THE UNIVERSITY OF CHICAGO | Center for Research Informatics

# Introduction to CRI's HPC Cluster for Bioinformatics Computing

Lei Huang
Wenjun Kang
Michael Jarsulic

April 30, 2013

I will be asking for your feedback via an online survey at the end of this training. Your participation would be greatly appreciated.

# Outline

- CRI resource and support
- Access BIOS HPC cluster
- Transfer data files to cluster
- Run jobs on HPC cluster
- Exercises
  - Transfer data files
  - Quality control analysis of NGS data
  - Sequence alignment with BWA and Bowtie2

# Training wiki page

https://wiki.uchicago.edu/display/CRIwksp

# Outline

- CRI resource and support
- Access BIOS HPC cluster
- Transfer data files to cluster
- Run jobs on HPC cluster
- Exercises
  - Transfer data files
  - Quality control analysis of NGS data
  - Sequence alignment with BWA and Bowtie2

# CRI resource and support

- **BIOS HPC cluster** (bios.cri.uchicago.edu)
  - 1024 cores (2.22 GHz AMD Opteron 6274 CPU)
  - 256 GB RAM (4GB per core)
  - 700 TB storage
  - OS: Red Hat Linux Enterprise 6.0
  - Software installed:
    - MPICH2 and Open MPI
    - MOAB scheduler / Torque Resource Manager
    - Java
    - Python
    - Perl
    - PGI C/C++/FORTRAN compiler; Intel C/C++/FORTRAN compiler
    - R/Bioconductor
    - Other Bioinformatics software

# CRI resource and support

- **Large memory server** (lmem-cri.uchicago.edu)
  - 160 cores (2.40GHz Intel® Xeon® E7-8870 CPU)
  - 1 TB RAM
  - 40 TB storage
- **Backup and Restore System**
  - Tivoli Storage Manager (TSM)
- **Self Service Data Analysis Environment**
  - CRI Galaxy (https://crigalaxy.uchicago.edu )

# CRI Support

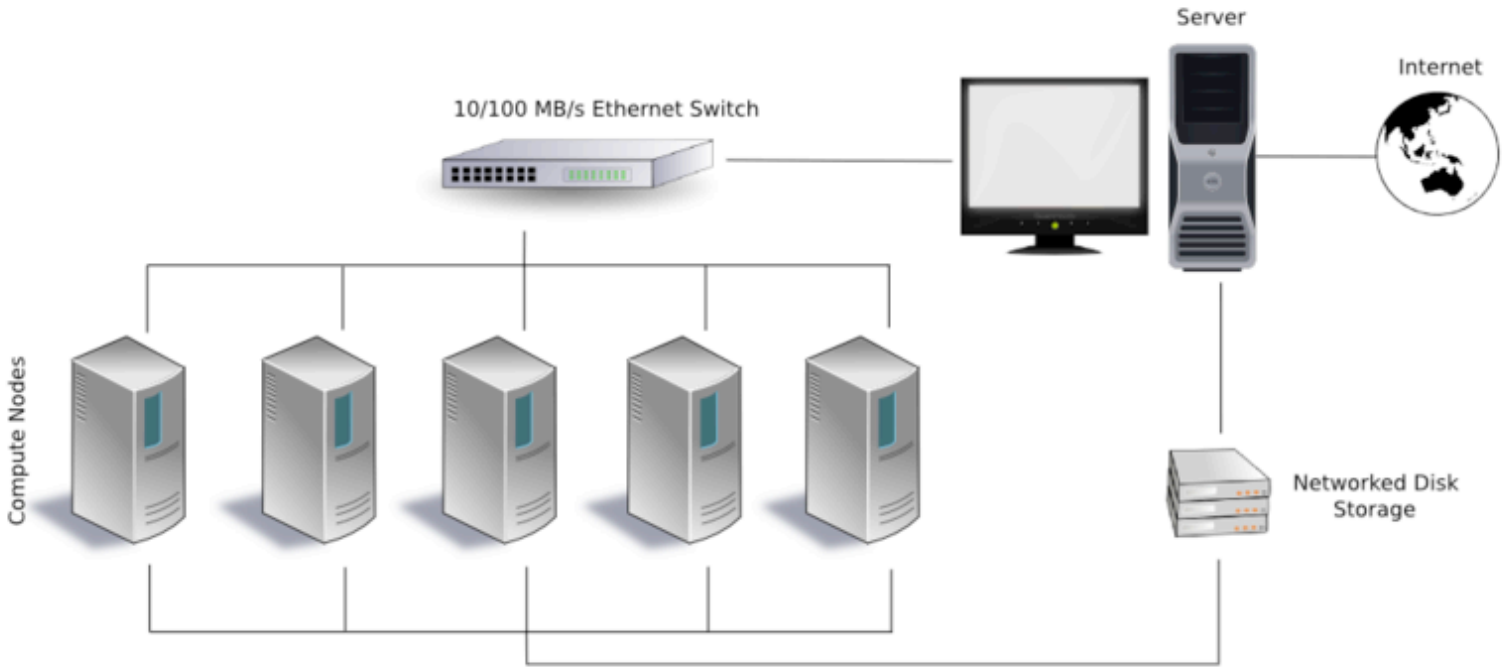
- CRI website:
  - http://cri.uchicago.edu
  - Email: support@rt.cri.uchicago.edu

- Bioinformatics core
  - https://biocore.cri.uchicago.edu
  - Email: bioinformatics@bsd.uchicago.edu

# Outline

- CRI resource and support
- Access BIOS HPC cluster
- Transfer data files to cluster
- Run jobs on HPC cluster
- Exercises
  - Transfer data files
  - Quality control analysis of NGS data
  - Sequence alignment with BWA and Bowtie2

# What is a Computer Cluster




Source: http://en.wikipedia.org/wiki/Computer_cluster

# Do you really need a Cluster

- If your problem
  - Is not tractable on the desktop computer
  - Is memory-hogging
  - Requires rapid turnaround of results
  - Would benefit from having jobs scheduled
  - Is computationally intensive that might eat up desktop resources
- Does your job run on Linux
- Can your job be executed without manual intervention, i.e. batch mode

**Consider running it on the Cluster**

# Procedures of running jobs on cluster

- Develop strategy for running jobs
- Have required software installed
- Develop job submission scripts
- Run your job

# Use BIOS HPC Cluster

**bios.cri.uchicago.edu**

- **Access to BIOS**
  - A BSDAD accounts is required to access bios.cri.uchicago.edu
  - Temporary accounts used during the training session will be deactivated.
- **Requesting BSDAD account**
  - Contact CBIS at **help@bsd.uchicago.edu** or call 773-702-3456
  - Please note: For new accounts CBIS requires that the request comes from a section admin
- **Account Eligibility**
  - All BSD members
  - Non-BSD members or outside collaborators must have a BSD faculty sponsor
- **Technical Support**
  - Issues accessing BIOS or need technical assistance, contact the CRI Help Desk at **support@rt.cri.uchicago.edu** or call (773) 834-8475

# Install SSH Client

- **For Windows user**
  - Download and install Putty from http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
  - Download and install Xming and Xming fonts
    - http://sourceforge.net/projects/xming
    - http://sourceforge.net/projects/xming/files/Xming-fonts/7.5.0.47/
- **For Mac/Unix/Linux user**
  - Already installed

# Log into BIOS Cluster

- **For Mac/Unix/Linux user**
  - Type `ssh –X username@bios.cri.uchicago.edu`
  - Type in your password when prompted
  - Type yes if you are prompted to accept a key
- **For Windows user**
  - Open PuTTY
  - In Host Name box, enter `bios.cri.uchicago.edu`
  - Select SSH as the Connection Type
  - Verify the port number set in Port Box is 22
  - Press the Open button at the bottom
  - Type in your username and password when prompted

# Log into BIOS Cluster

- **For Mac/Unix/Linux user**
  - Type `ssh –X username@bios.cri.uchicago.edu`
  - Type in your password when prompted
  - Type yes if you are prompted to accept a key
- **For Windows user**
  - Open PuTTY
  - In Host Name box, enter `bios.cri.uchicago.edu`
  - Select SSH as the Connection Type
  - Verify the port number set in Port Box is 22
  - Press the Open button at the bottom
  - Type in your username and password when prompted

# Prepare for exercises

```
> mkdir ~/CRI_training
> cd CRI_training
> mkdir Ex1 Ex2 Ex3 Ex4
```

**Note**: If you are doing “real-world” data analysis on BIOS cluster, consider running all your analyses under **/scratch** directory, it has more space and faster access compared to your **home** directory. Please contact CRI tech support for details about obtaining access, etc.

