**Part 1 – For Loops, Pipelines, and TDF Visualization**

Author: Daniel Ramírez, 2022

Samuel Hunter, 2023

--- STAGING WORKING AREA IN SCRATCH ---

Navigate to \****your\**** github repo clone. Git pull to get the updated repo.

cd /scratch/Users/your\_username/sr2023

git pull

Today we’ll be working with multiple FASTQ files at once. Copy all files from:

/scratch/Shares/public/sread2023/data\_files/day5/fastq/

sample1\_day5\_igv.RNA.end1.fastq

sample1\_day5\_igv.RNA.end2.fastq

sample2\_day5\_igv.RNA.end1.fastq

sample2\_day5\_igv.RNA.end2.fastq

To your Day5 working directory. Make a directory for error and output files, if you don’t already have one!

In the past couple days, we’ve run sample one-by-one using individual scripts for each step. While this is fine to do, it becomes very tedious when we’re talking about dozens (or even hundreds!) of samples. Instead, we can use a second script to submit jobs automatically. First, let’s explore **for loops:**

1. Make a new script called example\_for\_loop.sh and type:

A black background with white text

Description automatically generated

1. Exit vim and run the script:

A black screen with white text

Description automatically generated

Notice your script ran the “echo $index” command multiple times, but the output changed from 0 to 3. For loops perform the code in the body of the loop for each entry in the sequence you gave it. We iterate over these values and assign them to the variable “index” for each loop (you can set the variable name to whatever you want).

1. We can do this with other commands too, not just numbers. Edit your example\_for\_loop.sh to instead use ls:

A black screen with white text

Description automatically generated

1. And run it as before:

A screen shot of a computer code

Description automatically generated

Now, the value of “index” is set to each sequential value in the output of our ls command. Let’s change the body of the for loop now to run a different command other than echo.

1. Edit your example\_for\_loop.sh:

A computer screen with white text

Description automatically generated

Note you’ll need to include the indir path too, since we’re running a command on the file itself!

1. Run the script

A black background with white text

Description automatically generated

Now, instead of just printing the file name, we’re printing the first line of the file.

Notice that we’ve only been running one process though. If we want to process many large files on the cluster, we need to submit multiple jobs. How would we utilize loops to submit multiple jobs?

1. Open up the script **d5-fastq-to-tdf.sbatch:**

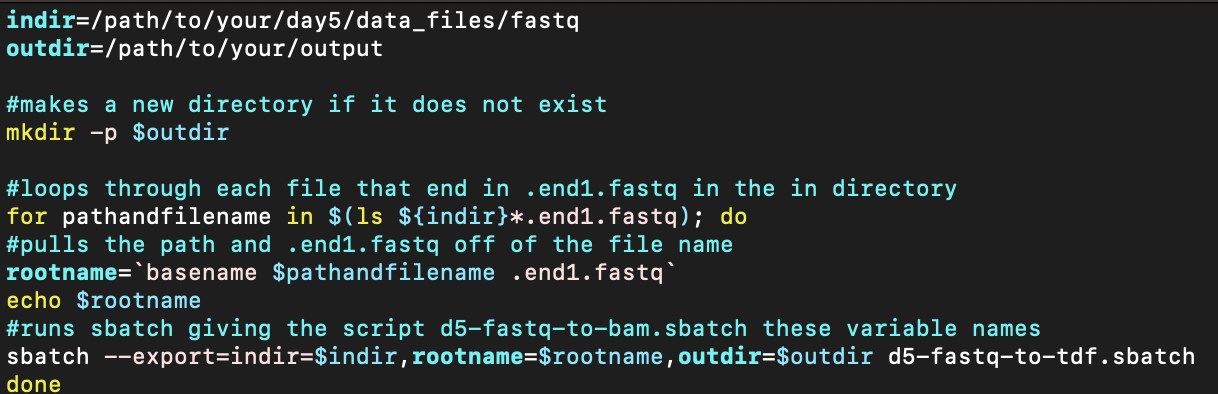
A computer screen with text and numbers

Description automatically generated

1. Edit the SBATCH parameters for your email, and e\_and\_o paths. Note that we’re setting FILENAME, INDIR, BAM, SAM, and QC, but we never set the value for rootname, indir, or outdir. We’ll set these using another script.

The rest of the script is a combined pipeline of everything you ran this week. There’s no need to edit anything else. Look through each step of the pipeline- you’ll see it covers every step from QC of your original FASTQ files, mapping, bedGraph generation, all the way up to compressed TDF files for visualization.

1. Open up a new file and write the script below. Save it as: runloopfastqtotdf.sh



1. Change the path for indir to your Day5 fastq directory, and the path for outdir to your desired output directory. Then, take a look at the for loop starting on line 8:

The first line lists all of the files in indir which match the pattern \*.end1.fastq. For each iteration, our loop will use the next file name and store that value in the variable pathandfilename

Next, the value for rootname is set using the basename command. This strips off the .end1.fastq portion of each file.

Finally, we use sbatch –export to submit the script **d5-fastq-to-tdf.sbatch** as a job on the compute cluster. –export assigns each variable in the new job to a new value. In this case, for each job, “rootname” will change each time to a new file.

Save and exit the file.

Since sbatch is used inside the script to actually submit the jobs, all we have to do is run the loop:

A black background with white text

Description automatically generated

Now we have TDF files, just as we learned during Day 4.

**Part 2 - Week 1 assessment**

List the files in /scratch/Shares/public/sread2023/day5/assessment\_fastq

Pick any of the available day5 datasets (or do them all together with a loop!). Adapt the scripts you’ve made in the first week and run the following steps on your dataset:

1. Run a QC check on the raw FASTQ files
2. Trim the FASTQ files and run a QC check again
3. Map the FASTQ files to hg38
4. Compress the mapped files into BAM files
5. Generate bedGraph files from BAM files
6. Generate TDF files from bedGraphs
7. Transfer the TDF to your local computer and visualize using the IGV Web App

BONUS: Combine the scripts into a single pipeline that takes the FASTQ files all the way to TDFs.

