Automating the analysis path from sequence reads to count matrix

Lingzi Li

August 7, 2019

Some shell scripting rules

1) The first line in your script must be:

```
#!/bin/bash
```

This line lets the environment know the file is a shell script and the location of the shell.

2) The name of your shell script must end with a *.sh*

3) Before executing your script, you should make the script executable by using the following command:

```
chmod +x your_shell_script.sh
```

4) To run the .sh file:

```
# Any of the following commands can work
./your_shell_script.sh

sh your_shell_script.sh

bash your_shell_script.sh
```

Important concepts

Variable

Assign any name and value using the assignment operator: '='. You can check the current definition of your variable by typing into your script: echo $variable_name

Example:

```
# Assign the name SM4393_R1.fq.gz to a variable called fq
fq=SM4393_R1.fq.gz

# Check the definition of the variable fq
echo $fq
```

basename

To ensure that all the output files from the workflow are properly named with sample IDs, we need to extract the "base name" (or sample ID) from the name of the input file.

Example:

```
fq=SM4393_R1.fq.gz

# Grab base of filename for naming outputs
basename $fq _R1.fq.gz
```

```
# This syntax is necessary for assigning the output of a command to a variable.
id=`basename $fq _R1.fq.gz`

# OR this also works
id=$(basename $fq _R1.fq.gz)

# Check the definition of the variable id
echo $id
```

read

The read command is used to get a line of input into a variable.

Example: **demo_read.sh**

```
#!/bin/bash
read -p "Please enter one word to describe today's weather: " response
echo "The weather today is $response"

echo "AGAIN: Please enter one word to describe today's weather: "
read response
echo "The weather today is $response"
```

Positional parameters

Example: **demo_interactive.sh**

```
#!/bin/bash
# USAGE: bash demo_interactive.sh <1st_parameter> <2nd_parameter>

echo "How many parameters this script is called with? $# parameters"
echo "The script name is $0"
echo "The first parameter is $1"
echo "The second parameter is $2"
echo "All parameters are $*"
echo "AGAIN! All parameters are $@"
```

$# - Number of parameters specified in the command line.

$0 - Name of the shell script being executed.

$* or $@ - This denotes all the parameters passed to the script at the time of its execution.

for loop

Syntax:

```
for variable in list
do
execute commands
done
```

Example:

```
# Get stat from BAM files
for bamFile in *_Aligned.sortedByCoord.out.bam
do
  baseFilename=`basename $bamFile _Aligned.sortedByCoord.out.bam`
  echo "***Now extract statistics from alignment of ${baseFilename}***"
```

```
  samtools flagstat -@ 16 $bamFile >${baseFilename}.stat.txt
done
```

while loop

Syntax:

```
while condition_is_true
do
execute commands
done
```

Example:

```
# Get stat from BAM files
ls *_Aligned.sortedByCoord.out.bam | while read bamFile
do
  baseFilename=`basename $bamFile _Aligned.sortedByCoord.out.bam`
  echo "***Now extract statistics from alignment of ${baseFilename}***"
  samtools flagstat -@ 16 $bamFile >${baseFilename}.stat.txt
done
```