

MG_HW6: Gene Calls with Prodigal version 3

James Thornton

Abstract

This protocol provides the procedure to generate gene calls on your contigs using Prodigal.

Citation: James Thornton MG_HW6: Gene Calls with Prodigal. **protocols.io**

dx.doi.org/10.17504/protocols.io.f47bqzn

Published: 17 Oct 2016

Protocol

Step 1.

Login to the HPC and move into Cluster(ICE).

cmd **COMMAND**

```
$ ssh hpc  
$ ice
```

➕ **NOTES**

James Thornton Jr 17 Oct 2016

Option 3 if you have menu enabled.

Step 2.

Move into your class directory.

cmd **COMMAND**

```
$ cd /rsgrps/bh_class/username  
Use YOUR username to be in the right directory
```

Step 3.

Copy the following into a new script called run-interactive.sh :

cmd **COMMAND**

```
#!/bin/bash  
  
#PBS -W group_list=bh_class  
#PBS -q windfall  
#PBS -l jobtype=cluster_only  
#PBS -l select=1:ncpus=2:mem=4gb  
#PBS -l walltime=24:00:00  
#PBS -l cput=24:00:00  
#PBS -l place=pack:shared  
#PBS -M netid@email.arizona.edu
```

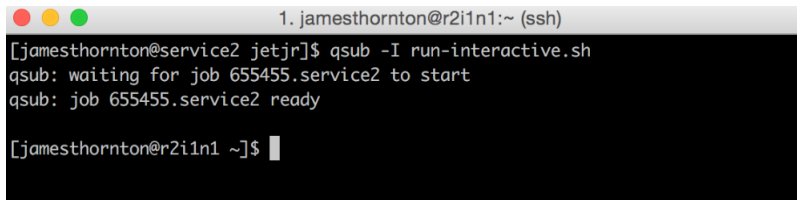
```
#PBS -m bea
Replace netid
```

Step 4.

Submit run-interactive.sh interactively using qsub:

```
cmd COMMAND
qsub -I run-interactive.sh
The capital "I" indicates this will be an interactive job
```

EXPECTED RESULTS

A terminal window titled '1. jameshornton@r2i1n1:~ (ssh)' shows the command '[jameshornton@service2 jetjr]\$ qsub -I run-interactive.sh' being executed. The output is 'qsub: waiting for job 655455.service2 to start' followed by 'qsub: job 655455.service2 ready'. The prompt then changes to '[jameshornton@r2i1n1 ~]\$'.

Step 5.

Once the job is ready move back into your class directory.

```
cmd COMMAND
$ cd /rsgrps/bh_class/username
Use YOUR username
```

Step 6.

Make a directory named prodigal:

```
cmd COMMAND
$ mkdir prodigal
```

Step 7.

Move into your assembly directory which contains your contigs:

```
cmd COMMAND
$ cd /rsgrps/bh_class/username/assembly
```

NOTES

James Thornton Jr 17 Oct 2016

Note: Its possible your contigs are in another directory (megahit-out). Move to the directory where final_contigs.fa are located. Remember final_contigs.fa is the combined assemblies from your partner.

Step 8.

Load prodigal and run it on your fixed_contigs.fa to generate gene calls.

```
cmd COMMAND
$ module load prodigal/2.6.2
$ prodigal -i fixed_contigs.fa -o ../prodigal/gene_calls -a ../prodigal/proteins.faa -
```

```
d ../prodigal/nucleotides.fna
```

IMPORTANT: make sure to run prodigal on the fixed_contigs.fa file which was generated when simplifying the fasta header lines. The output will be placed in the prodigal directory that was created and the file name will be gene_calls in addition to a file containing the protein (proteins.faa) and nucleotide (nucleotide.faa) sequences for the genes.

📌 NOTES

James Thornton Jr 17 Oct 2016

Since final_contigs.fa may be located somewhere other than the assembly directory, you can write out the full paths to make sure the output goes where its suppose to:

```
$ prodigal -i /path/to/final_contigs.fa -o /rsgrps/bh_class/username/prodigal/gene_calls -  
a /rsgrps/bh_class/username/prodigal/proteins.faa -  
d /rsgrps/bh_class/username/prodigal/nucleotides.fna
```

■ ANNOTATIONS

James Thornton Jr 17 Oct 2016

The prodigal command is executed in one line.