# Outline

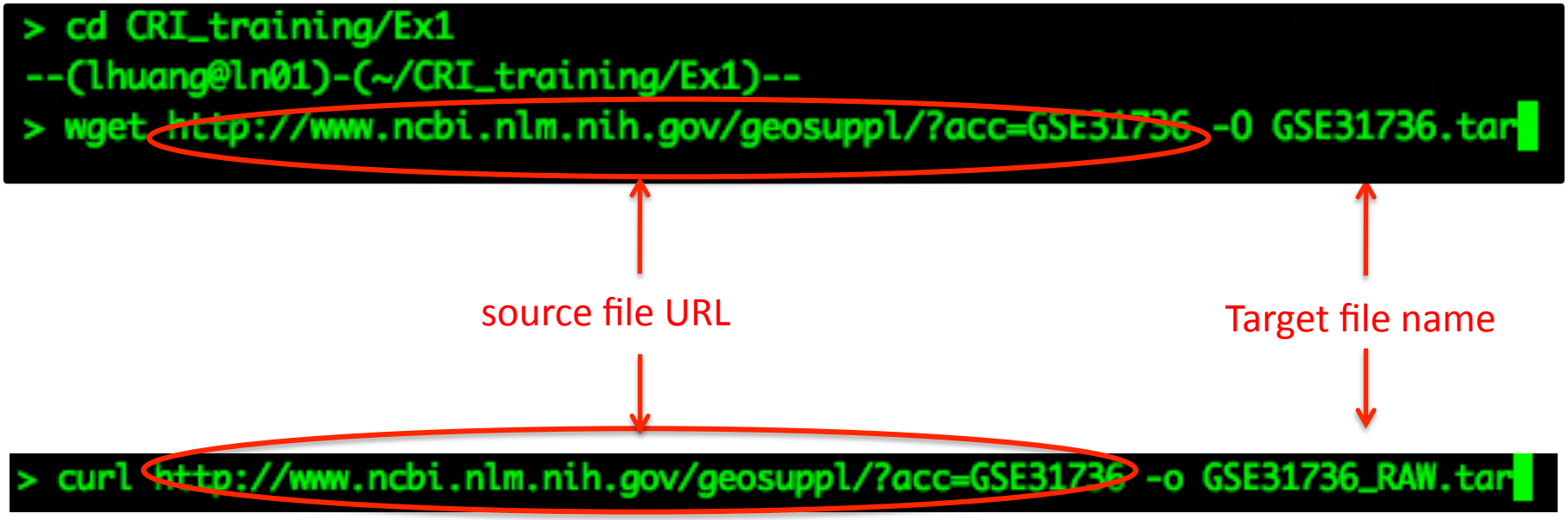
- CRI resource and support
- Access BIOS HPC cluster
- Transfer data files to cluster
- Run jobs on HPC cluster
- Exercises
  - Transfer data files
  - Quality control analysis of NGS data
  - Sequence alignment with BWA and Bowtie2

# Transfer Data Files to Cluster

- **From remote site**
  - *wget* and *curl* command
- **From local computer**
  - *scp* and *rsync* command (Mac/Unix/Linux)
  - WinSCP software (Windows)

# Exercise 1.1 Download file from remote to BIOS

- **From remote site**
  - *wget* and *curl* command



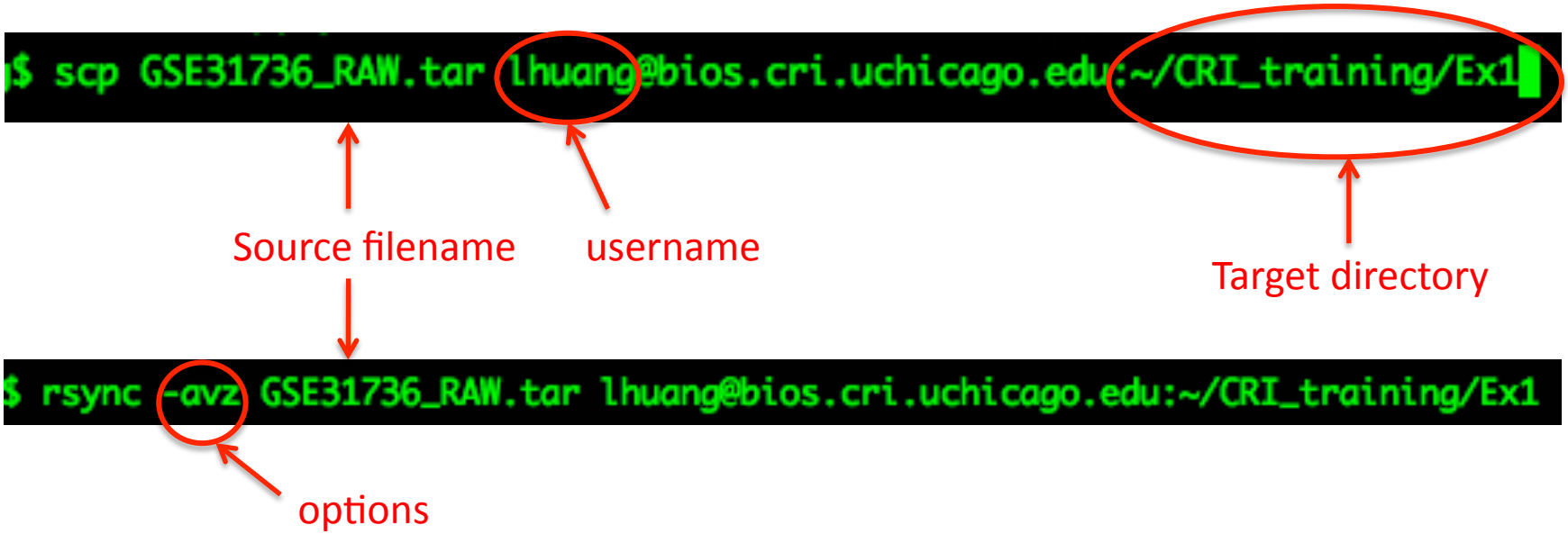
# Exercise 1.2 Upload file to BIOS

- **From local computer**

  Download the following file to your local computer
  ftp://logia.cri.uchicago.edu/tutorials/Apr2013/Ex1/GSE31736_RAW.tar

