Student B number: **B236494**

Hi!
Well done for all the effort that you have put into doing BPSM ICA1!
There were two things that need to be completed for this ICA.

1. **50% of the marks**: a PUBLIC GitHub repository called ICA1 containing a `ccrypt` encrypted tar.gz file called Bxxxxxx-2023.ICA1.tar.gz.cpt, as detailed in the BPSM git and GitHub lecture. The tar.gz.cpt file should **only** contain the pipeline programme/code for me to run on our server, **as well as the full git log** of this coding project (in a `all_the_things_I_did` file). The pipeline programme/code itself should contain lots of "comments" so than anyone looking at the code knows what each bit is doing
2. **50% of the marks**: a PDF file called Bxxxxxx-2023.ICA1.pdf, submitted via Learn Dropbox, that:
    - gives the link for your PUBLIC GitHub ICA1 repository (e.g. https://github.com/Bxxxxxx-2023/ICA1)
    - gives the `ccrypt` encryption key to open the Bxxxxxx-2023.ICA1.tar.gz.cpt file
    - has an overview/flowchart that describes how the pipeline processes the data
    - briefly indicates why you have chosen the programme parameters/flags that you have for each step
    - indicates any things the user will have to do to make things work (hopefully very few!)
    - highlights any difficulties, if any, that you have come across
    - indicates any additional/alternative features that you think might be beneficial to include

---

Code comments

- **Could I clone the GitHub repository asked for?**
  Yes, no problems!
- **Date/time on GitHub repo**

```
git log | grep Date
```

```
Date:   Sun Oct 22 19:54:42 2023 +0100
```

- **Could I decrypt the file with the password given?**
  Yes, worked fine!
- **Did the GitHub repository only include the pipeline code as a tar.gz filefile, as requested?**
  Yes

```
ls -1
```

```
all_the_things_I_did
B236494-2023.ICA1.tar.gz
final_full_script_ICA1.sh
```

- **Was your user name visible anywhere?**

```
grep Author *things* | cut -d ' ' -f2,3 | sort | uniq
```

  No, couldn't find it at all: well done!
- **What did the git log suggest to me?**

```
egrep "Date" *things*
```

```
egrep -v "Auth|Date|commit" *things* | awk '(NF>1)'
```

  Dates

```
Date:   Sun Oct 22 19:38:58 2023 +0100
Date:   Sun Oct 22 19:35:25 2023 +0100
Date:   Sun Oct 22 19:21:58 2023 +0100
Date:   Sun Oct 22 18:54:07 2023 +0100
Date:   Sun Oct 22 18:52:34 2023 +0100
Date:   Sun Oct 22 18:47:36 2023 +0100
Date:   Sun Oct 22 18:40:23 2023 +0100
Date:   Sun Oct 22 18:37:14 2023 +0100
```

  Messages

```
    Removed hello.txt
    Made final changes, Script completed
    Added the code for task 5
    Added the code for task 4
    Added the code for task 3
    Added the code for task 1 and task 2
    Initialising the .sh file, makes new dir to run ICA
    This is the first file
```

  Was git actually used during coding at all? It really doesn't look like it, unless you have done all the coding over a space of 60 minutes or so. The messages also should be informative. Doing this properly was an important component of the ICA.

  Like most normal users, I did not read the manual first... to see if the programme is intuitive to use...
- **Could I run the script directly as a script (did it have a shebang line?)?**
  Yes, and it ran to the end.
- **Did the script ask me where I wanted to put files while it was running/when it was finished?**
  No, I never got asked: it assumed I wanted to put it in my homespace, not where I was.
- **Could I run the script directly in my home space (i.e. no links to places I can't get to easily without immediately knowing who you are)?**
  Yes, seemed to find everything.
- **Did the code use the `Tco2.fqfiles` content to set up loops for pretty much all steps?**

```
grep -c Tco2.fqfiles *.sh
```

```
5
```

```
cat *.sh | grep Tco2.fqfiles
```

```
cp ~/folder_to_run_ICA/ICA1/fastq/Tco2.fqfiles .     ###dont forget the . at the end
        # Extracting data from Tco2.fqfiles based on the combination of variables
         result=$(cat Tco2.fqfiles | cut -f 2,4,5 | grep "$clone" | grep "$time" | grep "$state")
cat Tco2.fqfiles | cut -f 1,2,4,5 > all_the_fqfiles.txt      ##putting all the names of Tco2.fqfiles into a txt file for easy use
```

  Yes, `Tco2.fqfiles` was used, but.... not sure why you needed to make all_the_fq_files.txt "for easy use" when you already had the main file? Was that hard to use?! this would NOT work with the full dataset with samples from other groups because the `grep` would not put them in any list. This would defeat the whole purpose of the experiment!

```
# All the details were provided in a structured format, seven fields per line
```

```
head -n 3 Tco2.fqfiles
```

```
SampleName      SampleType      Replicate       Time    Treatment       End1    End2
Tco230  Clone1  1       0       Uninduced       Tco-230_1.fq.gz Tco-230_2.fq.gz
Tco935  Clone1  1       24      Induced Tco-935_1.fq.gz Tco-935_2.fq.gz
```

  These can be extracted as variables, as the file is parsed, line by line, using a `while` loop something like this

```
while read SampleName SampleType Replicate Time Treatment End1 End2
do
echo "Processing sample ${SampleName}..."
# do cool stuff here
done < ${my_Tco_detailsfile}
```

  loop.

