Data Carpentry Genomics Workshop

University of Kansas
Clark Instruction Center, Watson Library 3rd floor
Tuesday, January 9 - Thursday, January 11, 2022
9:00 am - 4:30 pm CST (UTC-6) all days
Lunch break each day 12:00 noon - 1:00 pm (lunch on your own)
Shorter breaks during each morning and afternoon session

Map of nearest lunch locations: https://maps.app.goo.gl/Hu7Mmu3SjMcjKVK4A
KU Dining locations near Watson are closed during the semester break.
You are welcome to bring your lunch and eat in Watson.

Course website: https://kulibraries.github.io/2024-01-09-ku-dc/
This etherpad: https://pad.carpentries.org/2024-01-09-ku-dc

Getting Help:

On Zoom (Tuesday):
Use your red X reaction or put a question or "I need help" in the Zoom chat

In person (Wednesday and Thursday):
Yellow sticky note = "I need help" / "Slow down" / "I'm not finished"
Blue sticky note = "I understand" / "I'm ready to move on"

Connection instructions for your AMI instance:

To connect on the command line:

In **macOS** or **Linux**, open your Terminal program.
In **Windows**, open Git Bash.
At your prompt, type:

```
ssh dcuser@(your AMI instance name).compute-1.amazonaws.com
```
Use the password included in your email

If you are using Windows and a command line program other than Git Bash, put this in the Zoom chat or put up your "help" sticky note and someone will help you.

To connect to RStudio:

All operating systems:
In your brower address bar, type:
```
(your AMI instance name).compute-1.amazonaws.com:8787
```
Your username and password is the one included in your email

----- ***Scroll to the bottom for FRESH notes*** ----- ***Scroll to the bottom for FRESH notes*** -----

Please fill out the pre-workshop survey located on the course website:

https://kulibraries.github.io/2024-01-09-ku-dc/#surveys

Tuesday, January 9, 2024

Attendees (Name / department or affiliation)

- Blanca Rodriguez / Ecology and Evolutionary Biology
- Huan Gong/ Pharm Chem
- Daniel Ibanez IV/ Biodivserity Institute
- Jing Tian/Pharm Tox
- Lauren Atkinson/Molecular Biosciences
- Moises Gualapuro
- Ana Cortez/Leadership Program
- Ziyu Zhu/Chemistry
- Austin Nguyen / Ecology and Evolutionary Biology
- Carolyn Jackson / K-State Libraries
- Kun Jia/PharmTox
- Claire Utzman/ Microbial Ecology
- Amar Kumar/Computational Biology
- Alfred Buabeng/KU Molecular Bioscience

Instructors (Name / department or affiliation)

- Katie Hanson/ KU Department of Molecular Biosciences
- Caroline Kisielinski / Biodiversity Institute

Helpers (Name / department or affiliation)

- Jamene Brooks-Kieffer / KU Libraries

- Boryana Koseva / Children's Mercy Research Institute
- Vanna Hay/ KU Anthropology
- Sarah Unkel / KU Anthropology
- Daniel Montezano / KU Computational Biology

Project Organization and Management

9:20 AM with Katie
Lesson material: https://datacarpentry.org/organization-genomics/

As scientists we generate a lot of data even prior to sending samples out to be sequenced - that's metadata and it's important

- -use both lab notebooks and excel to manage this metadata
    - -follow proper practices for organizing excel sheets

KEEP RAW DATA RAW

When submitting your samples to be sequenced:

- -will have to fill out a form with the sample metadata
    - -look at example for reference: https://datacarpentry.org/organization-genomics/files/sample_submission.txt
    - -consistant in dates(year,month,day)
    - -computer will not register "A" and "a" as the same thing, keep consistant
    - -abbreviate (wild type--wldt)
    - -keep decimals consistant (look at Volume and RIN columns-what is wrong here?)
    - -avoid using special characters (R will not recognize the special character in the Volume column, for example. Hint: use "ul" instead)
    - -Volume and Concentration columns are very important to include, the sequencing team will have to equal out those volumes when processing

Once you get your sequence results back:

- example sheet: https://datacarpentry.org/organization-genomics/files/sample_submission.txt
- -organized by sample_id
    - -so to associate this data with out submission, use the sample_id to match up with tube_barcode
- -filename:
    - We did pair end sequencing
    - -_R1 is read 1 (foward strand)
    - -_R2 is read 2 (reverse strand)
    - -.gz extension means this is a compressed gzip format (saves disc space)
        - -when you read these files into the terminal, you will input them as this zipped format and then unzip after
- -file size:
    - -genomic data is HUGE

        - -this columns is over a terabyte of data alone
    - -need to be sure you are able to transfer this large amount of data and not loose any (check that files after transfer are the same size as on the sample sheet)
- -ALWAYS keep a backup of this file
    - -store it in a place that is accessible by at least you and your PI
    - -store where there is backups(resFS, KU community cluster), back up in at least 2 places)
        - -leave backups RAW(allows for clear, reproducable data)
    - -ask your PI if your department has a secure server you can have access to

<u>Yes we generate our own data, but in most studies we also utalize reference data</u>

- -most studies are required to make their data publicly avaliable
- -many repositories to do so, but these are common:
    - -NCBI (https://www.ncbi.nlm.nih.gov/)
    - -NCBI sequence read archive (https://trace.ncbi.nlm.nih.gov/Traces/sra/)
    - -EMBL-EBI (https://www.ebi.ac.uk/)

Introduction to the Command Line

9:59 AM with Katie
Lesson material: https://datacarpentry.org/shell-genomics/

<u>The shell:</u> https://datacarpentry.org/shell-genomics/instructor/01-introduction.html

- -a computer program that introduces the command line interface
- -allows to control computer with keyboard instead of our typical graphical interface
- -your mouse will really only be used to scroll up and down
- -it takes time to get comfortable with the command line, don't worry it just takes practice!
- -the shell makes work less boring as well
    - -working on large amount of files
        - -the shell makes many operations automated so you do not have to edit each file individually
        - -also much faster and not suseptible to human typing error
- -makes work more reproducable, everyone has access to the same software as you

- **<u>How to access the shell:</u>**

- go to search bar and open either:
- Mac/Linux: Terminal
- PC: command prompt or Git Bash (either one is fine)
- look up above under "To connect on the command line:" section for instructions for logging on.
- Note: this AMI (or AWS) information came in the same email that included the Zoom information for today's session
- -physically type "yes" to security prompt, this is just your computer asking about safety, but we trust this connection
- -Note: when typing in password you will not see anything, the screen will appear blank but it is typing, then hit enter.
- -Once logged in, you will see a bunch of information about past logins, etc.
    - -Do not need. To get rid of, type "clear". Should give you a blank slate.

        - -did not lose that info (scroll up if want to review).
            - **[Moving around in the shell:](https://datacarpentry.org/shell-genomics/instructor/02-the-filesystem.html)** **https://datacarpentry.org/shell-genomics/instructor/02-the-filesystem.html**
- -Essentially you are just accessing a series of directories that hold folders
- -when see the "$" it means the shell is waiting for a new command from you
- -Where are we in our directories?
    - -type "pwd" (print working directory)
        - -example output: /home/dcuser
            - -we are in a directory called dcuser which is stored inside of our home directory
- -What is in this directory?
- -type "ls" (listing)
        - -example output: R   r_data   shell_data (other directories inside of dcuser)
- -How do we move directories?
- -type "cd shell_data"and hit enter- you are now in the /home/dcuser/shell_data directory
    - - "~/shell_data$"
    - HINT: "~" stands for your home directory
    - -what is in this folder? Type "ls" to see what is stored here
    - -want to know what these file types are? use options!
        - -ls -F (the option -F tells us what type of file/directories are in the folder)
            - Example: sra_metadata/; the "/" means this is a directory
- -want to know what else you can do with "ls"?
            - -type "man ls" ("man" is short for manual)
                - -this opens the manual that gives us the discription of this command and its options
                    - -can scroll with arrow keys or use space bar to go down a page
                    - -hit "b" to go back
                    - - hit "g" to return to top of manual
                    - - hit "q" to get out of the manual and return to your shell screen
    - -Exercise: try using the -l option with ls, what does this tell us? Hint: ls is the command, -l is the option for the command
        - -give us name of the owners of the files and when they were last modified
        - -in the beginning the "d" means this is a directory
        - -the "rwxr-x---" is the premissions of who can read, write, and edit these files
- - now go into untrimmed_fastq directory
    - -"cd untrimmed_fastq"
    - - once in this directory, type"ls -F"
        - -you will see you have 2 fastq files (these files hold the sequences and their quality scores)
- -return to home directory: type "cd" and hit enter
    - -you are now back in your home directory, your landing page to start out; where all of your files are stored

TIP: When coding you have to spell correctly and pay attention to capitalization! To help avoid these errors, we can use tab completion. Example: type "cd she" and hit <tab>. This will finish out typing your command for you (as long as that directory exists within that directory).

