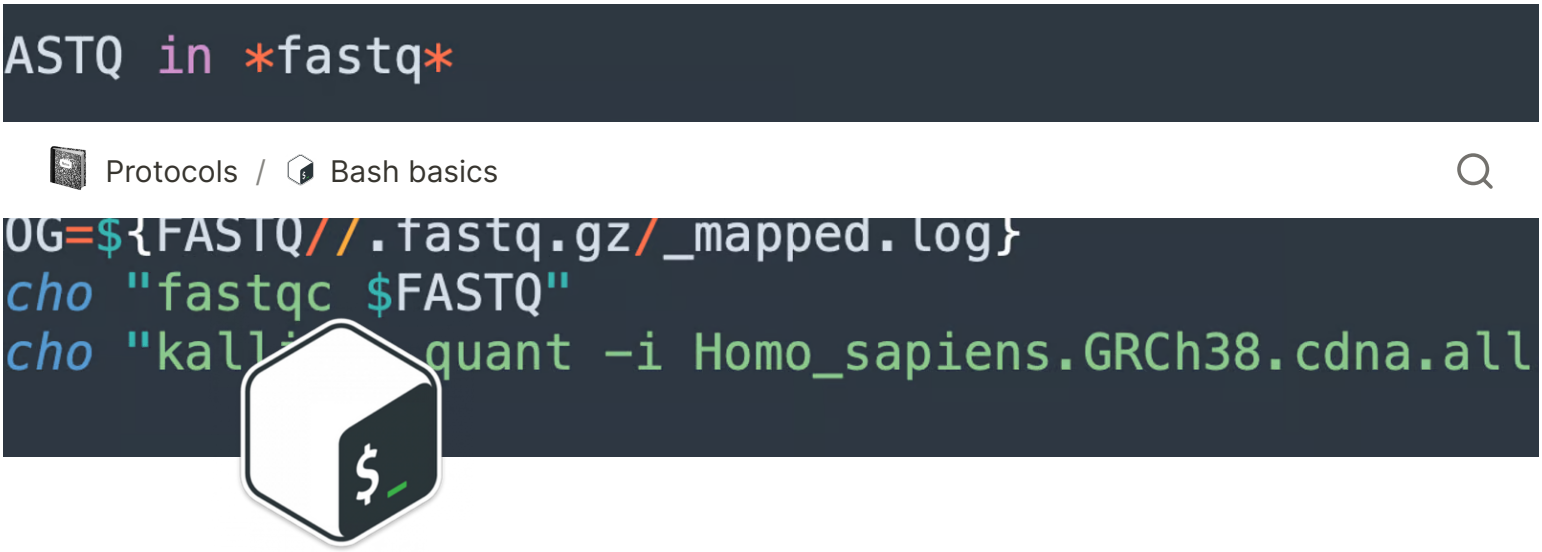


Bash basics

This document accompanies the first lab exercise from the DIYtranscriptomics course.

Helpful command line tips

Understanding file permissions

Getting to know Bash parameters

Getting to know 'for loops'

Using a loop to automate QC and alignment of multiple fastq files

Some other useful loops for aligning reads using other software tools

Helpful command line tips

If you're new to Bash, there are lots of online resources for learning, but here are a few of the commands that will help you move around and carry out basic tasks. Note that some of these commands may

only work if run as `sudo` .

Note: for Windows OS, you will need to install either Windows Subsystem for Linux (WSL) or Git for Windows. I recommend WSL, but if you choose Git for Windows, you'll need to add C:\Program Files\Git\usr\bin to your system variables path for all of the common bash commands to work

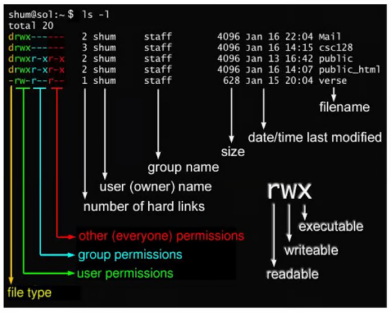
Common bash commands

typing this (if you're on a Mac)	or this (if you run Windows)
`cd /`	`cd /`
`cd ~`	no direct shortcut
`cd ..`	`cd ..`
`cd ../../`	`cd ../../`
`cd path/to/some/folder`	`cd path/to/some/folder`
`tar -xvzf [fileName.tar.gz]`	`tar -xvzf [fileName.tar.gz]`
`gzip [filename]`	`gzip [filename]`
`ls` and `ls -l` and `ls -a`	`ls` and `ls -l` and `ls -a` or
`ls -l | wc -l`	`ls -l | wc -l`
`mv [fileName or folderName] [directory]`	`mv [fileName or folderName] [directory]`
`du -a -h | sort -hr`	no direct shortcut
`du -sh *`	`du -sh *`
`df -h`	`df -h`
`tree -d`	`tree -d`
`lsblk`	no direct shortcut
pressing up arrow	pressing up arrow