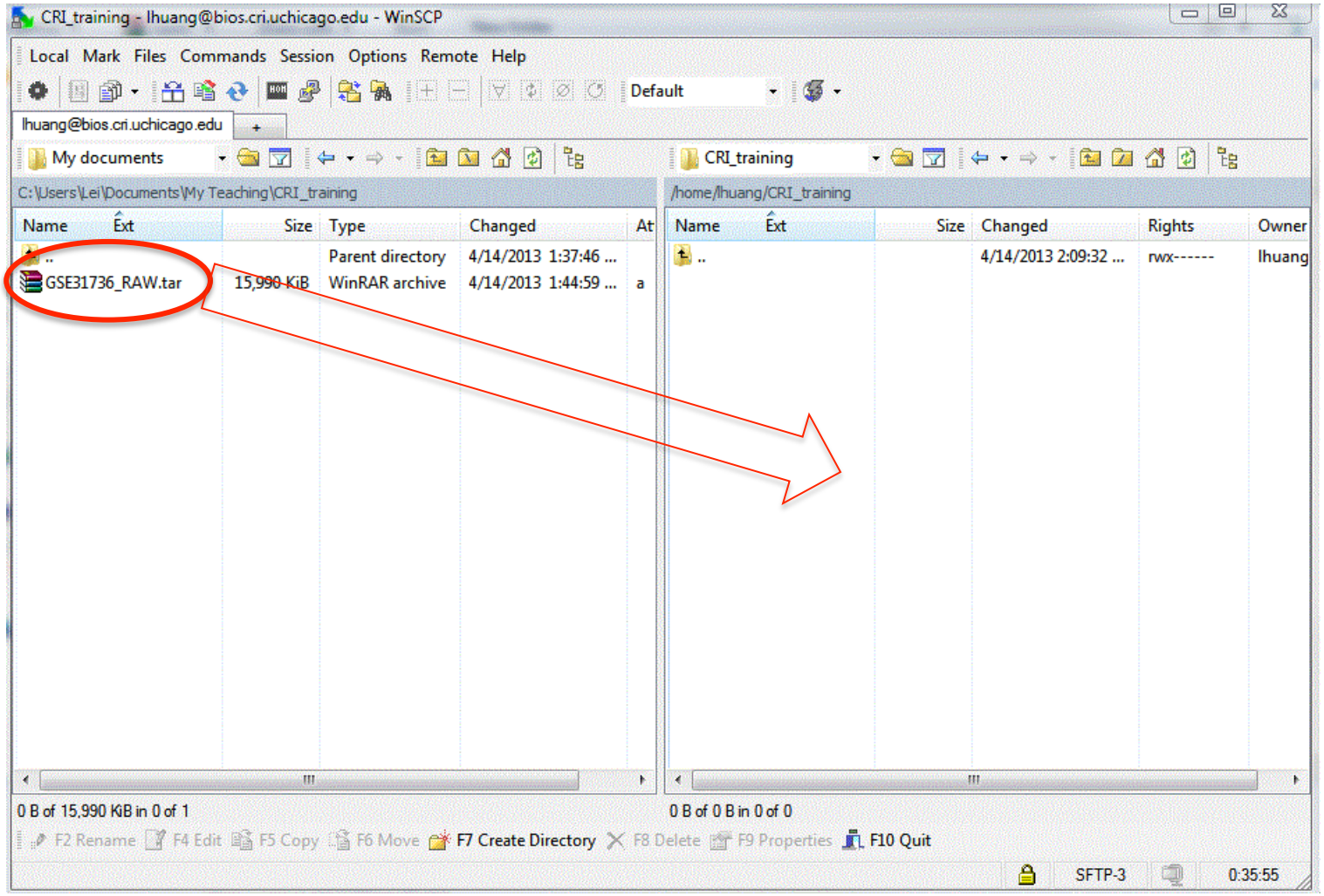
- **For Mac/Unix/Linux users**

  *scp* and *rsync*



# Exercise 1.2 Transfer Data Files

- For Windows user: WinSCP



# Outline

- CRI resource and support
- Access BIOS HPC cluster
- Transfer data files to cluster
- Run jobs on HPC cluster
- Exercises
  - Transfer data files
  - Quality control analysis of NGS data
  - Sequence alignment with BWA and Bowtie2

# Running an interactive job

- When
  - Debugging
  - Need to see the intermediate results from a program, e.g. R
- How

```
> qsub -I
qsub: waiting for job 120521.sc01 to start
qsub: job 120521.sc01 ready
```

# Quality check on NGS data

- FastQC – Java program
- Import data from FASTQ, SAM or BAM files
- Tell you in which areas there may be problems
- Summary graphs and tables to quickly assess your data
- Export results to an HTML-based report

# Exercise 2: running an interactive job on BIOS

- Quality control analysis of two NGS sequence read files
- Download two files in FASTQ format

```
> cd ~/CRI_training/Ex2
--(lhuang@ln01)-(~/CRI_training/Ex2)--
> wget ftp://logia.cri.uchicago.edu/tutorials/Apr2013/Ex2/seqGood.fastq
--(lhuang@ln01)-(~/CRI_training/Ex2)--
> wget ftp://logia.cri.uchicago.edu/tutorials/Apr2013/Ex2/seqBad.fastq
```

# FASTQ file format

Line 1: sequence id
Line2: sequence

Line 3:
Line 4: quality score

```
> head -8 seqGood.fastq
@ERR030881.46427389/1
CAAGACCTGGGCGTTCAGGCTGCCACTGCTGTCCATATCCCCTACCACAG
+
HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHEHHHHHHHHHHHHHHHHH
@ERR030881.71523653/1
CAAGACCTGGGCGTTCAGGCTGCCACTGCTGTCCATATCCCCTACCACAG
+
HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHIHHHHHHHFH
--(lhuang@ln01)-(~/CRI_training/Ex2)--
> tail -8 seqGood.fastq
@ERR030881.31409188/1
GCTGAAGCCATCCAACACCTATTTTTGTACAGCCTGTGAACTAAGAAAAT
+
55555@@>>AHHIHHHFHGHHHGFG?CCCHFFHHHHHHHHHHGHHHHHEE
@ERR030881.34210561/1
CTTCTATTGGTTACAGAAGTCATATTTTTAAAACCTATATAAATAACATG
+
5555444445DD>D=4444555514AA<8:5444515444444444445
```

# Read count

Method 1

```
> grep --count ^@ERR030881 seqGood.fastq
28477
```

Method 2

```
> wc -l seqGood.fastq
113908 seqGood.fastq
```

113908/4=28477

# Run fastQC in interactive mode

```
> qsub -I
qsub: waiting for job 110149.sc01 to start
qsub: job 110149.sc01 ready
> cd ~/CRI_training/Ex2
> fastqc seqGood.fastq
> fastqc seqBad.fastq
> exit
```

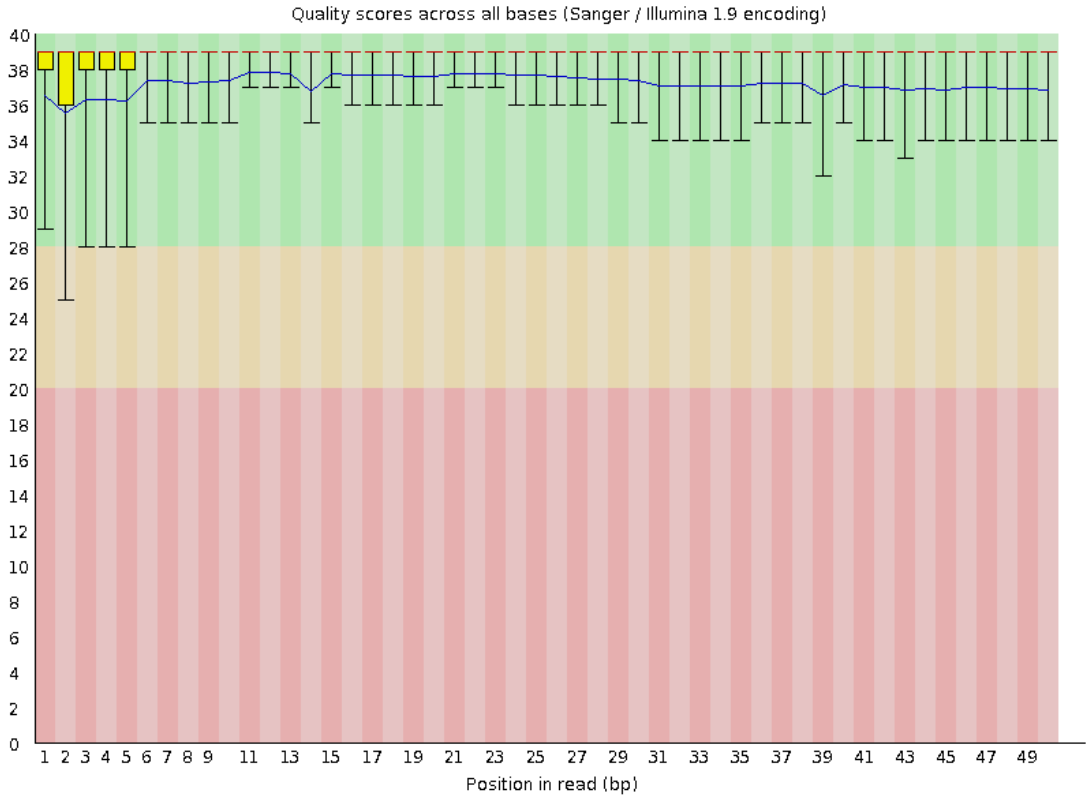
# FastQC summary

```
> cd ~/CRI_training/Ex2/seqGood_fastqc
--(lhuang@ln01)-(~/CRI_training/Ex2/seqGood_fastqc)--
> ls
fastqc_data.txt  fastqc_report.html  Icons  Images  summary.txt
--(lhuang@ln01)-(~/CRI_training/Ex2/seqGood_fastqc)--
> cat summary.txt
PASS    Basic Statistics        seqGood.fastq
PASS    Per base sequence quality       seqGood.fastq
PASS    Per sequence quality scores     seqGood.fastq
FAIL    Per base sequence content       seqGood.fastq
FAIL    Per base GC content     seqGood.fastq
FAIL    Per sequence GC content seqGood.fastq
PASS    Per base N content      seqGood.fastq
PASS    Sequence Length Distribution    seqGood.fastq
FAIL    Sequence Duplication Levels     seqGood.fastq
FAIL    Overrepresented sequences       seqGood.fastq
FAIL    Kmer Content    seqGood.fastq
```

Is sequence trimming needed before alignment?