- - notice when trying on directories/files with similar headings, the tab completeiong will only complete as far as they match up and then will ask you to put in next unique identifer in the title
    - -example: ls SRR09 (there is a SRR097977 and a SRR098026) so hit <tab> <tab> to see what options you have in your folder.
- <u>-Not sure about which command you are looking for?</u> use <tab> <tab>
    - example: I know it starts with a "pw". So I type "pw" <tab> <tab> and it will show me all avalible commands that begin with "pw"
- <u>-What if I want to go back one folder?</u> Type "cd .."
- <u>-What about moving back 2 directories?</u> Type "cd ../../"
- -Now return to home directory "cd"
    - -go to shell_data directory
        - -cd shell_data
- <u>Exercise:</u> There is a hidden file in this directory. How do we find it? Identify the name of the file. Hint: check the options in the manual ("man ls").
    - <u>-Answer:</u> Use -a option with ls (ls -a); this option ignores entries that begin with a "." (entries that start with a "." are hidden). Then go to the .hidden directory ("cd .hidden") and type "ls". The file is called "youfoundit.txt".
        - <u>Tip:</u> you have to tell the computer to make the directory/file hidden; it is not automatically
            - Example: "mkdir .hidden_dir" makes a hidden directory called ".hidden_dir"
            - However some software will automatically install hidden directories/files so they are not messed with by accident and usally do not need to be altered.

<u>Working with files:</u> https://datacarpentry.org/shell-genomics/instructor/03-working-with-files.html

- Can I see what is found within a lower directory without going to that directory? Yes, type the relative path. <u>Example</u>: "ls shell_data/untrimmed_fastq/"
    - The *relative path* is based on where you are currently.
    - The *absolute path* is not dependent on your current location/perspective
        - <u>Example:</u> "cd /home/dcuser/shell_data/.hidden
            - can run this command from anywhere in your files and will take you to this place
- Now move to untrimmed_fastq: "cd ~/shell_data/untrimmed_fastq"
    - Lets look at our fastq files contained in this folder: "ls *.fastq"
    - <u>Wildcard:</u> The "*" symbol is a wildcard, in the sample above it is listing anything that ends with .fastq. Another example is with "ls SRR* to find anything that begins with SRR. If want to find a file/directory with specific middle part type "ls *"..", so ls *977* finds files with 977 in the middle of the name.

<u>TIP:</u> if you want to re-run a command you have already typed, hit the up arrow on your keyboard and it will run through your previous commands. Can also type "history" to look at command history. Can use !# to run a specfic command line. Example "!539" will run the command on line 539 of the history. (Note: this will refresh everytime you log off).

- Now return to your untrimmed_fastq directory, "cd ~/shell_data/untrimmed_fastq"
- <u>How can I view a specifc file?</u>
    - "cat *filename*"
        - example "cat SRR098026.fastq"
        - This will print the entire file to the screen. However this is not optimal for large files.

        - To view a similar read only version of the file type "less *filename*"
            - note this is a read only copy/you cannot edit the file with cat or less.
            - less is easier to navigate through and uses the same commands as man
        - How do I search within that file?
            - type "/"characters to search"; ex: "/CAA"
        - leave cat or less by typing "q"
    - Can I view just a specfic part of the file, not the whole file?
        - Type "head *filename*" to see first 10 lines of the file
            - Hint: can choose how many lines you see, "head -n #of lines *filename*" (ex: head -n 5 *filename*)
        - Type "tail *filename*" to see the last 10 lines
            - Hint: same as above you can specify the number of lines you see at the end with tail -n #oflines *filename*.

How do we read a fastq file?

- Each read is split up into 4 lines:
- First line begins with "@". The first line tells you about file discription and the length
- The second line shows the actual sequence
- Third line always begins with a "+" and often has the same info as line 1
- The fourth line is the quality score of each read. Aka what is the probablity this base was called correctly?
    - Follows the PHRED scoring:
    - Quality encoding: !"#$%&'()+,-./0123456789:;<=>?@ABCDEFGHIJK
    - | | | | |
    - Quality score: 0........10........20........30........40..
    - This quality score is logarithmically based, so a quality score of 10 reflects a base call accuracy of 90%, but a quality score of 20 reflects a base call accuracy of 99%. Hint: If it looks like cursing in a cartoon("#$%!") it is not a good score.

Creating files:
You NEVER want to change your raw data!
So to use the data, first make a copy!

- To copy a file: use "cp"; cp *filetocopy nameofcopy*
    - example: "cp SRR098026.fastq SRR098026-copy.fastq"
- Usually want to store our copies somewhere else
- So let's make a directory:
    - mkdir *directoryname*
        - example "mkdir backup"
- Now let's move our copy to that directory:
    - mv copyfilename wheretomovefile
        - example: "mv SRR098026-copy.fastq backup"
- Want to rename a file?
    - mv *filenametochange newfilename*
        - example: mv SRR098026-copy.fastq SRR098026-backup.fastq

File Permissions:

- who can read and write to a file
- in the permissions, we have 10 positions:
    - drwxrwxrwx The first [rwx] is for the user, the second set of [rwx] is for a specifc group

and the last set of [rwx] is for other.
    - R =read only
    - w= write/make changes to
    - X=execute
- To edit permissions used chmod
  - example: chmod -w *filename*
    - this makes us unable to write the file
  - To check if it worked: use rm
    - example: rm *filename*
      - The computer will stop you and double check if you want this write-protected file to be deleted.

TIP: once you use rm to remove you can never get that file/directory back!!

- rm only works with files
- Want to remove a directory?
  - rm -r *directoryname (-r stands for recursively)*

1:00pm with Caroline

Lesson material: https://datacarpentry.org/shell-genomics/instructor/04-redirection.html

Searching files:

- **Grep** is a useful command for searching files for specific information
  - Example: grep NNNNNNNNNN *filename* will pull out all instances of "NNNNNNNNNN" in that file
  - can add arguments to the grep command
    - Example: grep -B1 -A2 "NNNNNNNNNN" *filename*
      - B1 asks for the line before that string of characters
      - A2 asks for the 2 lines after that string of characters
- **Tip: if computer freezes on you after entering an inccorect command, hit cntrl+c (not command)**
- How do I put the grep output into a new file?
  - grep "..." filename > newfilename; ex. grep NNNNNNNNNN SRR098026.fastq > bad_reads.txt
    - this puts the grep findings into this new text file instead of printing it directly to the screen.
- How do I know how many words are in my file?
  - **wc** is a command for word count
    - wc filename
    - Example: wc bad_reads.txt shows lines, words, and charater counts in that order
    - or if just want lines, use option -l; ex: wc -l bad_reads.txt
  - Exercise: how do we know how many of those lines are reads? Hint: what do we know

about fastq files?