You could use this loop numerous times to do different things, that is absolutely fine!
For example, the sample group for each sample can be automatically derived by combining fields (variables) obtained from the relevant fields (columns) of `Tco2.fqfiles` which in turn could be used to generate file names, make folders, in fact, pretty much anything/everything you needed.

We used a very similar sort of approach when we processed our BLAST outputs in Lecture07.

- **Could I have run the script if there had been additional samples?**
  Yes, it coped with the extra samples, but not the new groups: use the `Tco2.fqfiles` sample details file more carefully!
- **Did the code have lots of comments so I could see what the code was doing?**
  Yes, sufficient, a few more would have been OK
- **Was it logically laid out?**
  Yes
- **Did it use variables, or were most things hard-coded?**
  Used variables but some critical things (sample groups/sequence name parts) were "hard-coded" too?!
- **Did it use loops, or were most things hard-coded?**
  Used loops
- **Did it use things that we hadn't covered in the course, like bash functions?**
  Yes, quite a bit of awk arrays.
- **Was there evidence that a flow-chart or similar had been used in planning the design (i.e. output from one bit being the input for the next)?**
  Yes, clearly planned in advance
- **Were the replicates kept separate?**
  Yes, they are replicates! We might need them to do statistical analyses later on...

Were the commands correct?

- **initial processing stages**
  DIDN'T use info file `Tco2.fqfiles` ...
  You do NOT need to copy ANY files...
  You do NOT need to unzip ANY files....
- **fastqc**
  `fastqc` accepts wild cards, and can be multi-threaded, so could/should have run `fastqc` on all samples with one command: much faster and more efficient: you did this, but with only 1 thread...
  `fastqc` also works on gzipped files, so there is no need to unzip OR move the fastq files.
  `fastqc` can be run with the `--extract` parameter so that you don't need to unzip the outputs
  A summary table of all samples would be very useful to look at, and saved as a record for the future? If one was generated, I wasn't told when it was running. I dont think any of the fastqc output was looked at??
  I didn't see any code for processing the `fastqc` outputs (or automatically running `firefox` to display the outputs), so I didn't know what I had got (how many reads, etc!): were the samples any good at all?!
  How about running `firefox` to display (some of) the outputs?
- **Were there problems using `firefox` (if used)?**
  User not shown any html outputs
- **Was the user given an option to quit after viewing the QC**
  No, analysis continued irrespective of how "good" or "bad" the data were.
  We did learn how to get user input in the lectures:

```
# Maybe we could ask the user what country to process?!
read -p "What country shall we process the data for? " wantedcountry
What country shall we process the data with?
```

- **bowtie2 or hisat2 alignment**
  One could have used the `Tco2.fqfiles` for this step too...
  Were any of the samples poor quality in terms of alignments.... if so, which one was it...? It was Tco-17 that was REALLY bad, and Tco-580 was probably marginal/worriesome too. I didnt see any assessment of the alignments: the code carried on regardless...
  Genome index building: reference does not need to be unzipped, but if it was, it should be gzipped up again after (or just deleted locally, if you copied it)
  Would probably recommend you use MANY more than 1 thread for genome indexing and/or the alignment steps; `nproc` will show you how many we have on our server... and it is MUCH faster! Remember the big fruit salad problem?!
  Use a pipe so that the alignment output is made directly into bam format, that is then sorted and indexed using `samtools`, without having to save the intermediate SAM format (increased speed, decreased diskspace)
  You made life more tricky for yourself by putting each sample into a new folder... instead of perhaps adding the sample outputs to a sample group folder...?? You wouldnt have needed that nested for loop structure over all combinations of clone, time, and state, AND it would not have had to be hard coded either.
- **bedtools**
  this was essentially hard-coded...so won't work for the anything other than the initial dataset
  possibly incorrect parameter used with `bedtools` which made your analyses afterwards much harder
  `bedtools multicov *.bam` will put all the counts data in one output file in bamfile name order, which makes the subsequent processing a lot easier, and means `bedtools` is only run once (it's not very fast)!
  You chose to do it sample by sample rather than extract the relevant columns from a single output file, so ran bedtools many times. That's fine, just a different way, not sure which might be fastest/most flexible?

