Student B number: **B217754**

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-2022.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-2022.ICA1.pdf, submitted via Learn Dropbox, that:
   - gives the link for your PUBLIC GitHub ICA1 repository (e.g. https://github.com/Bxxxxxx-2022/ICA1)
   - gives the `ccrypt` encryption key to open the Bxxxxxx-2022.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:   Mon Oct 24 12:37:58 2022 +0100
  ```
- **Could I decrypt the file with the password given?**
  Yes, but incorrect password had been provided initially. The tar.gz file wouldn't decrypt with the correct password?

  ccrypt -d B217754-2022.ICA1.tar.gz.cpt

  ```
  Enter decryption key:
  ccrypt: B217754-2022.ICA1.tar.gz.cpt: key does not match -- unchanged
  ```
- **Did the GitHub repository only include the pipeline code as a tar.gz filefile, as requested?**
  Yes

  ls -1

  ```
  all_the_things_I_did
  B217754-2022.ICA1.tar.gz
  ICA1_SCRIPT_B217754.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* | awk '{if(NF>1){print $2,$3,$4}}' | uniq -c
  egrep -v "Auth|Date|commit" *things* | awk '(NF>1)'
  Dates

  ```
  2 Mon Oct 24
  2 Sat Oct 22
  1 Wed Oct 19
  1 Tue Oct 18
  1 Mon Oct 17
  1 Fri Oct 14
  ```

  Messages

```
Final dry-run, arranging git log and debugging
Formalising sections of the script, modules 1-3 are working. Preparing module 5 now
forgot to add current draft script
Extensive testing of the first 'module' used to generate FASTQC and extract the qualtiy messages. Not much luck but close
Fleshed out quality warning script
Ran fastqc on the zip files, extracted to a FASTQC_OUTPUT directory, removed .html files
Downloaded the ICA materials from the server
```

Coding was done over multiple sessions, good use of messages.

- **Could I run the script directly as a script (did it have a shebang line?)?**
  No, it fell over, and I tried to fix it, but it was going to be a major job, so I gave up, sorry...

ALL THE FOLLOWING COMMENTS ARE BASED ON LOOKING AT THE CODE, NOT RUNNING IT

```
Welcome aivens2......
......Copying the ICA data and Pair-ended RNAseq sequence data from the server to the present working directory
#################1####################
Making folder for use by FastQC
Running FastQC on pair-ended data held in this directory and removing .html output files
#####################################
Making folder for quality warnings to be held, unzipping the fastqc.zip output files

96 archives were successfully processed.
ICA1_SCRIPT_B217754.sh: 37: cannot open reads_quality_warnings.txt: No such file
0 reads_quality_warnings.txt
#####################################
Quality warnings were identified by fastqc and were appended to the file ./RNAi_experiment_data/reads_quality_warnings.txt
There were  quality flags in the categories: overrepresented sequences, per sequence quality score and per base sequence quality
#####################################
#################2####################
gzip: ./Tcongo_genome/TriTrypDB-46_TcongolenseIL3000_2019_Genome.fasta.gz: No such file or directory
mkdir: cannot create directory './RNAi_experiment_data/bowtie2': No such file or directory
ICA1_SCRIPT_B217754.sh: 60: cd: can't cd to ./RNAi_experiment_data/bowtie2
Unzipping the reference genome for bowtie2, making directories for the output data
Settings:
  Output files: "bowtie.*.bt2"
  Line rate: 6 (line is 64 bytes)
  Lines per side: 1 (side is 64 bytes)
  Offset rate: 4 (one in 16)
  FTable chars: 10
  Strings: unpacked
  Max bucket size: default
  Max bucket size, sqrt multiplier: default
  Max bucket size, len divisor: 4
  Difference-cover sample period: 1024
  Endianness: little
  Actual local endianness: little
  Sanity checking: disabled
  Assertions: disabled
  Random seed: 0
  Sizeofs: void*:8, int:4, long:8, size_t:8
Input files DNA, FASTA:
  ../../Tcongo_genome/TriTrypDB-46_TcongolenseIL3000_2019_Genome.fasta
Error: could not open ../../Tcongo_genome/TriTrypDB-46_TcongolenseIL3000_2019_Genome.fasta
Total time for call to driver() for forward index: 00:00:00
Error: Encountered internal Bowtie 2 exception (#1)
Command: /usr/local/bin/bowtie2-build-s --wrapper basic-0 ../../Tcongo_genome/TriTrypDB-46_TcongolenseIL3000_2019_Genome.fasta bowtie
ICA1_SCRIPT_B217754.sh: 64: cd: can't cd to ../../fastq
Indexing the reference genome for use by bowtie2
ICA1_SCRIPT_B217754.sh: 71: Bad substitution
```

Dont read in to `done`, as you have already defined your variable `${x}` and its trying to read in the output at the same time!

```
for x in ./  #FASTQC
do
...
grep -v "PASS" ${x}*fastqc/summary.txt >> ../reads_quality_warnings.txt
done  <  reads_quality_warnings.txt # remove this highlighted bit
```

This won't work as a variable as there are spaces and no $ sign, () etc

`warning_count= wc -l reads_quality_warnings.txt`

I dont think this worked, should only be 1 up, not two, so it couldnt find the genome folder, then the rest didnt work...