  - Answer: Divide the total lines in bad_text.txt by 4. So 537/4=total reads that are bad. Note: this will give you a decimal, but that is because before the last read in the file there is a "--" which makes our answer a decimal.
    - The "--" is an artifact of using grep.
      - This means the grep skipped the lines in between these two lines. For instance in this example you have read 133 and then read 177. So grep skipped over 134-176.
- I want to do the same thing to my other fastq file and add it to my bad_reads file.
  - use grep
    - To *overwrite* what is in the bad_reads file with the new data use: grep -B1 -A2 NNNNNNNNNN SRR097977.fastq > bad_reads.txt
    - To *combine* new data with the other data in the file use ">>"
      - grep -B1 -A2 NNNNNNNNNN SRR097977.fastq >> bad_reads.txt
- But lets make this process faster for all of my files:
  - grep -B1 -A2 NNNNNNNNNN *.fastq > bad_reads.txt
    - now we have done the same thing as before but we did both files at once!
    - remember to use that wildcard * it will save you a lot of time!
- Can we make this command even less cumbersome? Yes, with piping!
  - a **pipe** is the symbol | and can be used to string together multiple commands on one line
    - example: grep -B1 -A2 NNNNNNNNNN SRR098026.fastq | wc -l
      - so we are asking the computer to pull out all reads with these 10 consecutive N's in this file and then tell us the resulting number of lines, all in one go
        - should just recieve an output of the line count immediatly (in this example it is 537)
  - piping can be used for many purposes, another example is we can remove the "--" in the bad_reads.txt file
    - grep -B1 -A2 NNNNNNNNNN SRR098026.fastq | grep -v '^--' > bad_reads.txt
      - -v means remove the lines with characters that are in the '...' quotations.
      - The ^ within the quotations means at the beggining of the line; so it will only pull out lines with -- where the -- is at the beginning of the line
    - now if you "tail bad_reads.txt", you should not have any lines with -- anymore, additonally your word count should change to 536
    - **Writing For loops: (https://datacarpentry.org/shell-genomics/instructor/04-redirection.html#writing-for-loops)**
- For loops are a way to further automate your workflow. With for loops we can execute many commands on a large sample set repetitively and in a timely manner.
- Shell is designed to work with for loops
- First, Lets practice making variables, example:
  - First make a variable: foo
    - "foo=abc"
  - To check what foo stands for use echo. "echo foo is $foo"
    - $ symbol before the variable name represents the value of a variable

        - echo foo is ${foo}EFG
            - will result in "foo is abcEFG"
        - tip: "echo" means to print to the screen
- Now lets make a for loop.
        - **for filename in *.fastq** then hit *enter*
            - filename is another variable here
            - **Tip:** make variable names obvious, so you know what you are doing when you return to this code later
            - for loops always start with "for"
        - **> do**
            - once you see the >, it means you are in the loop, on line 2 type "do" and then hit enter to go to next line, then type:
        - **> head -n 2 ${filename}**
            - then hit enter
        - **> done**
            - hit enter and then your loop will start
            - "done" signals to the computer to go back to the beginning of the loop and run again until done.
        - So this for loop is giving us the first two lines of every file in this directory, it goes through one file then loops around and works on the second etc etc until it has ran through all files.
        - **Making the same for loop but now writing output into new file:**
        - **for filename in *.fastq**
        - **> do**
        - **> head -n 2 ${filename} >> seq_info.txt**
        - **> done**
    - Note: you will still know what fastq file each read came from if you look at the first line in the read.
        - **Using basename in for loops: (https://datacarpentry.org/shell-genomics/instructor/04-redirection.html#using-basename-in-for-loops)**
    - after you work with your file a lot, with each change you may change the name or the file extension
    - Example: basename SRR097977.fastq .fastq
        - This will remove the ".fastq" at the end and just give us SRR097977
        - Using this in a loop:
            - **for filename in *.fastq**
            - **> do**
            - **> name=(basename ${filename} .fastq)**
                - we are creating a variable called "name", that variable is the result of the entire function that follows. So this loop is cycling throuh all of our .fastq files and listing the basenames without the .fastq at then end.
            - **> echo ${name}**
            - **> done**
        - **Practice, lets create a loop that appends the date to each our text file names:**
- **for filename in *.txt**
- **> do**

        - **> name=$(basename ${filename} .txt)**
        - **> mv ${filename} ${name}_2024_01_09.txt**
        - **> done**
    - <u>Tip: cntrl+c can get you out of your loop environment</u>

<u>Writing files: (https://datacarpentry.org/shell-genomics/instructor/05-writing-scripts.html)</u>

- <u>to make brand new files:</u>
    - use: **nano** *filename*
    - <u>example</u>: nano README.txt
        - we are now in a text file called README.txt which we can edit/write in
            - at the bottom you can see different control ooptions (cntrl X lets you exit; cntrl o lets you save your changes)
        - can type whatever you want into the .txt file; but README files often have a description of what is in directory or what a software does.
    - after exiting the nano workspace, you should have a new README.txt file when you type ls
    - <u>Exercise</u>: re-open your README.txt file and add the date to that file.
- <u>We can use the command line to write scripts.</u> **Scripts** are powerful tools in bioinformatics. We can string together mutliple commands and save them so they are repoducible.
    - They take a lot of work to write intially, but once you have a script that works is saves a lot of time as you can use it again and again with new data
- Lets practice:
    - **nano bad_reads_script.sh**
        - **grep -B1 -A2 -h NNNNNNNNN *.fastq | grep -v '^--' > scripted_bad_reads.txt**
            - <u>note:</u> grep is in purple, nano recognized that grep is a command and is color coding it to help us with our writing. As we write nano has further colorized some of our syntax for us; redirect features are now in green and our search pattern is in yellow.
        - save file with **cntl+o** to save and then **cntl+x** to exit.
        - <u>run script:</u> **bash bad_reads.sh**
            - should have output of scripted_bad_reads.txt that contains all bad reads
        - **bash** calls our script for us
    - <u> If wanted to turn this into an executable program, would have to change permissions:</u>
            - **ls -l bad_reads_script.sh**
                - shows permissions, currently there is no x, so it is not executable
            - **chmod +x bad_reads_script.sh**
                - **ls -l bad_reads_script.sh**
                    - should now have the x which means it is executable
            - <u>to run this program:</u> **./bad_reads_script.txt**

<u>Moving and downloading data from external sources:</u>

(https://datacarpentry.org/shell-genomics/instructor/05-writing-scripts.html#moving-and-downloading-data)

- often we have to download reference or comparative datasets from a public data source and put

into our command line to use.
- Two main programs:
    - wget
    - curl
        - some envirnments have one program and some have the other
            - to learn which one you have use "which *program_name*"
                - example: "which wget" gives you /usr/bin/wget which means you have it.
- navigate back to home directory ("cd")
        - type: wget ftp://ftp.ensemblgenomes.org/pub/release-37/bacteria/species_EnsemblBacteria.txt
            - hit "enter" and you will see the download progress
            - note: sometimes you want to download from an http websource, if so, make sure to change the link to say http instead of ftp (in both places).
    - OR type:curl -O ftp://ftp.ensemblgenomes.org/pub/release-37/bacteria/species_EnsemblBacteria.txt
        - -O flag, which simultaneously tells curl to download the page instead of showing it to us and specifies that it should save the file using the same name it had on the server: species_EnsemblBacteria.txt

Save files to your personal(local) computer from the remote computer:

(https://datacarpentry.org/shell-genomics/instructor/05-writing-scripts.html#transferring-data-between-your-local-machine-and-the-cloud)

- return to shell_reads/untrimmed_fastq, we will download our bad_reads_script.sh to our computer
    - **scp <file I want to move> <where to move the file>**
        - scp stands for secure copy
- open up a new window
    - Tip: 'PS1='$ ' will remove your username from the screen and make it look cleaner.
- In the new window, we are in our local environment, but the file we want is in our amazon instance on a remote computer.
- To move this remote file to your personal computer, in the new window:
    - **cd Desktop/**
    - or make sure you are in desktop: cd -> ls, if you have Desktop/ then cd to desktop;
        - however pc users may have their desktop in OneDrive: cd \Users\username\OneDrive\Desktop\
        - For OneDrive on KU, the path looks like /c/Users/j408b004/OneDrive - University of Kansas/Desktop/ (from GitBash)
    - once in your desktop:
        - **scp *login_info*: /home/dcuser/shell_data/untrimmed_fastq/bad_reads_script.sh .**
            - note: pc users may have to use \ instead of /
            - the period at the end is saying to copy this file to our current directory
        - check for successful download: **ls *.sh**

