024-08-13
# description: plotting in ggplot2!

# load the packages
library(dplyr)
library(tidyr)
library(ggplot2)
library(readr)

# save script in scripts folder in dc workshop folder

# view variants table
(variants <- read_csv("~/r_data/combined_tidy_vcf.csv"))
View(variants)
head(variants)
names(variants) # gives you all column names

ggplot(data = variants, mapping = aes(x = POS, y = DP) +

- geom_point()

# ctrl enter to run

coverage_plot <- ggplot(data = variants, mapping = aes(x = POS, y = DP)

coverage_plot +

- geom_point(alpha = 0.5, color = "blue")

# alpha - transparency of points
# use colors() in the console to get the list of available colors

coverage_plot <- ggplot(data = variants, mapping = aes(x = POS, y = DP, color = sample_id) ggplot(data = variants, mapping = aes(x = POS, y = DP, color = sample_id)) +

- geom_line(alpha = 0.5) +
- geom_point()

ggplot(data = variants, mapping = aes(x = POS, y = DP, color = sample_id)) +

- geom_point(alpha = 1) +
- geom_line()

# There are so many features to play around with!
# Try different alpha values, different geoms, change the order of your geoms, change colors, etc.
# If you want to move a line up or down, you can select it and then press option/alt + up/down

ggplot(data = variants, mapping = aes(x = POS, y = DP, color = sample_id)) +

- geom_point(alpha = 0.2) +
- geom_line(alpha = 0.8) +
- labs(x = "Base Pair Position",
    - y = "Read Depth (DP)",
    - title = "Read Depth vs Position",
    - color = "SRA number")

# save your plot
# width and height are automatically in inches
ggsave("depth.pdf", width = 6, height = 4)
ggsave("depth.png", width = 6, height = 4)

# save plot as an object (it won't show up in the plots pane when you run it, but it will show up in the environment pane)
read_depth_plot <- ggplot(data = variants, mapping = aes(x = POS, y = DP, color = sample_id)) +

- geom_point(alpha = 0.2) +
- geom_line(alpha = 0.8) +
- labs(x = "Base Pair Position",
    - y = "Read Depth (DP)",
    - title = "Read Depth vs Position",
    - color = "SRA number")

ggsave(plot = read_depth_plot, filename = "depth.png", width = 6, height = 4)

Break until 10:10

Challenge

Use what you just learned to create a scatter plot of mapping quality (MQ) over position (POS) with the samples showing in different colors. Make sure to give your plot relevant axis labels.
use geom_point()

ggplot(data = variants, mapping = aes(x = POS, y = MQ color = sample_id)) +

- geom_point() +

- labs(x = "Base Pair Position",
    - y = "Mapping Quality (MQ)",
    - title = "Mapping Quality vs Position",
    - color = "SRA number") +
- theme(text = element_text(family = "Times",
    - size = 18))

# faceting
ggplot(data = variants, mapping = aes(x = POS, y = MQ color = sample_id)) +

- geom_point() +
- labs(x = "Base Pair Position",
    - y = "Mapping Quality (MQ)",
    - title = "Mapping Quality vs Position",
    - color = "SRA number") +
- facet_grid(sample_id~.) # rows by columns
- theme_bw(base_size = 12) # changes backround to white and increases font size
- ggplot(data = variants, mapping = aes(x = POS, y = MQ color = sample_id)) +
- geom_point() +
- labs(x = "Base Pair Position",
    - y = "Mapping Quality (MQ)",
    - title = "Mapping Quality vs Position",
    - color = "SRA number") +
- facet_grid(sample_id~.) # rows by columns
- theme_minimal()
- ggplot(data = variants, mapping = aes(x = POS, y = MQ color = sample_id)) +
- geom_point() +
- labs(x = "Base Pair Position",
    - y = "Mapping Quality (MQ)",
    - title = "Mapping Quality vs Position",
    - color = "SRA number") +
- facet_grid(sample_id~.) # rows by columns
- theme(panel.grid = element_blank()) # gets rid of grid lines
- # Bar plot
- # fill is the color to fill and object, color is the outline color of the object
- ggplot(variants, aes(x = INDEL, fill = sample_id)) +
    - geom_bar() +
    - facet_grid(sample_id~.)