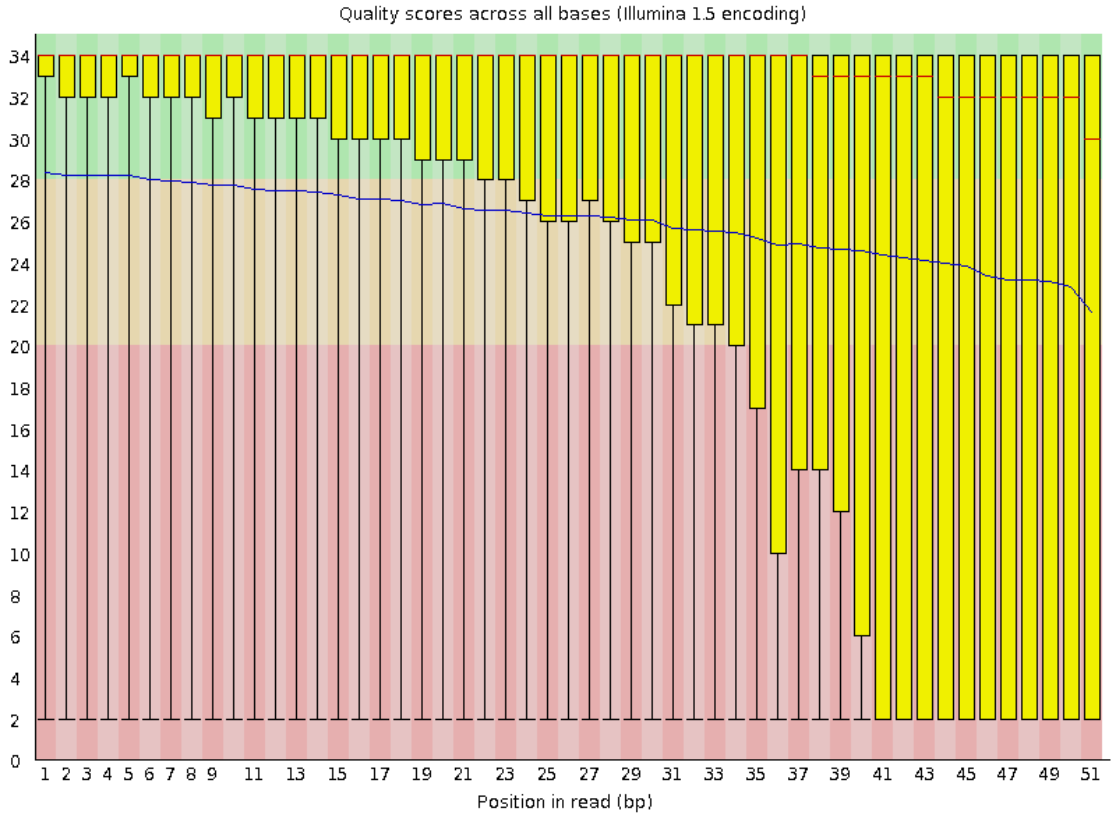
# Per base quality for “good” FASTQ file

```
> cd seqGood_fastqc
--(lhuang@ln01)-(~/CRI_training/Ex2/seqGood_fastqc)--
> ls
fastqc_data.txt  fastqc_report.html  Icons  Images  summary.txt
--(lhuang@ln01)-(~/CRI_training/Ex2/seqGood_fastqc)--
> firefox fastqc_report.html
```



# Per base quality for “bad” FASTQ file

```
> cd ../
--(lhuang@ln01)-(~/CRI_training/Ex2)--
> cd seqBad_fastqc
--(lhuang@ln01)-(~/CRI_training/Ex2/seqBad_fastqc)--
> ls
fastqc_data.txt  fastqc_report.html  Icons  Images  summary.txt
--(lhuang@ln01)-(~/CRI_training/Ex2/seqBad_fastqc)--
> firefox fastqc_report.html
```



# Prepare batch job submission

- BIOS is a batch, queued system
  - Jobs must be submitted to a queue, wait and then be processed
  - No interaction with your job once submitted
- User need to provide a job submission script containing
  - Directives to request the proper resources for the execution of your job
  - Commands that will be executed on the cluster nodes

# Common job operation commands

- **Submit jobs**

  > qsub <*job_script*>

- **Monitor job status**

  > qstat

  > showq

- **Alter jobs after submission**

  > qdel <*job_id*>

  > qdel *all*

# Job submission script template

```
#!/bin/bash
################################
# Resource Manager Directives #
################################
### Set the name of the job
#PBS -N jobname
### Select the shell you would like to script to execute within.
#PBS -S /bin/bash
### Inform the scheduler of the expected runtime, where
### walltime=HH:MM:SS.
#PBS -l walltime=0:59:00
### Inform the scheduler of the number of CPU cores for your job. This
### example will allocate two cores on a single node
#PBS -l nodes=1:ppn=2
### Inform the scheduler of the amount of memory you expect
### to use. Use units of 'b', 'kb', 'mb', or 'gb'.
#PBS -l mem=512mb
### Set the destination for your program's output: stdout and stderr.
#PBS -o $HOME/${PBS_JOBNAME}.e${PBS_JOBID}
#PBS -e $HOME/${PBS_JOBNAME}.o${PBS_JOBID}
##################
# Job Execution #
##################
# the program to be executed
./command &> output
```

# Exercise 3: Running FastQC in batch mode

- Paired-end RNASeq data from Illumina's Human BodyMap 2.0 project
- Generated on HiSeq 2000 instruments in 2010
- Consist of 16 human tissue types, including adrenal, adipose, brain, breast, colon, **heart**, **kidney**, liver, lung, lymph, ovary, prostate, skeletal muscle, testes, thyroid, and white blood cells.

# Download and view compressed FASTQ files

```
> cd ~/CRI_training/Ex3
> wget ftp://logia.cri.uchicago.edu/tutorials/Apr2013/Ex3/*.gz
> ls
```