- There are other gui's you can use for downloading files to your personal machine:
  - Cyberduck (Mac or PC)
  - mobaxterm (PC only)

Data Organization:https://datacarpentry.org/shell-genomics/instructor/06-organization.html

- Easy to forget about but very important. You want to keep track of what you did because I promise you will not remember when you come back to it in a month.
- Genomics projects in particular, can quickley accumulate tons of files.
- Find your organization process and stay consistant with it.
- Just like you have a lab notebook, you can keep a bioinformatic notebook.
  - In your bioinformatic notebook you can have answers to questions like:
    - What were your best alignment results?
    - Which folder were they in: Analysis1, AnalysisRedone, or AnalysisRedone2?
    - Which quality cutoff did you use?
    - What version of a given program did you implement your analysis in?
- Next, return to your amazon instance and then redirect to your home directory.
  - <u>Exercise:</u> create a series of directories:
    - dc_workshop
    - dc_workshop/docs
    - dc_workshop/data
    - dc_workshop/results
- it is always good to have your data on two different servers (in case 1 crashes) as well as backed up on a hard drive.
- Remember you can also use the chmod -w to give you that back up which stops you when you try to remove that file.
- Also can use "history" to review previous code.
  - Only want to see a few lines? Try "history | tail -n 10" for example, to show the last 10
- You can create your own digital log to keep track of what you have done:
  - history | tail -n 7 >> dc_workshop_log_2024_01_09.sh
    - should now have this .sh file in your directory
    - However the above history command is listed in our new file, but we do not want that, so:
  - nano dc_workshop_log_2024_01_09.sh
    - then go in and delete that line
    - can go up to the top of the file and create notes using the # symbol for you to know what you did here in the future.
      - <u>Example:</u> #2024-01-09
      - #Created sample directories for the Data Carpentries Workshop

<u>TIP:</u> <u>The symbol '#' tells the computer this is not code to run but personal notes.</u>

    - cntrl+O and then cntrl+X to leave

<u>How to log out of your terminal:</u>

  - type "exit" and hit enter

Introduction to Cloud Computing

Notes go here

Wrap up today

Wednesday, January 10, 2024

Attendees (Name / department or affiliation)

- Blanca Rodriguez / Ecology and Evolutionary Biology
- Amar Kumar/Computational Biology
- Lauren Atkinson
- Moises Gualapuro/ Computational Biology
- Austin Nguyen / Ecology and Evolutionary Biology
- Claire Utzman/ Microbial Ecology
- Ana Cortez/ Leadership Program
- Ziyu / KU Chemistry
- Alfred Buabeng/KU Molecular Bioscience

Instructors (Name / department or affiliation)

- Katie Hanson / KU Molecular Biosciences
- Caroline Kisielinski | Biodiversity Institute

Helpers (Name / department or affiliation)

- Sarah Unkel / KU Anthropology
- Vanna Hay/ KU Anthropology
- Boryana Koseva / Children's Mercy Research Institute
- Daniel Montezano / KU Computational Biology
- Jamene Brooks-Kieffer / KU Libraries

Data Wrangling and Processing

<u>Fastq File Downloading</u>

- Make a new directory in your home (~) directory
    - **Command**: mkdir -p ~/dc_workshop/data/untrimmed_fastq/
        - Option -p makes new directories within the given path
        - Confirm you have made this directory using the command pwd
- Download the data (This may take a while)

        - **Commands**:
            - curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR258/004/SRR2589044/SRR2589044_1.fastq.gz
            - curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR258/004/SRR2589044/SRR2589044_2.fastq.gz
            - curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR258/003/SRR2584863/SRR2584863_1.fastq.gz
            - curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR258/003/SRR2584863/SRR2584863_2.fastq.gz
            - curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR258/006/SRR2584866/SRR2584866_1.fastq.gz
            - curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR258/006/SRR2584866/SRR2584866_2.fastq.gz
        - Use ls command to see in the directory. You should have six files. These files are for three samples, each with reads 1 and 2.
- Uzip the fastq.gz files. It is not necessary to do this.
    - **Command**: gunzip SRR2584863_1.fastq.gz
- View these files as needed
    - Remember, line 1 begins with the @ symbold and provides the sample info, line 2 provides the DNA sequence, line 3 begins with a + symbol, line 4 provides the quality score for each nucleotide
    - **Command**: head -n 4 SRR2584863_1.fastq.gz
        - The head command provides the first four lines within the file because we use option -n 4
    - **Command**: tail -n 4 SRR2584863_1.fastq.gz
        - The tail command provides the last four lines within the file because we use option -n 4
    - Tip: if your read quality (line 4) looks like a swear word, your reads are likely not good quality.

<u>Assess the quality of the reads within these files</u>

- The program FastQC provides box and whisker plots to demostrate the quality of our reads
    - Boxes represent the 1st-3rd quartile range while the whiskers represent the 0 to 4th quartile range
    - Each value on the X axis represents the reads at a given base pair (bp) position
    - Values on the Y axis show the measured quality of these reads
    - The red line indicates the median quality
    - Boxes within the green area are of good quality, boxes within the yellow area is acceptable, and boxes within the red area are of poor quality.
- **Command**: fastqc *.fastq*
- Use the ls command to check the outputs.
    - You should see files ending in .fastq, .fastqc.html, .fastqc.zip for each sample.
- Create a new directory

        - **Command**: mkdir -p ~/dc_workshop/results/fastqc_untrimmed_reads
    - Move the .zip files to the new directory
        - **Command**: mv *.zip ~/dc_workshop/results/fastqc_untrimmed_reads/
    - Move the .html files to the new directroy
        - **Command**: mv *.html ~/dc_workshop/results/fastqc_untrimmed_reads/
        - Check: You should only see the .gz files in the original directory
    - Move to the newly created directory
        - **Command**: ~/dc_workshop/results/fastqc_untrimmed_reads/
        - Check: You should see the .zip and .html files for each sample
    - Now use the secure copy protocol (scp)
        - Open a new terminal window, do not log into the AMI
        - Use the ls command to list the direcotries within your home directory. If you see Desktop listed simply move to your Desktop directory.
            - Command: cd Desktop
        - If you do not see the Desktop directory listed, use your file explorer to locate your Desktop directory. Use this directory path below.
            - Remember this will be different for everyone
            - **Command**: cd ./<directory path to your Dektop>
        - Now we aare ready to move the files. Use the terminal window set to your local computer's Desktop directory.
        - **Command**: scp dcuser@<unique AP address>:~/dc_workshop/results/fastqc_untrimmed_reads/*.html .
    - Now, navigate to your Desktop using your file explorer and select the .html file to view the fastQC results

<u>Working with the .zip files</u>

- Retrun to your AMI terminal
    - Use ls to list the files in your directory. You should see .zip and .html files
    - Unzip the .zip files
        - **Create a for loop**:
            - for filename in *.zip
            - do
            - unzip $filename
            - done
        - Use ls -F to check the contents of the directory. You should not see additional directories (indicated by / at the end).
            - Use ls -F to check the contents of one of these directories.
                - **Command**: ls -F SRR2584863_1_fastqc
                - You should see the files below
                    - fastqc_data.txt fastqc.fo fastqc_report.html Icons/ Images/ summary.txt
        - View the summary.txt file
            - **Command**: less SRR2584863_1_fastqc/summary.txt
            - This file provides PASS and WARN for various parameters of the sample
            - To exit the summary.txt file hit q
        - Concatenate the summary files for each sample
            - **Command**: cat */summary.txt > ~/dc_workshop/docs/fastqc_summaries.txt
        - Move to the docs directory
            - **Command**: cd ~/dc_workshop/docs

        - Use the ls command to check the contents. You should see a file named fastqc_summaries.txt
        - To view the file. Hot q to exit
            - **Command**: less fastqc_summaries.txt
        - Search for FAIL within the file
            - **Command**: grep FAIL fastqc_summaries.txt
            - You should see a few lines of fails. That's okay.