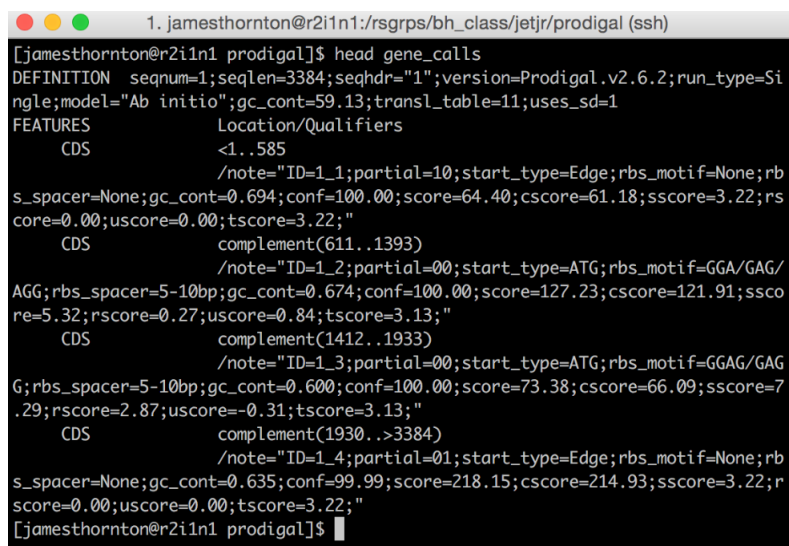
Step 9.

Move into your prodigal directory and make sure the gene calls were generated.

cmd COMMAND

```
$ cd /rsgrps/bh_class/username/prodigal  
$ head gene_calls
```

📄 EXPECTED RESULTS



```
1. james Thornton@r2i1n1:/rsgrps/bh_class/jetjr/prodigal (ssh)  
[jamesthornton@r2i1n1 prodigal]$ head gene_calls  
DEFINITION  seqnum=1;seqlen=3384;seqhdr="1";version=Prodigal.v2.6.2;run_type=Single;model="Ab initio";gc_cont=59.13;transl_table=11;uses_sd=1  
FEATURES             Location/Qualifiers  
     CDS             <1..585  
                     /note="ID=1_1;partial=10;start_type=Edge;rbs_motif=None;rbs_spacer=None;gc_cont=0.694;conf=100.00;score=64.40;cscore=61.18;sscore=3.22;rscore=0.00;uscore=0.00;tscore=3.22;"  
     CDS             complement(611..1393)  
                     /note="ID=1_2;partial=00;start_type=ATG;rbs_motif=GGA/GAG/AGG;rbs_spacer=5-10bp;gc_cont=0.674;conf=100.00;score=127.23;cscore=121.91;sscore=5.32;rscore=0.27;uscore=0.84;tscore=3.13;"  
     CDS             complement(1412..1933)  
                     /note="ID=1_3;partial=00;start_type=ATG;rbs_motif=GGAG/GAGG;rbs_spacer=5-10bp;gc_cont=0.600;conf=100.00;score=73.38;cscore=66.09;sscore=7.29;rscore=2.87;uscore=-0.31;tscore=3.13;"  
     CDS             complement(1930..>3384)  
                     /note="ID=1_4;partial=01;start_type=Edge;rbs_motif=None;rbs_spacer=None;gc_cont=0.635;conf=99.99;score=218.15;cscore=214.93;sscore=3.22;rscore=0.00;uscore=0.00;tscore=3.22;"  
[jamesthornton@r2i1n1 prodigal]$
```

Step 10.

Use your scripting or unix skills to detect how many genes were detected on each of the assemblies. Create a table in google documents that shows the number of "ORFs" or open reading frames

detected on each assembly. We will add additional data to this table later on # of genes with annotation, so the table should have a single column for #ORFs and the rows should be the name of each sample "SRR..."