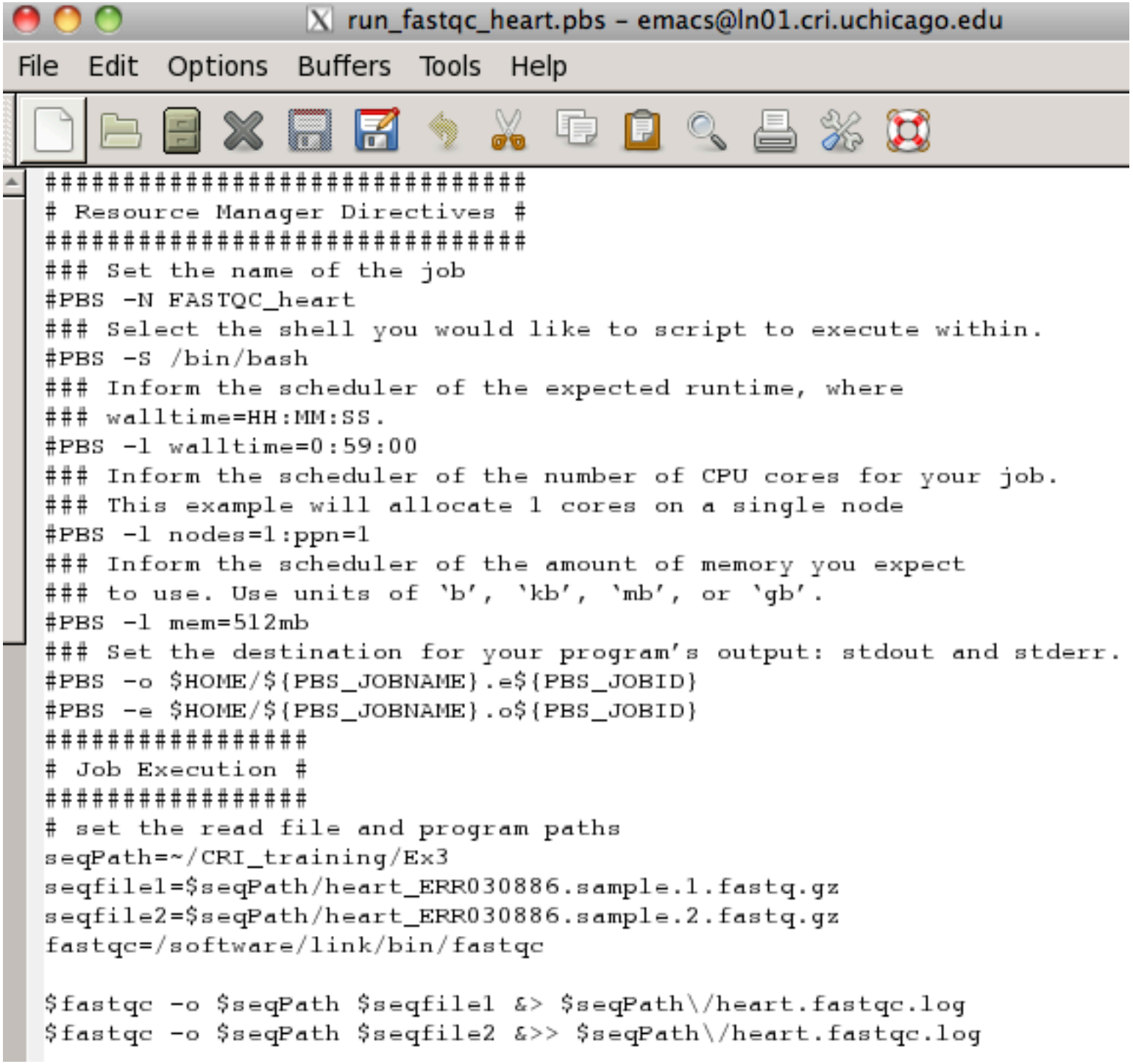
```
> zless heart_ERR030886.sample.1.fastq.gz
```

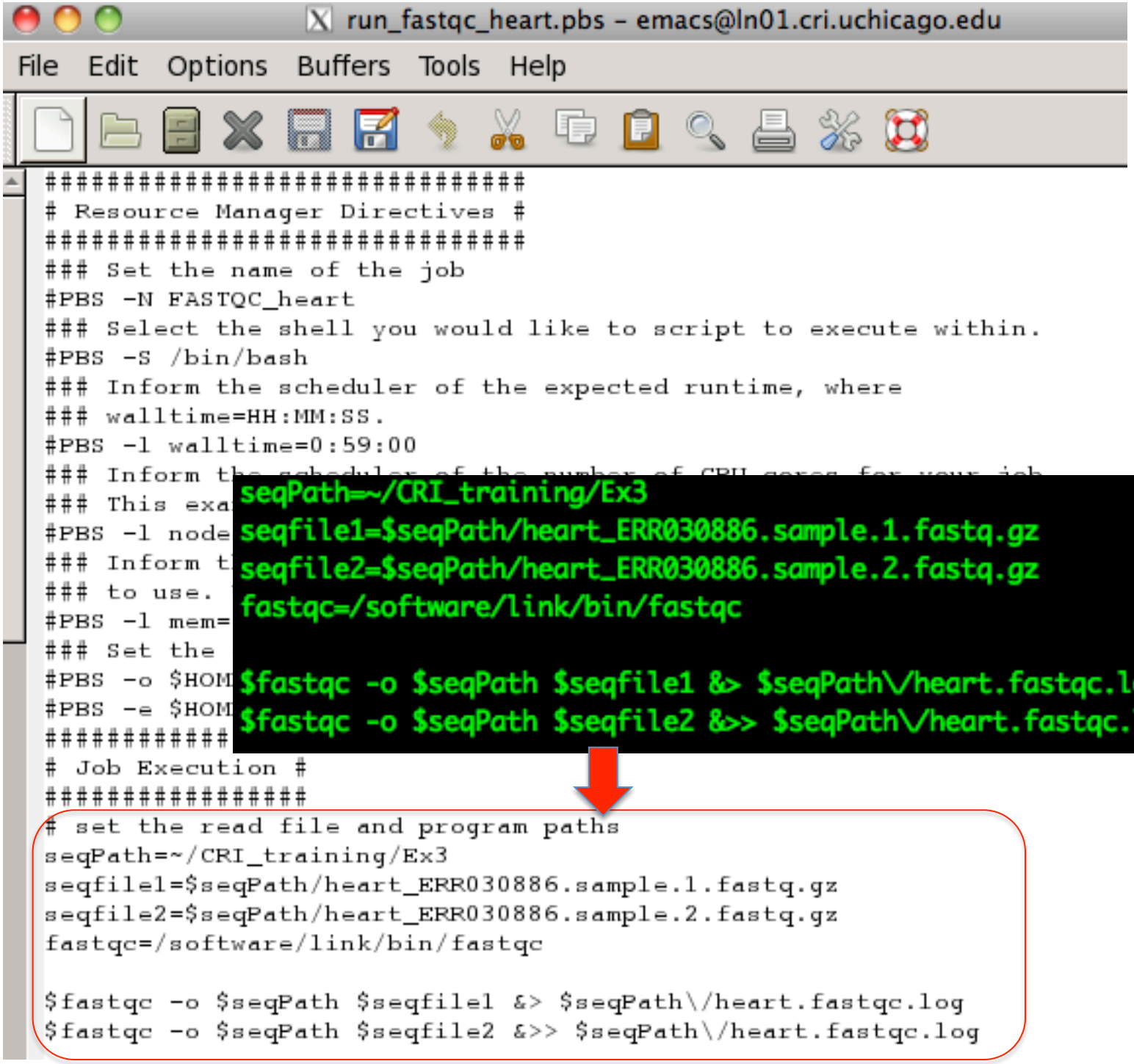
```
@ERR030886.19085532/1
CTGATGTGGACAATGGGCTTGACCATTTCCTCAAGTGCTTTCTGTCTCTC
+
HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
@ERR030886.21011256/1
ATGAGATATTTTTATTGAAAAGAAAAGAGAACCTGAGAACTCTACAGATT
+
HHHHHHHHBHHHGGGGGHHHHHHGGGGHHHHHHHHHHHHHHHHHHEHH
@ERR030886.36367438/1
GGGACATCAGAAGCCACAAATGAGCCTCCCGAGGATGTTCTGCTGAAACA
+
HHHHHHHHGHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
```

zcat also works

# Create a job submission script for fastQC on heart tissue sequence read files

> emacs run_fastqc_heart.pbs &


run_fastqc_heart.pbs - emacs@ln01.cri.uchicago.edu
File Edit Options Buffers Tools Help
##################################
# Resource Manager Directives #
##################################
### Set the name of the job
#PBS -N FASTQC_heart
### Select the shell you would like to script to execute within.
#PBS -S /bin/bash
### Inform the scheduler of the expected runtime, where
### walltime=HH:MM:SS.
#PBS -l walltime=0:59:00
### Inform the scheduler of the number of CPU cores for your job.
### This example will allocate 1 cores on a single node
#PBS -l nodes=1:ppn=1
### Inform the scheduler of the amount of memory you expect
### to use. Use units of 'b', 'kb', 'mb', or 'gb'.
#PBS -l mem=512mb
### Set the destination for your program's output: stdout and stderr.
#PBS -o $HOME/${PBS_JOBNAME}.e${PBS_JOBID}
#PBS -e $HOME/${PBS_JOBNAME}.o${PBS_JOBID}
##################
# Job Execution #
##################
# set the read file and program paths
seqPath=~/CRI_training/Ex3
seqfile1=$seqPath/heart_ERR030886.sample.1.fastq.gz
seqfile2=$seqPath/heart_ERR030886.sample.2.fastq.gz
fastqc=/software/link/bin/fastqc
$fastqc -o $seqPath $seqfile1 &> $seqPath\/heart.fastqc.log
$fastqc -o $seqPath $seqfile2 &>> $seqPath\/heart.fastqc.log