`cd ../.. #ICA1`

This would have probably been fine if the folders had been defined as variables, e.g.

```
experimentplace="$HOME/ICA1/RNAi_experiment"
qcplace="${experimentplace}/FASTQC"
groupsplace="${experimentplace}/groupstuff"
mkdir -p ${experimentplace}
mkdir -p ${qcplace}
mkdir -p ${groupsplace}
fastqc -t 48 --outdir ${qcplace} -extract ${experimentplace}/*.fq.gz
```

- **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 where I was running it from, which was fine this time.

- **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 `Tco.fqfiles` content to set up loops for pretty much all steps?**

  grep -c Tco.fqfiles *.sh

  ```
  8
  ```

  cat *.sh | grep Tco.fqfiles

  ```
  }' Tco.fqfiles >> ../WTz.txt
  }' Tco.fqfiles >> ../Clone1z.txt
  }' Tco.fqfiles >> ../Clone2z.txt
  i}' Tco.fqfiles >> ../T0z.txt
  }' Tco.fqfiles >> ../T24z.txt
  }' Tco.fqfiles >> ../T48z.txt
  }' Tco.fqfiles >> ../Inducedz.txt
  }' Tco.fqfiles >> ../Uninducedz.txt
  ```

  Yes, `Tco.fqfiles` was used, but... not in "right" way... this would NOT work with the full dataset with samples from other groups and wasnt really how to do it, I could see that you knew that, just didnt know what was the right thing!

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

  head -n 3 Tco.fqfiles

  ```
  SampleName      SampleType      Replicate       Time    Treatment       End1    End2
  Tco5053 Clone2  3       0       Uninduced       Tco-5053_1.fq.gz        Tco-5053_2.fq.gz
  Tco5110 Clone1  3       0       Uninduced       Tco-5110_1.fq.gz        Tco-5110_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 `Tco.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?**
  No, sadly, wasn't sufficiently generic: use the `Tco.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 perhaps then looked at the contents, rather than just deciding on what could be done WITH the contents, e.g. what was in each column, what it meant, but some critical things (sample groups/sequence name parts) were "hard-coded" too?!
  ...so you made yourself a HUGE amount of work!
- **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?**
  No, but that doesn't matter here, I was just curious.
- **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 really use info file `Tco.fqfiles`...
  You used variables as references to places/files: good!
  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 it was good to see `fastqc` was run on all samples with one command: much faster and more efficient, but could have given it more threads...
  `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 was useful to look at (if it had been made)
  I saw processing the `fastqc` outputs but I didn't get a chance to look at them as the programme kept on running: I do not 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?
- **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**
  Why didn't you use the `Tco.fqfiles` for this step too?!
  It looks like each sample was aligned, but the output kept on being put in the same file name, overwriting it each time, so you wouldnt be able to do ANY of the subsequent steps like comparing groups. The command was correct, but you need a sam file (or bam file) for EVERY sample.
  Were any of the samples poor quality in terms of alignments.... if so, which one was it...? It was Tco5934 that was REALLY bad, and Tco6114 was probably marginal/worriesome too.
  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; `htop` will show you how many we have on our server... and it is MUCH faster! Remember the big fruit salad problem?! You used lots for the bowtie2 alignment step
  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)
- **bedtools**
  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 file name order, as you did, which makes the subsequent processing a lot easier, and means `bedtools` is only run once (it's not very fast)!
  `bedtools multicov *.bam` uses the `*` wildcard, so there is no need to list individual bamfiles, which of course meant that the script couldn't cope with a different set of samples with different set of file names.

  The sample groups can be "built" using the information in `Tco.fqfiles`, so that can be used to direct where output goes too.
- **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 `Tco.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
```

Your code:
Didnt get this far unfortunately.
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?**
Yes, lots, thanks, but a lot of them were AFTER what was being done, probs best if they go before the relevant steps that would be more indicative!
- **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-2022/ICA1)**
Yes
2. **gives the correct password to open the *Bxxxxxx*-2022.ICA1.tar.gz.cpt file ...!**
No, but provided later when asked
3. **has an overview/flowchart that describes how the pipeline processes the data**
Yes, an overview of coding lines (but the code didnt have line numbers) was given, but not enough detail really: what were the inputs and what were the outputs? Even a basic flow diagram would have been good I think?
4. **briefly indicates why you have chosen the parameters/flags that you have for each step**
Yes, info in the code too.
5. **indicates any things the user will have to do to make things work**
No instructions, but it was fairly obvious what was required

6. **highlights any difficulties, if any, that you have come across**
   Yes, thanks for being honest, and I can see how you were trying to do it, can see your thought patterns!
   Main thing is using the `Tco.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 < Tco.fqfiles
```

7. **indicates any additional/alternative features that you think might be beneficial to include**
   Yes, some good suggestions, you have clearly thought about it.

## Any final comments

There were unfortunately quite a few things missing or incomplete, which I have commented on above. I think most of it would have run apart from the "wrong directory" problem, so credit to you for that, but I'm sorry, can't really give credit for the latter stages, as they just wouldnt work at all as their inputs was only 1 sample!
I hope you find this feedback helpful and constructive; once you have read it, feel free to come and talk to me, happy to go through it with you to make sure you know how to do it.
I have waived the "wrong password" penalty, but do get it correct next time please!
Cheers
al

## Mark awarded: 57