<u>Cleaning the reads via trimming</u>

- View the options for the program Trimmomatic that we will be using to trim.
    - **Command**: trimmomatic
    - PE stands for Paired End and SE stands for Single End reads
    - We are working with paired end reads, so trimmomoatic expects two input files per command (reads 1 and 2 per sample).
- Move to the untrimmed_fastq directory
    - **Command**: cd ~/dc_workshop/data/untrimmed_fastq
- Copy NexteraPE-PE.fa to your current working directory. Remeber to add the space and . at the end to copy to the directory. The NexteraPE-PE.fa file has the adapter sequences to be trimmed.
    - **Command**: cp ~/.miniconda3/pkgs/trimmomatic-0.38-0/share/trimmomatic-0.38-0/adapters/NexteraPE-PE.fa .
    - Use ls to check that you now see NexteraPE-PE.fa in your working directory
- Running trimmomatic on a fastq file
    - **Command (See Tip below)**: trimmomatic PE SRR2589044_1.fastq.gz SRR2589044_2.fastq.gz \
    - SRR2589044_1.trim.fastq.gz SRR2589044_1un.trim.fastq.gz \
    - SRR2589044_2.trim.fastq.gz SRR2589044_2un.trim.fastq.gz \
    - SLIDINGWINDOW:4:20 MINLEN:25 ILLUMINACLIP:Nextera-PE-PE.fa:2:40:15
        - SLIDINGWINDOW
            - Looking at windows consiting of 4 reads, cutting quality scores less than 20 within this window
        - MINLEN
            - Sets the minimum length of the reads to 25 base pairs
        - ILLUMINACLIP
            - Indicates the adaptor to be cut (must be in the working directory
            - Don't worry about what the numbers mean. They are specific to adaptors.
        - **Tip**: Using the \ lets the shell know you are not done coding if you wish to continue on a new line without hitting enter to execute a command
        - The resulting output will be automaticly visible in the terminal.
        - Use ls -l -h to check the contents of the directory
            - The sizes of the fastq.gz files should now be samller after the trimming is complete.
        - Zip the unzipped files
            - **Command:** gzip SRR2584866_1.fastq
        - Run trimmomatic on multiple samples
            - **Create a 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
                - Refer to the notes on SLIDINGWINDOW, MINLEN, ILLUMINACLIP commands from above.
                - Check that you have fastq.gz, .trim.fastq.gz, and un.trim.fastq.gz files in the directory using the ls command

<u>Sequence Alignment</u>

- Use the cd command to change your working directory to the dc_workshop directory
- Use ls to ensure you have a directory called data. If you do not have this directory use the mkdir command to create one.
- Make a directory
    - mkdir -p data/ref_genome
- Download the reference genome
    - **Command**:
    - 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
    - Tip: Use of the -L option enables the program to be redirected in the case of a URL moving
- Unzip the reference genome
    - **Command**: gunzip data/ref_genome/ecoli_rel606.fasta.gz
- Additional Download
    - **Command**:
    - curl -L -o sub.tar.gz https://ndownloader.figshare.com/files/14418248
    - Use ls command to check directory contents. You should see the sub.tar.ga file. This is a form of archived file.
- Untar the file (extracts the contents)
    - **Command**: tar xvf sub.tar.gz
    - Make directory within the data directory
        - **Command**: mkdir -p data/trimmed_fastq_small
    - Move the subdirectory into the new directory above
        - **Command**: mv sub/ ~/dc_workshop/data/trimmed_fastq_small/
    - Make additional directories within the data directory
        - **Command**: mkdir -p results/sam results/bam results/bcf results/vcf
- Index the reference genome
    - **Command**: bwa index data/ref_genome/ecoli_rel606.fasta
- Algin samples to the reference genome
    - Format of command below: bwa mem ref_genome.fasta <input_file_R1.fastq> <input_file_R2.fastq > <output.sam>
        - Requires 4 arguments including the reference genome, the forward reads, reverse reads, and output file
    - **Command**: bwa mem data/ref_genome/ecoli_rel606.fasta data/trimmed_fastq_small/sub/SRR2584866_1.trim.sub.fastq data/trimmed_fastq_small/sub/SRR2584866_2.trim.sub.fastq >

    results/sam/SRR2584866.aligned.sam
    - Check the contents in the output directory
        - **Command**: ls results/sam/
        - SAM stands for Single Alignment Mapping file
- Conver the SAM files into a BAM file
    - **Command**: samtools view -S -b results/sam/SRR2584866.aligned.sam > results/bam/SRR2584866.aligned.bam
    - Option -b indictaes that the output is a bam file
    - Option -S ensures version compatability
- Sort the alignment file
    - **Command**: samtools sort -o results/bam/SRR2584866.aligned.sorted.bam results/bam/SRR2584866.aligned.bam
    - Option -o indicates that the output should be written to new file
    - Check that you have created the sorted file in your results/bam directory
- View the stats of your alignment
    - **Command**: samtools flagstat results/bam/SRR2584866.aligned.bam

<u>Varaint Calling</u>

- SNV (common file type) stands for Single Nucleotide Variant
- Variant frequency caluclation of the sorted .bam file in comparision to the reference genome
    - **Command**:
    - bcftools mpileup -O b -o results/bcf/SRR2584866_raw.bcf \
    - -f data/ref_genome/ecoli_rel606.fasta results/bam/SRR2584866.aligned.sorted.bam
        - Option -O signals the output type
            - Here we use b to indicate a bcf file as the output type
        - Option -o indicates the output file
        - Option -f indicates the reference genome input
- Idenitfy the single nucleotide variants
    - **Command**:bcftools call --ploidy 1 -m -v -o results/vcf/SRR2584866_variants.vcf results/bcf/SRR2584866_raw.bcf
        - Option -m enable multiallelic and rare-variant calling
        - Option -v indicates that we want variants sites only
        - Options -o specifies the location of the output file
    - Check that you have generated a vcf file using ls results/vcf
- Filter the vcf file
    - **Command**: vcfutils.pl varFilter results/vcf/SRR2584866_variants.vcf > results/vcf/SRR2584866_final_variants.vcf
    - Tip: vcfutils.pl is a pre-written perl script. We know this is a perl script by the extension .pl
    - Check that you have generated a final_variants.vcf file using ls results/vcf
- View vcf file
    - **Command**: less -S results/vcf/SRR2584866_final_variants.vcf
        - Option -S allows long lines to be split for ease of readability
- Index the sorted bam file
    - **Command**: samtools index results/bam/SRR2584866.aligned.sorted.bam
- To view how the genome of interest aligned to the reference
    - **Command**: samtools tview results/bam/SRR2584866.aligned.sorted.bam data/ref_genome/ecoli_rel606.fasta
        - The first line shows the coordinate positions along the genome
        - The second line shows the sequence of the reference genome

        - The third line shows the consensus sequence for the sample
            - Dots represent positions that have aligned to reference genome
        - Letters indicate a variant present in reads within the lines below the solid horizontal line
        - To exit tview you can use q

<u>Writing a script for all the above commands to automate our research</u>

- Make a new directory
    - **Command**: mkdir -p ~/dc_workshop/scripts
- Change into the new directory
    - **Command**: cd scripts
- Create a new file (without opening the new file)
    - Command: touch read_qc.sh
- Use nano to open the script
    - **Command**: nano read_qc.sh
- Write the script of commands in nano
    - Note: set -e Tells the program to stop working if it encounters an error
    - **Copy commands as formatted below:**
        - set -e
        - cd ~/dc_workshop/data/untrimmed_fastq/
        - echo "Running FastQC ..."
        - fastqc *.fastq*
        - mkdir -p ~/dc_workshop/results/fastqc_untrimmed_reads

        - echo "Saving FastQC results ..."
        - mv *.zip ~/dc_workshop/results/fastqc_untrimmed_reads
        - mv *.html ~/dc_workshop/results/fastqc_untrimmed_reads

        - cd ~/dc_workshop/results/fastqc_untrimmed_reads

        - echo "Unzipping ..."
        - for filename in *.zip
        - do
        - unzip $filename
        - done