run_fastqc_heart.pbs - emacs@ln01.cri.uchicago.edu
File Edit Options Buffers Tools Help
#################################
# Resource Manager Directives #
#################################
### Set the name of the job
#PBS -N FASTQC_heart
### Select the shell you would like to script to execute within.
#PBS -S /bin/bash
### Inform the scheduler of the expected runtime, where
### walltime=HH:MM:SS.
#PBS -l walltime=0:59:00
seqPath=~/CRI_training/Ex3
seqfile1=$seqPath/heart_ERR030886.sample.1.fastq.gz
seqfile2=$seqPath/heart_ERR030886.sample.2.fastq.gz
fastqc=/software/link/bin/fastqc
$fastqc -o $seqPath $seqfile1 &> $seqPath\/heart.fastqc.log
$fastqc -o $seqPath $seqfile2 &>> $seqPath\/heart.fastqc.log
# Job Execution #
# set the read file and program paths
seqPath=~/CRI_training/Ex3
seqfile1=$seqPath/heart_ERR030886.sample.1.fastq.gz
seqfile2=$seqPath/heart_ERR030886.sample.2.fastq.gz
fastqc=/software/link/bin/fastqc
$fastqc -o $seqPath $seqfile1 &> $seqPath\/heart.fastqc.log
$fastqc -o $seqPath $seqfile2 &>> $seqPath\/heart.fastqc.log

# Submit a batch job

```
> qsub run_fastqc_heart.pbs
110156.sc01
--(username@ln01)-(~/CRI_training/Ex3)--
```

Check job status

```
> qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
110156.sc01               ...stqc_heart.pbs username               0 R batch
```

Another way

```
> showq
```

Or for specified user

```
> showq | grep username
```

# Do by yourself

- Create a job submission script (run_fastqc_kidney.pbs) for fastQC on kidney tissue sequence read files

- Submit run_fastqc_kidney.pbs

# Create a shell script for job submission scripts

If you have many job submission scripts

```
#!/bin/bash

qsub run_fastqc_heart.pbs
sleep 2
qsub run_fastqc_kidney.pbs
...
```

Then from login node

```
> sh submit_jobs.sh
```

# Exercise 4: Sequence alignment on HPC cluster

Copy sequence read files from Exercise 3

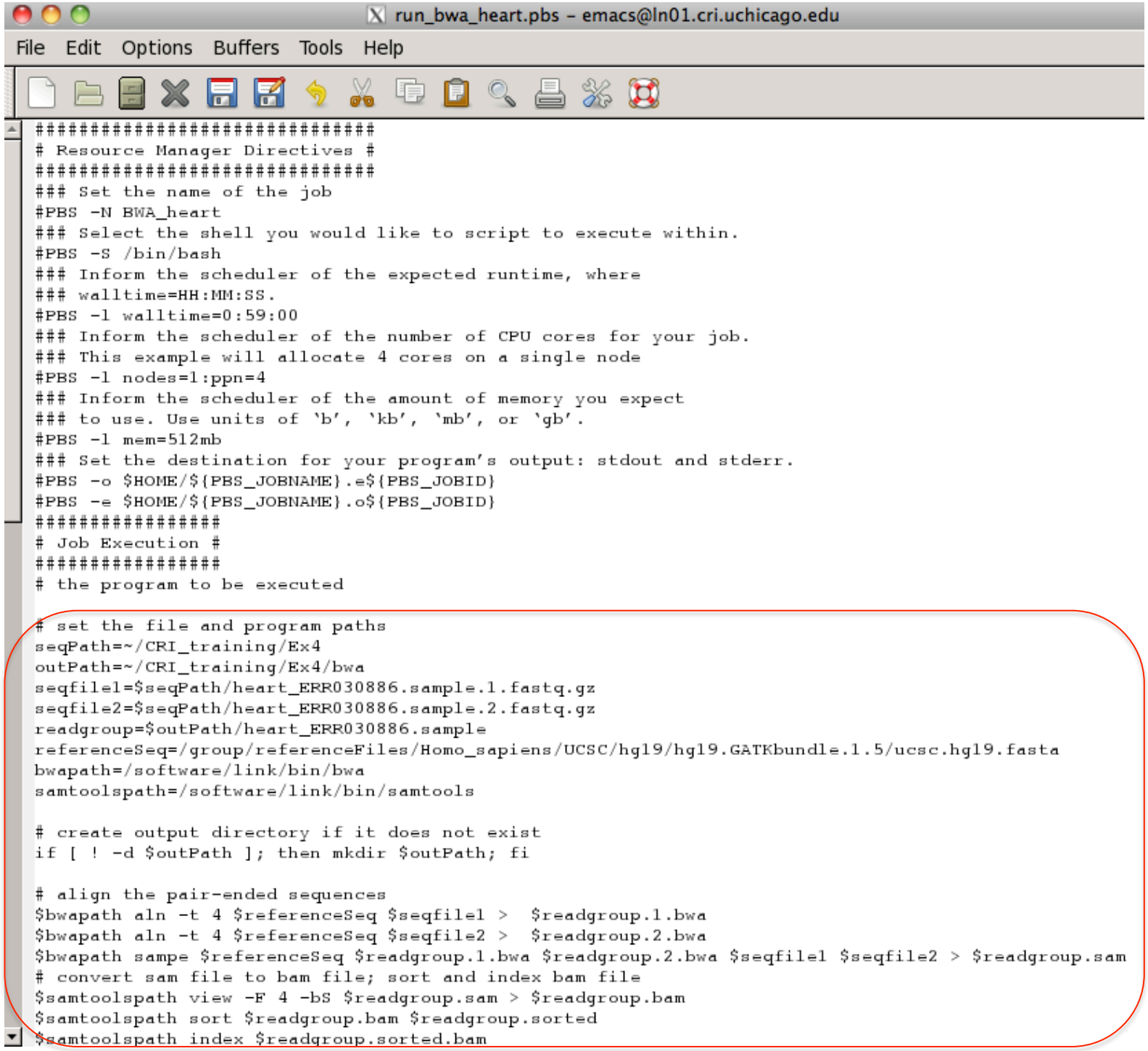
```
> cp ~/CRI_training/Ex3/*.fastq.gz ~/CRI_training/Ex4
```

