Bedtools Intersect All Script

Notes

0.1. DO NOT copy paste any of this document into your terminal, it won't work

0.2. In bash, spaces MATTER! You will get errors if you don't add appropriate spaces

0.3. In bash, indentations DO NOT matter, however adding these helps keep code organized

0.4. For all code below: things to write/update in red, comments in green

1 User Input and for-loops

1.1. Open a new file called intersect_all.sh, in that file:

1.2. Create a variable for a user input directory and full path to output file

```
input_directory=$1
output_file=$2
```

1.3. Create a for loop to print all filenames in user inputted directory.

```
input_directory=$1
output_file=$2

for file in `ls $input_directory`; do #NOT single quotes, special
    echo $file                         #character on same key as ~
done
```

1.4. Change filename to be the full path to file

```
input_directory=$1
output_file=$2

for file in `ls $input_directory`; do
    file=${input_directory}/${file} #Now we require that the input
    echo $file                      #directory NOT contain a trailing "/"
done
```

1.5. Execute the script

```
sh ./intersect_all.sh /full/path/to/your/bed ./
```

2 If statements

2.1. Add an if statement to only consider files with the .bed extension in the input directory

```
input_directory=$1
output_file=$2

for file in `ls $input_directory`; do
    if [[ $file == *".bed" ]]; then #Those are two equal signs in a row
       file=${input_directory}/${file}
       echo $file
    fi
done
```

3 Think

3.1. Bedtools intersect requires two files to intersect

→ First file we loop through is a special case

3.2. Intersection needs to be done serially (ie. intersect a with b, the result of that gets intersected with c, etc.)

→ The output of previous command will feed into next command

4 Create a counter to detect first file

4.1. Create a counter, add an additional if-statement to detect when the counter is at 0. Make sure you increment your counter after every loop!

```
input_directory=$1
output_file=$2
temp=${input_directory}/temp.txt
i=0

for file in `ls $input_directory`; do
    if [[ $file == *".bed" ]]; then
       file=${input_directory}/${file}
       if [[ $i == 0 ]]; then
           echo The first file is $file
       else
           echo $file
       fi
       i=1 #Becomes 1 only after first bed file encountered
    fi
done
```

5 Setup commands for initializing and looping

5.1. Use cat to initialize the output file if it's the first file encountered

```
input_directory=$1
output_file=$2
i=0

for file in `ls $input_directory`; do
    if [[ $file == *".bed" ]]; then
       file=${input_directory}/${file}
       if [[ $i == 0 ]]; then
           cat $file > $output_file
       else
           echo $file
       fi
       i=1
    fi
done
```

5.2. Intersect output file with next file if not the first file encountered

```
input_directory=$1
output_file=$2
i=0

for file in `ls $input_directory`; do
    if [[ $file == *".bed" ]]; then
       file=${input_directory}/${file}
       if [[ $i == 0 ]]; then
           cat $file > $output_file
       else
           bedtools intersect -a $output_file -b $file > $output_file
       fi
       i=1
    fi
done
```

6 Debug: Output file is empty!

6.1. Use echo command to print all variables

→ Everything looks fine

6.2. Use echo to print actual commands, execute them in your terminal one by one

→ When redirecting stdout (" >"), an empty output file is created first!

→ → Solution: use a temporary file to store results as you go

7 Implement temporary file

7.1. Within the user inputted directory, create a temporary file (note we won't give this file a .bed extension so we don't confuse our script later. Bedtools will still work on a .txt file). Remove it at the end.

```
input_directory=$1
output_file=$2
temp=${input_directory}/temp.txt
i=0

for file in `ls $input_directory`; do
    if [[ $file == *".bed" ]]; then
        file=${input_directory}/${file}
        if [[ $i == 0 ]]; then
            cat $file > $output_file
        else
            bedtools intersect -a $output_file -b $file > $output_file
        fi
        i=1
    fi
done
rm $temp
```

7.2. Initialize temp file

```
input_directory=$1
output_file=$2
temp=${input_directory}/temp.txt
i=0

for file in `ls $input_directory`; do
    if [[ $file == *".bed" ]]; then
        file=${input_directory}/${file}
        if [[ $i == 0 ]]; then
            cat $file > $temp
        else
            bedtools intersect -a $output_file -b $file > $output_file
        fi
        i=1
    fi
done
rm $temp
```

7.3. Fix bedtools intersect command

```
input_directory=$1
output_file=$2
temp=${input_directory}/temp.txt
```

```
i=0

for file in `ls $input_directory`; do
    if [[ $file == *".bed" ]]; then
       file=${input_directory}/${file}
       if [[ $i == 0 ]]; then
           cat $file > $temp
       else
           bedtools intersect -a $temp -b $file > $output_file
       fi
       i=1
    fi
done
rm $temp
```

7.4. Update temp file

```
input_directory=$1
output_file=$2
temp=${input_directory}/temp.txt
i=0

for file in `ls $input_directory`; do
    if [[ $file == *".bed" ]]; then
       file=${input_directory}/${file}
       if [[ $i == 0 ]]; then
           cat $file > $temp
       else
           bedtools intersect -a $temp -b $file > $output_file
           cat $output_file > $temp
       fi
       i=1
    fi
done
rm $temp
```

Warning: If you are getting an empty file at this point, make sure you don't have an old output file in your input directory

8 Optional: Handle unsorted bed files with pipes

8.1. Change cat command to a bedtools sort

```
input_directory=$1
output_file=$2
temp=${input_directory}/temp.txt
i=0
```

```
for file in `ls $input_directory`; do
    if [[ $file == *".bed" ]]; then
       file=${input_directory}/${file}
       if [[ $i == 0 ]]; then
           bedtools sort -i $file > $temp
       else
           bedtools intersect -a $temp -b $file > $output_file
           cat $output_file > $temp
       fi
       i=1
    fi
done
rm $temp
```

8.2. Add a sort command and pipe it to intersect command

```
input_directory=$1
output_file=$2
temp=${input_directory}/temp.txt
i=0

for file in `ls $input_directory`; do
    if [[ $file == *".bed" ]]; then
       file=${input_directory}/${file}
       if [[ $i == 0 ]]; then
           bedtools sort -i $file > $temp
       else
           bedtools sort -i $file | \
           bedtools intersect -a $temp -b stdin > $output_file
           # "\" lets us continue the code on the next line
           # "|" is a pipe character
           # "stdin" is a keyword that bedtools uses to retreive from pipe
           cat $output_file > $temp
       fi
       i=1
    fi
done
rm $temp
```

9 Optional: Make your script executable from anywhere

9.1. Add the bash she-bang so your system knows explicitly that this is a bash script

```
#!/bin/bash
input_directory=$1
```

```
output_file=$2
temp=${input_directory}/temp.txt
i=0

for file in `ls $input_directory`; do
    if [[ $file == *".bed" ]]; then
       file=${input_directory}/${file}
       if [[ $i == 0 ]]; then
           bedtools sort -i $file > $temp
       else
           bedtools sort -i $file | \
           bedtools intersect -a $temp -b stdin > $output_file
           cat $output_file > $temp
       fi
       i=1
    fi
done
rm $temp
```

9.2. Remove .sh extension for aesthetics

```
mv intersect_all.sh ./intersect_all
```

9.3. Make your script executable

```
chmod u+x intersect_all
```

9.4. Copy script to /usr/local/bin or add the script location to your path

```
scp intersect_all /usr/local/bin
#OR in your ~/.bashrc file add:
export PATH=$PATH:/full/path/to/intersect_all
```