        - echo "Saving summary ..."
        - cat */summary.txt > ~/dc_workshop/docs/fastqc_summaries.txt
    - To exit nano
        - Command: Control+x
        - Save the file? Y
        - Hit enter
- Run the script
    - **Command**: bash read_qc.sh

View and run another prepared scipt for variant calling

- Download the script
    - **Command**:
    - curl -O https://datacarpentry.org/wrangling-genomics/files/run_variant_calling.sh
- View the script
    - Command: nano run_variant_calling.sh
    - Edit the script
        - At the line beginning with "for" see below correction to add directory called "sub"
            - for fq1 in ~/dc_workshop/data/trimmed_fastq_small/sub/*_1.trim.sub.fastq
        - At the lines beginning with "fq1" and "fq2" see below correction to add directory called "sub"
            - fq1=~/dc_workshop/data/trimmed_fastq_small/sub/${base}_1.trim.sub.fastq
            - fq2=~/dc_workshop/data/trimmed_fastq_small/sub/${base}_2.trim.sub.fastq
        - Control+X
        - Save the changes? Y
- Run the script
    - **Command**: bash run_variant_calling.sh

You don't need to scp the script files from your AMI instance; they are here in the Etherpad or in the lesson materials:
https://datacarpentry.org/wrangling-genomics/05-automation.html

Illumina Sequencing Video: https://www.youtube.com/watch?v=fCd6B5HRaZ8

Anonymous feedback for the afternoon session:
https://kusurvey.ca1.qualtrics.com/jfe/form/SV_dbbtkFnMXjausV8
Please indicate in your feedback whether you were in person or on Zoom; this will help us tomorrow!

Wrap up today

Thursday, January 11, 2024

Attendees (Name / department or affiliation)

- Carolyn Jackson / K-State Libraries
- Daniel Ibanez / biodiversity institute
- Lauren Atkinson
- Austin Nguyen/ Ecology and Evolutionary Biology
- Ana Cortez
- Moises Gualapuro / Computational Biology
- Amar Kumar/Computational Biology

- Blanca Rodriguez / Ecology and Evolutionary Biology
- Jing Tian/pharm tox
- Claire Utzman / Microbial Ecology
- Alfred Buabeng/ KU Molecular Bioscience

Instructors (Name / department or affiliation)

- Caroline Kisielinski | Biodiversity Institute

Helpers (Name / department or affiliation)

- Sarah Unkel /KU Anthropology
- Vanna Hay /KU Anthropology
- Boryana Koseva / Children's Mercy Research Institute
- Jamene Brooks-Kieffer / KU Libraries
- Daniel Montezano / KU Computational Biology

Introduction to R and RStudio for Genomics

<u>Log Into RStudio using your web browser</u>

- Type in your unique IP address and hit enter.
- Use the username and password provided to log in.
- RStudio is a grphical interface for the R program

<u>Create an RStudio project</u>

- This creates a directory for your project, similar to directories in the shell.
- To create a RStudio project click the green plus sign with R behind it in the top left hand corner
- Select New Directory, then New Project
- Directory name: dc_genomics_R
- Click Browse and select the ~
- Click Create Project

<u>Create a RStudio Script</u>

- Click the green plus to create a script
- Save script by clicking the blue save button and name the script: genomics_r_basics

<u>Getting to know the RStudio enviornment</u>

- The top left panel is called the source panel and serves as your workspace where you will be creating your script and executing commands
- The top right panel is called the Environment/History. This shows you the datasets and objects (variables) you have created as well as their properties such as type and dimensions.
- The bottom left panel is called the Console and shows the commands executed. This display is similar to what you will see in the regular R program.
    - R uses a > to indicate the prompt (like $ in shell)
- The bottom right panel displays your working directory and the its contents
    - Files tab holds the files in the directory
    - Plots holds the plots that you create

        - Packages tab holds the packages you have loaded into R
        - Help tab allows you to look up functions

Writing comments

- Comments allow you to describe the purpose of the script and the functions within it
- Use the # symbol to initiate a comment. It will turn green.
    - **Type**:
    - #DC Genomic Workshop Intro to R Analysis
    - #Introduce the basics of R and how to utilize R for genomics for genomic analysis
    - #2024/01/11

Making a Section

- **Type**: #Directories####
    - You should se this show up in the Source column just to the right (may need to click the stacked horizontal lines

To Execute Commands Below

- Hit Control+Enter for PC/Command+Return for Mac
- If you see a red X to the left of the source panel, R is alerting you that something with the command is wrong

Check Working Directory

- **Command**: getwd()
    - In R you will want to make sure to use () within a command
- Click Run at the top right
    - You should now see the command displayed below in the Console

Change a Direcotry

- **Command**: setwd("/home/dcuser/dc_genomics_r")
    - You should see the command in the console

Functions

- #Functions####
    - To see what is in your directory
    - **Command**: dir()
    - To look up info about your packages, platform
        - **Command**: sessionInfo()

Getting Help with Functions using the Function round() as an Example

    - To get help with a function, place a ? at the beginning of a command
        - **Command**: ?round()
            - You will then see some helpful information in the bottom right panel
    - Alternatively you can use the command args as below for a function
        - **Command**: args(round)

Making Objects

- #Objects####
- Objects need a name, value, and assingnment operator (<-)
    - #name
    - #value

    - #<-
- **Command**: a <- 1
    - This indicates that the object named a is assigned the value 1
    - Check the value of a using a as a simple command
        - You should see the value of 1 returned in the console
- **Tip:** Shortcut for the Assignment Operator
    - PC: Alt+Minus
    - Mac:Option+Minus
- **Command**: human_chr_num <- 23
- Assignments do not have to be numerical.
    - Use quotation marks
    - **Command**: gene_name <- "pten"
- To print this object value
    - **Command**: gene_name
- You can string objects together to create a function
    - **Command**: human_diploid_chr_num <- 2 * human_chr_num
- To reasssign an object's value simply reenter the command to assign the value.
    - You will see the value update in the environment
- To remove an object us the remove command
    - **Command**: rm(gene_name)
    - This shoudl remove the object from the environment

<u>3 Data Types</u>

- #Numeric Number
- #Character strings of letters or numbers that are in ""
- #Logical Boolean Values TRUE or FALSE
- **Command**: chrom_num <- "chr02"
    - To check the type of data
        - **Command**: mode(chrom_num)
            - You should see that this in a character data type

<u>Mathematics in R</u>

- # + or - for addition and subtraction
- # * multiplication
- # / division
- # ^ or ** exponents
- # 11 %/% 4 division to the nearest whole number
- # 11 %% 4 division with remainder
- Example function wiht multiple operators
    - **Command**: round((1+sqrt(5))/2, 3)
    - You should get the value of 1.618

<u>Vectors</u>

    - These are a series of values of the same data type
        - **Comman**d: snp_genes <- c("OXTR", "ACTN3", "AR", "OPRM1")
    - Check the values were concatenated
        - **Command**: snp_genes
        - You should see these values returned
    - To detemine the length of a vector
        - **Command**: length(snp_genes)

        - You should see the value of 4 returned
    - To detemine the structure of a vector
        - **Command**: str(snp_genes)
            - You should see the value of [1:4] returned

Indexing

- Allows you to pull values based on the position using the [] instead of ()
    - **Command**: snp_genes[3]
    - You should see "AR" appear
- You can pull a consecutive series of values
    - **Command**: snp_genes[1:3]
- You can pull a non-consecutive series of values
- Uses the c() to combine within the []
    - **Command**: snp_genes[c(1:2,4)]
    - "OXTR" "ACTN3" "OPRM1" should be returned

Adding and Removing and Renaming values from vectors

- Add using the c()
    - **Command**: snp_genes <- c(snp_genes, "CYP1A1", "APOAS")
    - You should see these values have been added
- Remove using the
    - **Command**: snp_genes <- snp_genes[-6]
    - You should see the sixth value has been removed
- To rename a value
    - **Command**: snp_genes[1] <- "first snp"
    - You should see that the first values has been renamed to first snp

Logical Subsetting

- To ask if a value is greater than another
    - **Command**: snp_pos[snp_pos > 100000000]
- Other operators
    - # < > for greater OR less than
    - # <= >= for greater than or equal to AND less than or equal to
    - # == for equal to
    - # a|b for a OR b
    - # A&B for A AND B