`chmod ### [fileName]`	`chmod ### [fileName]`
`chown [yourUserName] [fileName]`	`chown [yourUserName] [fileName]`
`chgrp [yourUserName] [fileName]`	`chgrp [yourUserName] [fileName]`
`rm -rf [directoryName]`	`rm -rf [directoryName]`
`wget [URLtoFile]`	no direct shortcut
`nano [file.txt]`	`nano [file.txt]`
`export PATH="/path/to/your/software/:$PATH"`	no direct shortcut
`alias something="something else"`	`doskey something=something else`

Understanding file permissions

If you're trying to do something to a file or folder in Bash (e.g., delete, move, edit, create) and are unable to, chances are you need to modify the permissions (`chmod`), the owner (`chown`) or the group (`chgrp`). See below and read more about permissions here



Octal	Decimal	Permission	Representation
000	0 (0+0+0)	No Permission	---
001	1 (0+0+1)	Execute	--x
010	2 (0+2+0)	Write	-w-
011	3 (0+2+1)	Write + Execute	-wx
100	4 (4+0+0)	Read	r--
101	5 (4+0+1)	Read + Execute	r-x
110	6 (4+2+0)	Read + Write	rw-
111	7 (4+2+1)	Read + Write + Execute	rwx

Getting to know Bash parameters

Create your first parameter.

```
DNA="ATCG"
```

Use the echo program to see what you created.

```
echo $DNA
echo $DNA are my favorite bases
```

Try closing and relaunching your Bash application window. Notice that your 'dna' variable no longer exists, because it is a 'shell variable', which means it is contained exclusively within the shell in which it was set or defined. You can read more about variable types here.

Before moving on, go ahead an re-create your dna variable.

We forgot about poor uracil...what kind of RNA-seq class is this?! Let's add it to our variable now.

```
echo $DNAU #note that this didn't work!
```

We can fix this by taking advantage of parameter expansion using curly brackets {}

```
echo ${DNA}U
```

Getting to know 'for loops'

For loops allow you to iterate over a list of items (in our case, files in a folder) and carry out any number of actions on those files. Here's the general format of a for loop.

```
# don't try running this code
for item in [LIST]
do
  [COMMANDS]
done
```

Now we'll create a simple for loop that actually runs

```
for i in TACG CTAG CCTC GAAT
do
    echo "$i is a DNA oligonucleotide"
done
```

Let's apply the concept of parameter expansion to our new loop in order to easily find/replace 'T's with 'U's. Two things to take note of here. First, we're declaring a new parameter within the loop, and it will only exist when the loop is running. Second, we have taken advantage of some handy syntax that lets us find (//) and replace (/) parts of our original parameter.

```
for i in TACG CTAG CCTC GAAT
do
    FIX=${i//T/U}
    echo "$FIX is a RNA oligonucleotide"
done
```

Using a loop to automate QC and alignment of multiple fastq files

Now we put this all together to make the creation of shell script really simple.

```
for FASTQ in *fastq*
do
    OUT=${FASTQ//.fastq.gz/_mapped}
    LOG=${FASTQ//.fastq.gz/_mapped.log}
    echo "fastqc $FASTQ"
    echo "kallisto quant -i Homo_sapiens.GRCh38.cdna.all.i
    echo "done mapping reads for $FASTQ"
done
```

❓ Now that's all great, but we don't really want the output of this loop to print to our terminal screen (a.k.a. STDOUT). Instead we'd like to save it to a shell script so that we have a record of what was done. Let's do that now using the `>` operator

❓ How would you incorporate MulitQC into this loop?

❓ What do you think would happen if you removed `echo` and the `" "` enclosing the fastqc and kallisto commands?

❓ Since a single loop could kick off a very long running compute job, you'll want to know how to use the `screen`

program. I'll demonstrate for you.

⚠️ If you want to test out this code on real data, use the subsampled fastq files (only 10,000 reads in each file) described in the lab exercise associated with this lesson. Just keep in mind that you'll need to have the human reference transcriptome index available in your working directory.

For loops can be combined with conditional statements (if/then) to loop over only certain files in a directory

```
for FASTQ in *fastq*
if [[ $FASTQ == *subsample* ]];
then
  echo "fastqc $FASTQ"
else
  echo "$FASTQ was not processed"
fi
```

Some other useful loops for aligning reads using other software tools

Bowtie2

```
for READ1 in *fastq*
do
    READ2=${READ1//1.fastq.gz/2.fastq.gz}
    SAM=${READ1//_1.fastq.gz/.sam}
    BAM1=${READ1//_1.fastq.gz/_mapped_and_unmapped.bam}
```

```
    BAM2=${BAM1//_mapped_and_unmapped.bam/_unmapped.bam}
    BAM3=${BAM2//_unmapped.bam/_umapped_sorted.bam}
    FQ1=${BAM3//_umapped_sorted.bam/_dehosted_R1.fastq}
    FQ2=${BAM3//_umapped_sorted.bam/_dehosted_R2.fastq}
    bowtie2 -x /venice/reference_db/Homo_sapiens/ensembl_r
    samtools view -@ 24 -bS $SAM > $BAM1
    samtools view -@ 24 -b -f 12 -F 256 $BAM1 > $BAM2
    samtools sort -@ 24 -n $BAM2 -o $BAM3
    bedtools bamtofastq -i $BAM3 -fq $FQ1 -fq2 $FQ2
done
```