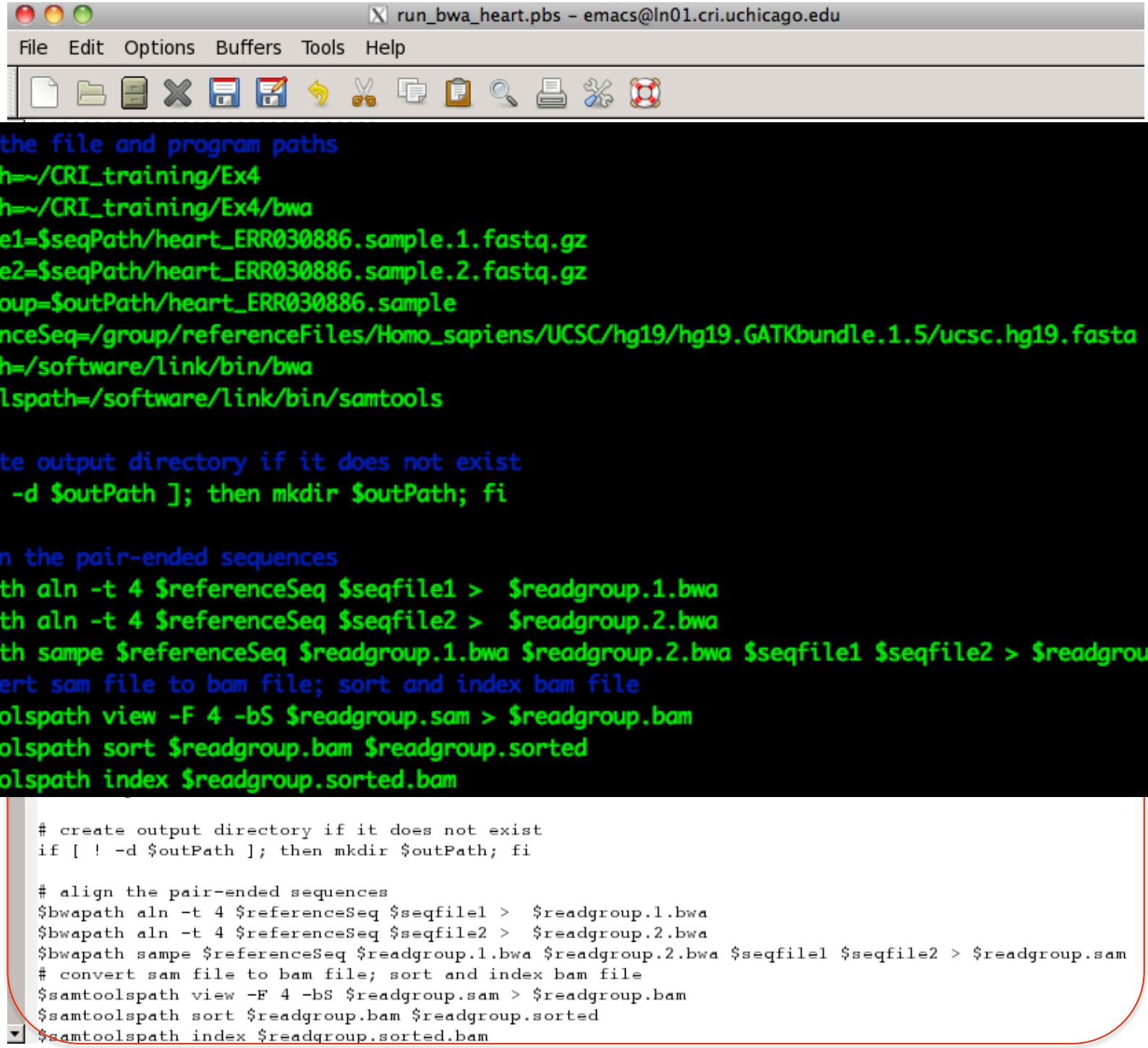
```
> ls
heart_ERR030886.sample.1.fastq.gz
heart_ERR030886.sample.2.fastq.gz
kidney_ERR030885.sample.1.fastq.gz
kidney_ERR030885.sample.2.fastq.gz
```

# Create a job submission script for alignment of heart tissue sequence read files

BWA aligner (http://bio-**bwa**.sourceforge.net/)

```
> emacs run_bwa_heart.pbs &
```


run_bwa_heart.pbs – emacs@ln01.cri.uchicago.edu
File Edit Options Buffers Tools Help
##################################
# Resource Manager Directives #
##################################
### Set the name of the job
#PBS -N BWA_heart
### Select the shell you would like to script to execute within.
#PBS -S /bin/bash
### Inform the scheduler of the expected runtime, where
### walltime=HH:MM:SS.
#PBS -l walltime=0:59:00
### Inform the scheduler of the number of CPU cores for your job.
### This example will allocate 4 cores on a single node
#PBS -l nodes=1:ppn=4
### Inform the scheduler of the amount of memory you expect
### to use. Use units of 'b', 'kb', 'mb', or 'gb'.
#PBS -l mem=512mb
### Set the destination for your program's output: stdout and stderr.
#PBS -o $HOME/${PBS_JOBNAME}.e${PBS_JOBID}
#PBS -e $HOME/${PBS_JOBNAME}.o${PBS_JOBID}
##################
# Job Execution #
##################
# the program to be executed
# set the file and program paths
seqPath=~/CRI_training/Ex4
outPath=~/CRI_training/Ex4/bwa
seqfile1=$seqPath/heart_ERR030886.sample.1.fastq.gz
seqfile2=$seqPath/heart_ERR030886.sample.2.fastq.gz
readgroup=$outPath/heart_ERR030886.sample
referenceSeq=/group/referenceFiles/Homo_sapiens/UCSC/hg19/hg19.GATKbundle.1.5/ucsc.hg19.fasta
bwapath=/software/link/bin/bwa
samtoolspath=/software/link/bin/samtools
# create output directory if it does not exist
if [ ! -d $outPath ]; then mkdir $outPath; fi
# align the pair-ended sequences
$bwapath aln -t 4 $referenceSeq $seqfile1 > $readgroup.1.bwa
$bwapath aln -t 4 $referenceSeq $seqfile2 > $readgroup.2.bwa
$bwapath sampe $referenceSeq $readgroup.1.bwa $readgroup.2.bwa $seqfile1 $seqfile2 > $readgroup.sam
# convert sam file to bam file; sort and index bam file
$samtoolspath view -F 4 -bS $readgroup.sam > $readgroup.bam
$samtoolspath sort $readgroup.bam $readgroup.sorted
$samtoolspath index $readgroup.sorted.bam


run_bwa_heart.pbs – emacs@ln01.cri.uchicago.edu
File Edit Options Buffers Tools Help
# set the file and program paths
seqPath=~/CRI_training/Ex4
outPath=~/CRI_training/Ex4/bwa
seqfile1=$seqPath/heart_ERR030886.sample.1.fastq.gz
seqfile2=$seqPath/heart_ERR030886.sample.2.fastq.gz
readgroup=$outPath/heart_ERR030886.sample
referenceSeq=/group/referenceFiles/Homo_sapiens/UCSC/hg19/hg19.GATKbundle.1.5/ucsc.hg19.fasta
bwapath=/software/link/bin/bwa
samtoolspath=/software/link/bin/samtools
# create output directory if it does not exist
if [ ! -d $outPath ]; then mkdir $outPath; fi
# align the pair-ended sequences
$bwapath aln -t 4 $referenceSeq $seqfile1 > $readgroup.1.bwa
$bwapath aln -t 4 $referenceSeq $seqfile2 > $readgroup.2.bwa
$bwapath sampe $referenceSeq $readgroup.1.bwa $readgroup.2.bwa $seqfile1 $seqfile2 > $readgroup.sam
# convert sam file to bam file; sort and index bam file
$samtoolspath view -F 4 -bS $readgroup.sam > $readgroup.bam
$samtoolspath sort $readgroup.bam $readgroup.sorted
$samtoolspath index $readgroup.sorted.bam

