Part 1 – Bedgraphs and TDFs, For Loops, Pipelines

Author: Daniel Ramírez, 2022
Samuel Hunter, 2023

--- STAGING WORKING AREA ---

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

```
cd /Users/your_username/sr2023
git pull
```

Make a directory for error and output files for today, if you don't already have one!

Bedgraphs and TDFs

We'll be picking up where we left off yesterday: starting with BAM files and converting them to Bedgraphs and their compressed form, TDFs. If you didn't make it all the way through the scripts yesterday, don't worry. We've provided the files:

/scratch/Shares/public/sread2023/cookingShow/day5/bam/
chr21Eric_repA.RNA.sorted.bam
chr21Eric_repA.RNA.sorted.bam.bai

Navigate to your github repository:

```
cd /Users/your_username/sr2023/day05/scripts
```

Open d5-bam-to-tdf.sbatch. Change the job name to "bam_to_tdf". Add your user email in place of <YOUR_EMAIL>. Change the path of –output and –error to your eofiles directory. Finally, change the value of OUTDIR to your desired output directory. Remember, delete the <> brackets!

```
#!/bin/bash
#SBATCH --job-name=<JOB_NAME>                          # Job name
#SBATCH --mail-type=ALL                                   # Mail events (NONE, BEGIN, END, FAIL, ALL)
#SBATCH --mail-user=<YOUR_EMAIL>                 # Where to send mail
#SBATCH --nodes=1                                          # Numbers of nodes
#SBATCH --ntasks=4                                        # Number of CPU (tasks)
#SBATCH --time=00:30:00                                 # Time limit hrs:min:sec
#SBATCH --partition=compute                           # Partition/queue requested on server
#SBATCH --mem=10gb                                       # Memory limit
#SBATCH --output=/scratch/Users/<YOUR_USERNAME>/eofiles/%x_%j.out
#SBATCH --error=/scratch/Users/<YOUR_USERNAME>/eofiles/%x_%j.err

################ SET REQUIRED VARIABLES ################################################


# First, we specify our file name, location, and outdir. ONLY change the outdir path to YOUR
# user directory. The rest of the script can be kept the same!

FILENAME=chr21Eric_repA.RNA
INDIR=/scratch/Shares/public/sread2023/cookingShow/day5/bam/
OUTDIR=/scratch/Users/<YOUR_USERNAME>/day5
```

The script is now ready to run. Read through the rest of the script, but don't change anything else!

```
[[sahu0957@ip-172-31-29-36 scripts]$ sbatch d5-bam-to-tdf.sbatch
```

This script is much longer than what you ran yesterday. We will be generating a histogram of reads for each genomic location in our BAM file (this is known as a bedGraph).

Because the BAM file is paired-end, we first have to separate Read1 and Read2 into separate files. Then, we generate bedGraphs for the + and – strands separately. This gives us a total of 4 bedGraph files. We then join those files back together into a single bedGraph which has both strands.

Check your error and output files to make sure the script ran correctly. When you have a TDF file, send it to your local computer and open it up in the IGV web app (or the Desktop version if you have it installed).

Bonus: For Loops and Pipelines

If you finish the portion above before we start on the assessment, start working on the next section. Otherwise, go straight to the Assessment on the last page.

List the files in the following directory:
/scratch/Shares/public/sread2023/data_files/day5/fastq/for_loops_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

In the past couple days, we've run samples 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. Go to your ~/sr2023/day05/scripts directory. Make a new script called example_for_loop.sh and type:

```
for index in $(seq 0 3)
do
echo $index
done
```

2. Exit vim and run the script:

```
-bash-4.2$ bash example_for_loop.sh
0
1
2
3
```

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).

3. We can do this with other commands too, not just numbers. Edit your example_for_loop.sh to instead use ls:

```
indir=/path/to/your/data_files/fastq
for index in $(ls $indir)
do
echo ${index}
done
```

4. And run it as before:

```
-bash-4.2$ bash example_for_loop.sh
sample1_day5_igv.RNA.end1.fastq
sample1_day5_igv.RNA.end2.fastq
sample2_day5_igv.RNA.end1.fastq
sample2_day5_igv.RNA.end2.fastq
-bash-4.2$
```

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.

5. Edit your example_for_loop.sh:

```
indir=/path/to/your/data_files/fastq
for index in $(ls $indir)
do
head -n 1 ${indir}/${index}
done
```

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

6. Run the script

```
-bash-4.2$ bash example_for_loop.sh
@HWI-ST753:238:C6VYVACXX:6:1101:1221:96440/1
@HWI-ST753:238:C6VYVACXX:6:1101:1221:96440/2
@HWI-ST753:238:C6VYVACXX:6:1101:1160:11268/1
@HWI-ST753:238:C6VYVACXX:6:1101:1160:11268/2
```

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?

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

```
#!/bin/bash
#SBATCH --job-name=fastqtotdf                          # Job name
#SBATCH --mail-type=ALL                                   # Mail events (NONE, BEGIN, END, FAIL, ALL)
#SBATCH --mail-user=YOUREMAIL@colorado.edu                # Where to send mail
#SBATCH --nodes=1                                         # Numbers of nodes
#SBATCH --ntasks=4                                        # Number of CPU (tasks)
#SBATCH --time=00:30:00                                   # Time limit hrs:min:sec
#SBATCH --partition=compute                              # Partition/queue requested on server
#SBATCH --mem=10gb                                        # Memory limit
#SBATCH --output=/path/to/your/e_o/%x_%j.out
#SBATCH --error=/path/to/your/e_o/%x_%j.err

################ SET REQUIRED VARIABLES ################################################


FILENAME=$rootname
INDIR=$indir
BAM=$outdir/bams/
SAM=$outdir/sams/
QC=$outdir/qc/
```

8. 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.

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

```
indir=/path/to/your/day5/data_files/fastq
outdir=/path/to/your/output

#makes a new directory if it does not exist
mkdir -p $outdir

#loops through each file that end in .end1.fastq in the in directory
for pathandfilename in $(ls ${indir}*.end1.fastq); do
#pulls the path and .end1.fastq off of the file name
rootname=`basename $pathandfilename .end1.fastq`
echo $rootname
#runs sbatch giving the script d5-fastq-to-bam.sbatch these variable names
sbatch --export=indir=$indir,rootname=$rootname,outdir=$outdir d5-fastq-to-tdf.sbatch
done
```

10. Change the path for indir to the Day5 fastq directory: /scratch/Shares/public/sread2023/data_files/day5/fastq/for_loops_fastq/

    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.

11. Save and exit the file. Run the loop to submit all of the jobs at once!

```
-bash-4.2$ bash runloopfastqtotdf.sh
sample1_day5_igv.RNA
Submitted batch job 9086126
sample2_day5_igv.RNA
Submitted batch job 9086127
```

Now we have TDF files, all with just one script.

Part 2 - Week 1 assessment

This Assessment is so you can test yourself on what you've learned so far. This is for you to track your progress, it is not a test! Keep track of how long it takes you to complete the full pipeline for processing FASTQ files.

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.

Condition	Biological Replicates	Read Pairs
Chr21 Eric	RepA	End1
		End2
	RepB	End1
		End2
	RepC	End1
		End2
Chr21 Ethan	RepA	End1
		End2
	RepB	End1
		End2
	RepC	End1
		End2