To Read in Data from an external source into RStudio

- Load our prepared csv file in RStudio
    - **Command**: variants <- read.csv("/home/dcuser/r_data/combined_tidy_vcf.csv")
- View the data in the csv file within the Console
    - **Command**: summary(variants)
- See the structre of the file
    - **Command**: str(variants)
- To see the first six lines of the file
    - Command: head(variants)
- To see the first six lines of the file
    - Command: tail(variants)

View the dataset in the Source in a more easily readable format

- **Command**: View(variants)

Pull a subset of the data from the whole set

- **Commands**:
- subset <- data.frame(variants[,c(1:3, 6)])
- str(subset)
- This should return data from the first through 3rd column as well as the sixth

Factors

- Vectors specialized in categorical data
- **Commands**:
  - alt_alleles <- subset$ALT
  - str(alt_alleles)
- This shoulde retrun data from column ALT (indicated by the $ symbol)

To plotting categorical variables

- **Command (typed as a single, full command)**:
  - snps <- c(alt_alleles[alt_alleles == "A"],
  - alt_alleles[alt_alleles == "T"],
  - alt_alleles[alt_alleles == "G"],
  - alt_alleles[alt_alleles =="C"])
- Plot
  - **Command**: plot(factor_snps)
- Make factor_snps into a table, then sort the table in increasing numerical observations, then we are pulling out the names
  - **Command**: ordered_factor_snps <- factor(factor_snps, levels = names(sort(table(factor_snps))))
  - **Command**: plot(ordered_factor_snps)
    - This should return a plot sorted by increasing order of observations
- Pull out specific rows and columns
  - **Commands**:
    - variants [1,1]
    - variants [2,]
    - variants [, 1]
    - variants [1:5, 1:3]
    - variants [c(1:3, 6), c(1:3, 6)]

Changing datatypes

- This changes the numerica data types to characters
- **Commands**:
  - snp_pos
  - str(snp_pos)
  - snp_pos_chr <- as.character(snp_pos)
  - str(snp_pos_chr)
- This changes the character data back to numeric
  - **Commands**:
  - snp_pos_chr <- as.numeric(snp_pos_chr)
  - str(snp_pos_chr)
- Alternatively you can use these **commands:**
  - as.numeric(factor_snps)

        - as.character(factor_snps)

Create a data frame and save

- Make a data frame
    - **Commands**:
        - SRR2584863_variants <- variants[variants$sample_id == "SRR2584863",]
        - head(SRR2584863_variants)
        - unique(SRR2584863_variants$sample_id)
- Save as a csv file
    - **Command**:
        - write.csv(SRR2584863_variants, file = "SRR2584863_variants.csv")

Anonymous feedback for the morning session:
https://kusurvey.ca1.qualtrics.com/jfe/form/SV_dbbtkFnMXjausV8
Please indicate whether you are in person or on Zoom - thank you!

Afternoon Session:

How to install packages:

- Cran: storage of R packages in the R software
- Command to install a package: install.packages("*packagename*")
    - **Example:** install.packages("ggplot2")
- Can also download packages via bioconductor.org: an open source software repository for bioinformatic tools
    - How to download from bioconductor:
            - **First, install the bioconductor manager:**
                - **command:** install.packages("BiocManager")
                - **Check which version installed**: BiocManager::version()
            - Now you can install packages from bioconductor:
                - **command:** BiocManger::install("vcfR")
    - How to navigate bioconductor:
        - Can see list of packages with **command**:
            - BiocManager::available()
- Can look for packages in the R cran as well by going to the *packages* tab in the lower right corner of the rstudio screen and using the search bar.
- Once you have installed the package, you need to load it to your workspace.
    - **Command:** library(vcfR)
- Tidyverse
    - within the cran there is a unified effort to create tidy data (no spaces in names, input is uniform, etc.)
    - Tidyverse is a whole sweet of packages that all use this same type of data structure but on big data (genomic data) and can communicate with and build upon each other.
        - Allows for reproducable research
        - ex: ggplot2 is one of these packages and there are other packages that can build upon ggplot2

- <u>Download Tidyverse packages with these **commands**:</u>
    - install.packages("dplyr")
        - dplyr provides tools for basic data manipulation, such as looking into dataframes.
    - install.packages("tidyr")
    - install.packages("readr")
    - <u>Note</u>: these are just a few of the packages available in tidyverse, there are many more you can use. Can also install tidyverse itself but that takes considerable time.
- Load packages into library once downloaded with **commands**:
    - library("dplyr")
    - library("tidyr")
    - library("ggplot2")
    - library("readr")

<u>Tip:</u> only have to install packages once, but must use the library() command to call in these packages to your workspace every time you want to use them.

Using Tidyverse to modify dataframes:

- <u>Import your own csv file with</u> **command**: variants <-read_csv("/home/dcuser/r_data/combined_tidy_vcf.csv")
    - <u>note</u>: Notice that we imported a cv earlier but when doing so with tidyverse you use a '_' instead of a '.'
- look at your data structure with **command:** glimpse(variants)
- <u>look into columns in your dataframe using the function select()</u>
    - **command:** select(variants, sample_id, REF, ALT, DP)
        - 1st argument is dataset, second is the columns you want pulled out
- <u>look at all columns except one:</u>
    - **command:** select(variants, -CHROM)
- <u>look at all columns that end with a specfic character/number:</u>
    - **command:** select(variants, ends_with("B"))
- <u>look into rows in your dataframe with the function filter()</u>
    - **command:** filter(variants, sample_id == "SRR2584863")
    - **command:** filter(variants, INDEL)
    - **command:** filter(variants, sample_id == "SRR2584863", QUAL >= 100)
        - filtering out rows with this sample id that have a quality score greater than 100
- <u>look through rows and columns with piping (%>%):</u>
    - pipe shortcut for mac: cmd + shift + m
    - pipe shortcut for pc: ctrl + shift + m
    - **command:** SRR2584863_variants <- variants %>%
        - filer(sample_id == "SRR2584863") %>%
        - select(REF, ALT, DP)
- <u>cut through dataframe with slice()</u>
    - **command:** SRR2584863_variants %>% slice(1:6)
- <u>create new columns from old ones using mutate()</u>
    - say you want to make a logged version of your data and then add it back into original dataframe.

        - **command:** variants %>%
            - mutate(POLYPROB = 1-(10 ^ -(QUAL/10)))
        - note: the dataframes in tidyverse are called tibbles. Whereas in base R we were just using a regular dataframe. With tibbles you will see the dataframe as long as the screen allows and then will get gray writing underneath that tells you about the rest of the columns not shown.
- splitting your data into a specific group with group_by()
    - say you want to see how many variants you have per sample:
        - **command:** variants %>%
            - group_by(sample_id) %>%
            - tally()
- reshape your dataframe:
    - *long format:* when every row is an observation that repeats
    - *wide format:* does not have values that repeat in first column
    - covert from long format to wide:
        - **command:** variants_wide <- variants %>%
            - group_by(sample_id, CHROM) %>%
            - summarize(mean_DP = mean(DP)) %>%
            - pivot_wider(names_from = sample_id, values_from = CHROM)
        - mean_DP is the mean read depth
        - summarize() is the function for basic summary statstics
        - this command makes every column the sample id name and every row the mean read depth of the samples

GGPLOT: Creating Plots

- make sure you have the ggplot2 library loaded, **command:** library(ggplot2)
- load in dataset with **command:**

variants <- read.csv("https://raw.githubusercontent.com/naupaka/vcfr-for-data-carpentry-draft/main/output/combined_tidy_vcf.csv")

**TIP: Remember to annotate your scripts so you know what each line of code is doing. This is a love letter to your future self as you will not remember what past you did. This will save you a TON of time in the future. **

- look at your dataset:
    - **command:** glimpse(variants)
    - **or command:** head(variants)
        - note: you will only get the first 6 lines automatically with head()
- the ggplot structure:
    - when you use ggplot, first you import your data, then whow you want to have things mapped with the asethetic and them geoms, the way you want data visualized,like points vs lines(data= <DATA>, mapping = aes(<MAPPINGS>)) + GEOM_FUNCTION()

- **command:** ggplot(data = variants, aes(x = POS, y = DP)) +
    - geom_point()
- <u>can turn your plot into objects and manipulate.</u>
    - **command**: coverage_plot <- ggplot(data = variants, aes(x = POS, y = DP))
    - **command**: coverage_plot + geom_point()

<u>TIP:</u> There are many types of geoms you can use to make many types of plots, from geom_point() for scatterplots to geom_line() for line graphs. But google around and see the other types to plot new kinds of graphs.