  The sample groups can be "built" using the information in `Tco2.fqfiles`, so that can be used to direct where output goes too, rather than "getting things back" later.
- **final processing stages**
  There were 15 different experimental groups in this dataset, with differing numbers of replicates. What they were and how many of each you could work out directly from the `Tco2.fqfiles` sample details file. As we did get more samples, the code below will still work perfectly well, but will produce different groups and replicate numbers of course!

```
while read SampleName SampleType Replicate Time Treatment End1 End2
 do
 echo ${SampleType}.${Time}.${Treatment}
done < ${my_Tco_detailsfile} | tail -n +2 | sort | uniq -c > reps.groups
```

```
cat reps.groups
```

```
      3 Clone1.0.Uninduced
      3 Clone1.24.Induced
      3 Clone1.24.Uninduced
      4 Clone1.48.Induced
      3 Clone1.48.Uninduced
      3 Clone2.0.Uninduced
      3 Clone2.24.Induced
      4 Clone2.24.Uninduced
      3 Clone2.48.Induced
      3 Clone2.48.Uninduced
      3 WT.0.Uninduced
      4 WT.24.Induced
      3 WT.24.Uninduced
      3 WT.48.Induced
      3 WT.48.Uninduced
```

For calculating averages, we already know that bash isn't good at doing maths, but `awk` definitely is. If your output file contained the bed file details, and then three columns of gene expression values (for example, but you could get these values from the reps.groups file above...) saved in a file called `Clone1.0.Uninduced.genecounts`, then you could use something like:

```
while read reps groups
```

```
do
 # bring in the variables so that awk can use them
 awk -v reps=${reps} -v groups=${groups} '{
 sum=0;
 # start at the relevant column and sum to the end of each row
 for (i=NF-reps+1; i<=NF; i++) sum+=$i;
 print sum/reps > groups".averages"
                }' ${groups}.genecounts ;
done < reps.groups
```

```
head -3 Clone1.0.Uninduced.genecounts
```

```
12 14 30
17 24 10
200 124 170
```

```
head -3 Clone1.0.Uninduced.averages
```

```
18.6667
17
164.667
```

The output files your code generates are not the mean expressions for each GENE, but for the sample as a whole. This was not what was asked for!
I think that you may have realised that when you put this line in the code?

```
echo "The gene descriptions were not added"
```

Were there any fold-change calculations?
Your code:
Looks like it produces the wrong output, and, as you have probably already realised, would not work for the full dataset either.
The ICA said: "the statistical mean (average) of the counts per gene for each group", but not all groups in the full set would be represented! What will happen when the new samples come, and they might be from clone3....??
If one of a pair of comparisons has 0 counts for a gene, that doesn't mean fold change will be zero. It might be infinite, so it is commonplace to add a very small value to make things non-zero, thereby preventing the division by/with zero issue.
You were not asked to do things in log2, but if you are interested, it's quite simple to do in awk: https://stackoverflow.com/questions/13687657/awk-and-log2-divisions

- **Were there any progress indicators?**
Not really, maybe only 1 or 2, it was mainly what the programmes themselves output. I didn't know if the programme was still working, or what it was doing. Some progress indicators are ALWAYS good to reassure the user that things are working!
`echo` is your friend.

- **Was there any two-way interaction with the user?**
No. Possibilities were for user to specify where the output might go, after QC (continue or not?), choice of `bowtie2` or `hisat2` etc.
Could even ask the user what comparisons they might have wanted? There's **105** possible pairwise combinations of 15 groups...

PDF comments

This bit was even more open-ended: this was for me to try and get an idea of "how" you are thinking about the problem, and any things that might need to be clarified to you and/or others.

Were these requests for the PDF answered?

1. **gives the link for your GitHub ICA1 repository (https://github.com/Bxxxxxx-2023/ICA1)**
   Yes
2. **gives the correct password to open the *Bxxxxxx*-2023.ICA1.tar.gz.cpt file ...!**
   Yes
3. **has an overview/flowchart that describes how the pipeline processes the data**
   Yes, but not enough detail really: what were the inputs and what were the outputs?
4. **briefly indicates why you have chosen the parameters/flags that you have for each step**
   No, nothing much put here to indicate anything other than default parameters being used (and in this case, they are NOT the best, remember the big fruit salad situation...?!). One of the key ones to change was to add threads to all of the programme steps, they would run much quicker then!
5. **indicates any things the user will have to do to make things work**
   No instructions, but the user had to do things! Not even the name of the programme that needed to be run...!
6. **highlights any difficulties, if any, that you have come across**
   Yes, thanks for being honest.
   You mentioned all the things that need to be there, and for a few of them, they WERE there, and for others, I think you just need to look at using the `Tco2.fqfiles` approach to its full potential (rather than just using parts of it/information in it), which, as noted above, would be along the lines of:

```
while read SampleName SampleType Replicate Time Treatment End1 End2
  do
# cool analysis stuff here
  done < Tco2.fqfiles
```

7. **indicates any additional/alternative features that you think might be beneficial to include**
   None suggested.

Any final comments

There were a few things missing or incomplete, as you yourself note, and which I have commented on above. DO look at the FAQ, as there were many useful clues there, as I am not convinced that you actually understood the question fully... I hope you find this feedback helpful and constructive: more than happy to chat about anything that would help!
Cheers
al

Mark awarded: 60