- **Challenge**

- Use the help file for geom_bar and any other online resources you want to use to remove the legend from the plot.
- ggplot(variants, aes(x = INDEL, fill = sample_id)) +
    - geom_bar(show.legend = F) +
    - facet_grid(sample_id~.)
- # Density plot
- ggplot(variants, aes(x = DP)) +
    - geom_density()

Challenge

Use geom_density to plot the distribution of DP with a different fill for each sample. Use a white background for the plot.

```
ggplot(variants, aes(x = DP, fill=sample_id)) +
          • geom_density(alpha = 0.5) +
          • theme_classic()
```

Here are some ideas:

- See if you can change the size or shape of the plotting symbol.
- Can you find a way to change the name of the legend? What about its labels?
- Try using a different color palette (see the Cookbook for R).
- https://www.cookbook-r.com/Graphs/Colors_(ggplot2)/

Challenge: Customize one of the plots we just made (or come up with your own from “variants” data)

```
ggplot(variants, aes(x=INDEL, fill=sample_id))+
  geom_bar(alpha=0.8, show.legend=F) +
  facet_grid(~sample_id) +
  theme_classic(base_size=12)

glimpse(variants)
as.factor(variants$INDEL)
variants$sample_id %>% unique() # find unique smaple ids

# make new column when INDELS are present/absent instead of TRUE FALSE
variants_plot <- variants %>%
    • mutate(INDEL_plot = factor(INDEL, labels = c("Absent", "Present"))


ggplot(variants, aes(x=INDEL_plot, fill=sample_id))+
  geom_bar(alpha=0.8, show.legend=F) +
  facet_grid(~sample_id) +
  theme_classic(base_size=12)
```

Break until 11:15

```
# if you have an idea for something you want to do, but you don't know how to do it... Ask Google!
# Stack Overflow has answers to many coding questions
# mpg is an example dataset that comes with the tidyverse package - you can use it!

variants_plot <- variants %>%
    • mutate(INDEL_plot = factor(INDEL, labels = c("Absent", "Present")),
          • sample_id = recode(sample_id,
          • "SRR2584863" = "15k",
          • "SRR2584866" = "50k",
                    •  "SRR2589044 = "5k")) # old = new

# changing the order of the facets
# factors are ordered, and you can change the order

variants_plot <- variants %>%
    •  mutate(INDEL_plot = factor(INDEL, labels = c("Absent", "Present")),
                    •  sample_id_group = factor(sample_id,
                    •  levels = c("SRR2589044", "SRR2584863", "SRR2584866"),
                    •  labels = c("5k generations", "15k generations", "50k generations")

ggplot(variants_plot, aes(x=INDEL_plot, fill=sample_id_group))+
  geom_bar(alpha=0.8, show.legend=F) +
  facet_grid(~sample_id_group) +
  theme_classic(base_size=12) +
  labs(y = "Sequence #",
            •  x = NULL,
            •  title = expression("Insertions in"~italic(E.~coli)~ "Cit gene"))
            •  title = expression("Insertions in"~italic("E. coli")~ "Cit gene"))
# if you want to post a question on Stack Overflow
sessionInfo()
dput(head(variants, 100))

write_csv(variants_plot, "modified_sample_names_variants.csv")

saveRDS(variants, "simulation_results.RDS")

variants <- readRDS("simulation_results.RDS

# make sure that you make a reproducible example if you are asking for help online
# reprex is a package that can help
```

Closing slides: https://docs.google.com/presentation/d/1YgFjVi7pTnsg97uNJv-5hyTlmTyDAPkPus4jgSr8yBI/edit?usp=sharing