- <u>To change the transparency of the data on the plot use alpha():</u>
    - **command:** coverage_plot + geom_point(alpha = 0.5)
- <u>To change the color use color():</u>
    - **command:** coverage_plot + geom_point(alpha = 0.5, color = "blue")
- <u>To change color to match sample ID:</u>
    - **command:** ggplot(data = variants, aes(x = POS, y = DP, color = sample_id)) +
        - geom_point(alpha = 0.5)

<u>Note:</u> this is not changing your underlying data it is just represnting it in a plot

- <u>Labelling axis and graph title:</u>
    - **command**: ggplot(data = variants, aes(x = POS, y = DP, color = sample_id)) +
        - geom_point(alpha = 0.5) +
        - labs(x = "Base Pair Position,

y = "Read Depth (DP)",
title = "Read Depth vs. Position")

- <u>To save your plot:</u>
    - **command:** ggsave("depth.pdf", with = 6, height = 4)

Stylizing plots with themes and facets:

<u>TIP</u>: Remember ggplot (and all of R)is open source, so you can google all kinds of themes and customization packages for it. It is really endless what you can find online, you can copy how the wall stree journal makes plots or make a color blind friendly plot for example..

- <u>Applying themes to your plot with a theme() layer</u>
    - Yes themes can change colors and such but you can edit other elements(like text) as well
    - command: ggplot(data = variants, aes(x = POS, y = DP, color = sample_id)) +
        - geom_point(alpha = 0.5) +
        - labs(x = "Base Pair Position,
    - y = "Read Depth (DP)",
    - title = "Read Depth vs. Position") +
        - theme(text = element_text(family == "Bookman"))
        - note: remember to keep your data legible and clear (so maybe this font is not the best, but cool to know you can do this)
- <u>Facet plots with ggplot</u>

        - facets seperate everything by whatever variable you give it
        - **command:** ggplot(data = variants, aes(x = POS, y = MQ), color = sample_id)) +
            - geom_point() +
            - labs(x = "Base Pair Position",
            -         y = "Mapping Quality (MQ)") +
            - facet_grid(~ sample_id)
        - change the orientation of the facets
            - ~ is a place holder
            - *rows* are denoted by first spot (rows, ~)
            - *columns* are the second spot (~, columns)
    - <u>Pre-set themes:</u>
        - type in "theme" in rstudio and all kinds of theme options will pop up (like black and white is theme_bw)
        - **command**: ggplot(data = variants, aes(x = POS, y = MQ), color = sample_id)) +
            - geom_point() +
            - labs(x = "Base Pair Position",
            -         y = "Mapping Quality (MQ)") +
            - facet_grid(~ sample_id)
            - theme_bw()+
            - theme(panel.grid = element_blank())
            - <u>Note:</u> this last line removed the gridlines from the background

Making Bar Plots in ggplot

- <u>first build your base plot:</u>
    - **command:** ggplot(data = variants, aes(x = INDEL, fill = sample_id)) +
        - geom_bar()
        - <u>note:</u> fill is how the colors of the bars in the plot are assigned, here it will be colored by sample id.
- <u>now make it a faceted bar graph as the green sample is overpowering the others:</u>
    - **command:** ggplot(data = variants, aes(x = INDEL, fill = sample_id)) +
        - geom_bar() +
        - facet_grid(sample_id ~ .)
- <u>Here if we run color instead of fill you will see only the outlines are colored now:</u>
    - **command:** ggplot(data = variants, aes(x = INDEL, color = sample_id)) +
        - geom_bar() +
        - facet_grid(sample_id ~ .)

Making Density Plots in ggplot

- <u>first make your base layers (the data type, aes, plot type):</u>
    - **command:** ggplot(data = variants, aes(x = DP))+
        - geom_density()

Note: Type in "geom_" to see suggestions of all kind of plots avaliable to plot

- other themes
  - to make it dark **command:** ggplot(data = variants, aes(x = DP))+
    - geom_density() +
    - theme_dark()
  - to make it remove the background **command**: ggplot(data = variants, aes(x = DP))+
    - geom_density() +
    - theme_minimal()
  - these are just two examples, you can find themes that change everything about your plot in different ways, just explore!

Note: each + on ggplot is a new layer on the plot so you can use both geom_bar and geom_point, you can keep adding to your base plot with more and more layers to make a very detailed figure!

ADVICE: “Never memorize something that you can look up” -- A. Einstein

Remember you are not alone with the errors and problems you encounter. In fact, many problems probably already have a solution out there by someone who has already encountered.

Tip: O'Reilly practical developer series is a useful set of books with step by step code and you can find them pretty cheap online.

Looking for help online:

For general R questions, Stack Overflow is probably the most popular online community for developers. If you start your question “How to do X in R” results from Stack Overflow are usually near the top of the list. For bioinformatics specific questions, Biostars is a popular online forum.

List of sites to ask for help:

- https://www.seqanswers.com/
- https://stackoverflow.com/
- https://www.biostars.org/

TIP: data-2-viz.com is a great resource that tells you what kind of plots you should use for specific types of data. Addionally, if you go to the "dedicated page" for the plot type, you can see an example and get the code for making that plot :https://www.data-to-viz.com/. But note it is not field specifc, and may not always help with genomic data, but nonetheless, still a GREAT resource! (Not all of your figures may be about the genomic data, fo instance you may need to discuss your workflow or sample set)

Asking for help using online forums:

- When searching for R help, look for answers with the ‘r’ tag: https://stackoverflow.com/questions/tagged/r

- Get an account; not required to view answers, but you need an account to post
- Put in effort to check thoroughly before you post a question; folks get annoyed if you ask a very common question that has been answered multiple times
- Be careful. While forums are very helpful, you can't know for sure if the advice you are getting is correct
- See the How to ask for R help blog post for more useful tips: https://blog.revolutionanalytics.com/2014/01/how-to-ask-for-r-help.html

Key points

- R provides thousands of functions for analyzing data, and provides several way to get help
- Using R will mean searching for online help, and there are tips and resources on how to search effectivel
- **Genomic data is too big to run on your own personal computer (if it is medical/health data it is illegal to store on your personal computer).** Your lab should have a high performance cluster or cloud setting for you to store and process your data!

Lesson for Computing in the Cloud:

- https://datacarpentry.org/cloud-genomics/04-which-cloud.html

Storage resources from KU:

Research File Storage: https://technology.ku.edu/catalog/research-file-storage
250GB free; $75/TB/year after that; can be billed to a grant or department

Your lab can apply for free HPC space and computing power from https://web.archive.org/web/20220820060934/https://www.xsede.org/
You must apply to NSF with a proposal (usually have high approval rate).

But you must justify the space and power you are requesting in your proposal and then have to provide a report by the end of the tenure.

ACCESS is the successor to XSEDE: https://access-ci.org/
There is an ACCESS contact on campus to facilitate these proposals - contact Jamene for more info

Wrap up the workshop and post-workshop feedback

Post workshop survey from The Carpentries: https://carpentries.typeform.com/to/UgVdRQ?slug=2024-01-09-ku-dc - please let us know how we did!

KU Carpentries News mailing list: https://lists.ku.edu/listinfo/ku-carpentries-news

- News about future workshops

KU Carpentries Instructors mailing list: https://lists.ku.edu/listinfo/ku-carpentries-instructors

- Future opportunities to help, get to know other KU instructors, explore Carpentries Instructor Training

----- end -----

丹尼尔·蒙特扎诺