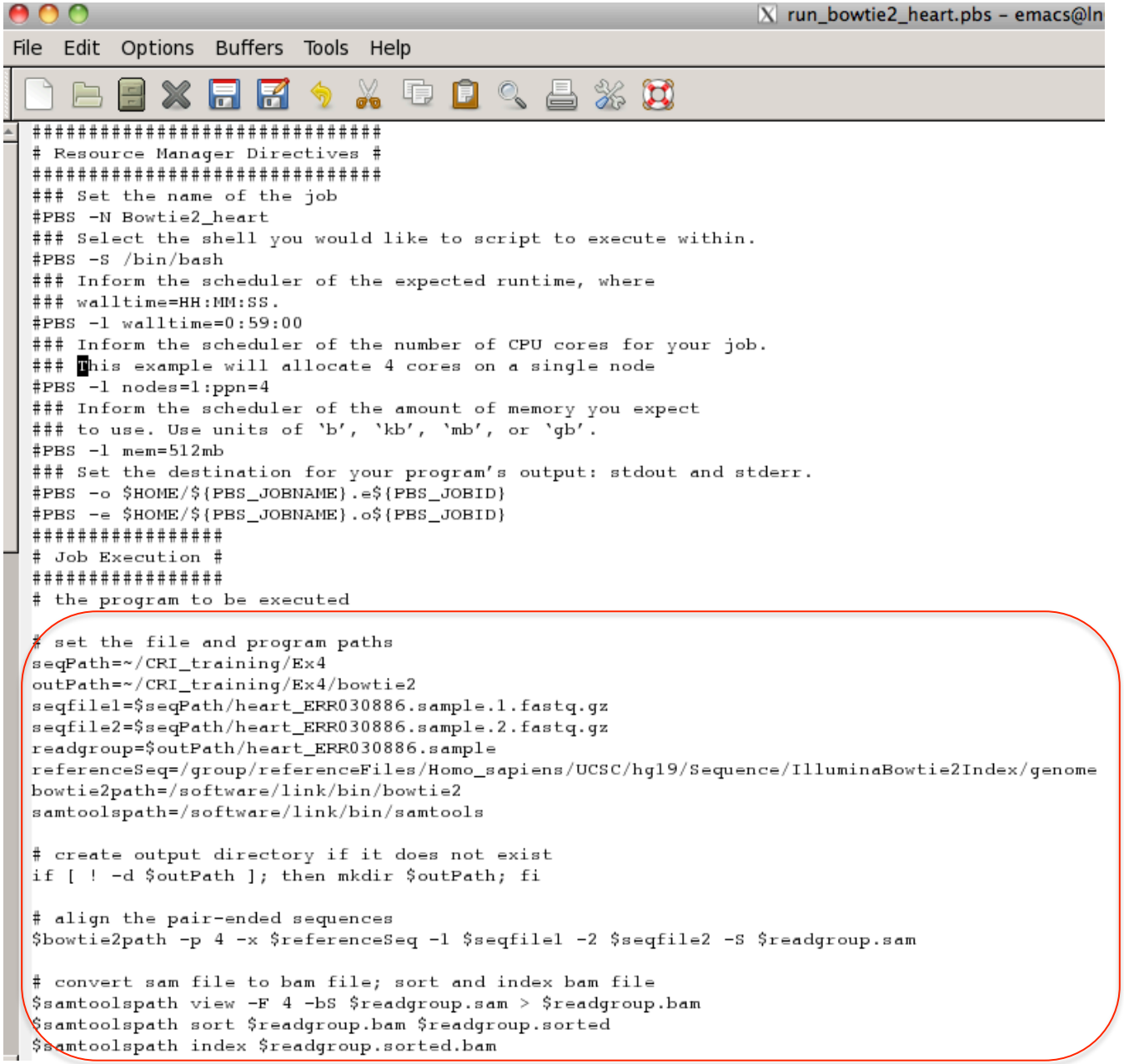
# Submit a batch job

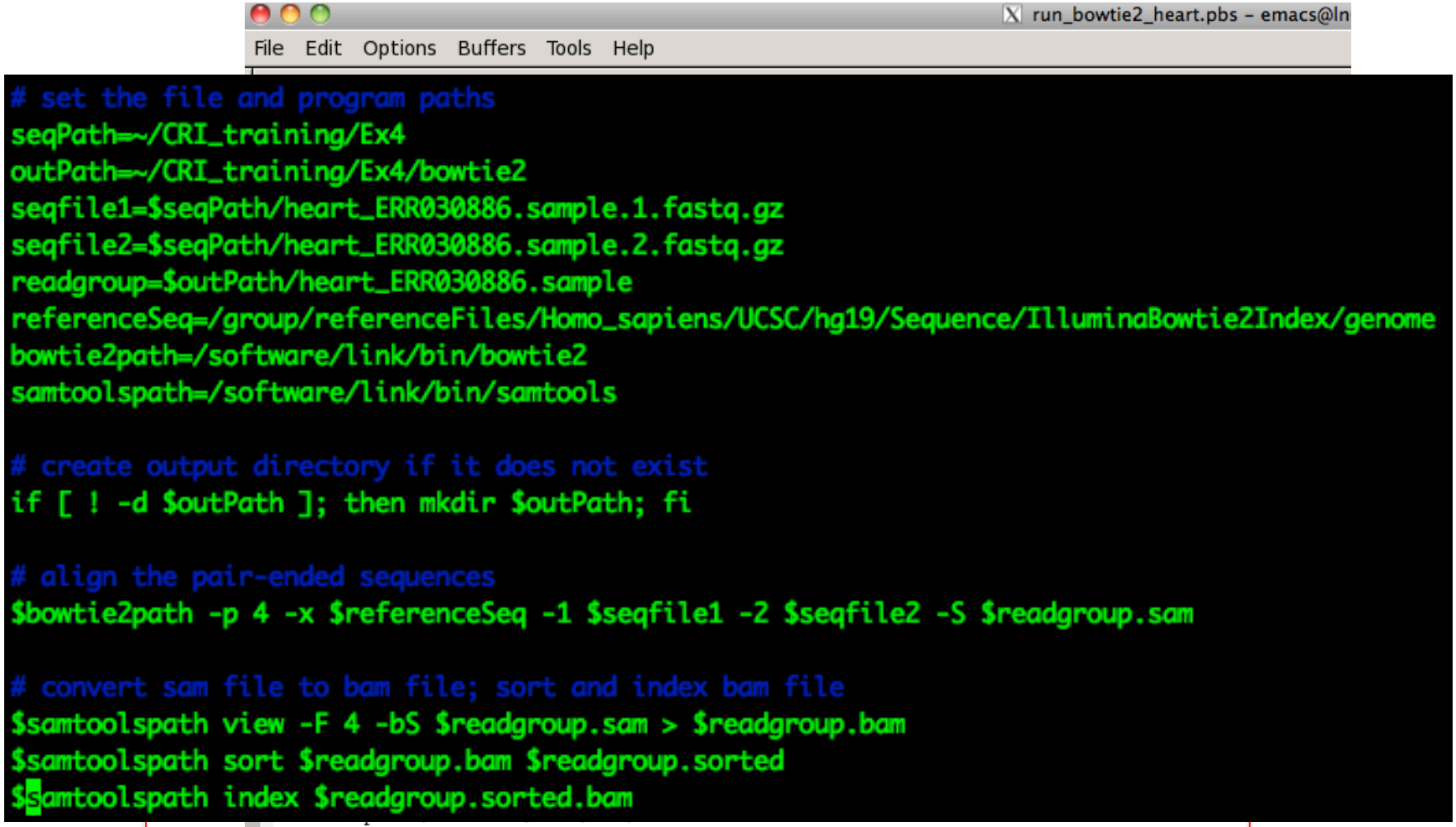
```
> qsub run_bwa_heart.pbs

> qstat
```

# Create a job submission script for alignment of heart tissue sequence read files

Bowtie2 (http://bowtie-bio.sourceforge.net/**bowtie2**/)

```
> emacs run_bowtie2_heart.pbs &
```


run_bowtie2_heart.pbs - emacs@ln
File Edit Options Buffers Tools Help
##################################
# Resource Manager Directives #
##################################
### Set the name of the job
#PBS -N Bowtie2_heart
### Select the shell you would like to script to execute within.
#PBS -S /bin/bash
### Inform the scheduler of the expected runtime, where
### walltime=HH:MM:SS.
#PBS -l walltime=0:59:00
### Inform the scheduler of the number of CPU cores for your job.
### This example will allocate 4 cores on a single node
#PBS -l nodes=1:ppn=4
### Inform the scheduler of the amount of memory you expect
### to use. Use units of 'b', 'kb', 'mb', or 'gb'.
#PBS -l mem=512mb
### Set the destination for your program's output: stdout and stderr.
#PBS -o $HOME/${PBS_JOBNAME}.e${PBS_JOBID}
#PBS -e $HOME/${PBS_JOBNAME}.o${PBS_JOBID}
##################
# Job Execution #
##################
# the program to be executed
# set the file and program paths
seqPath=~/CRI_training/Ex4
outPath=~/CRI_training/Ex4/bowtie2
seqfile1=$seqPath/heart_ERR030886.sample.1.fastq.gz
seqfile2=$seqPath/heart_ERR030886.sample.2.fastq.gz
readgroup=$outPath/heart_ERR030886.sample
referenceSeq=/group/referenceFiles/Homo_sapiens/UCSC/hg19/Sequence/IlluminaBowtie2Index/genome
bowtie2path=/software/link/bin/bowtie2
samtoolspath=/software/link/bin/samtools
# create output directory if it does not exist
if [ ! -d $outPath ]; then mkdir $outPath; fi
# align the pair-ended sequences
$bowtie2path -p 4 -x $referenceSeq -1 $seqfile1 -2 $seqfile2 -S $readgroup.sam
# convert sam file to bam file; sort and index bam file
$samtoolspath view -F 4 -bS $readgroup.sam > $readgroup.bam
$samtoolspath sort $readgroup.bam $readgroup.sorted
$samtoolspath index $readgroup.sorted.bam


run_bowtie2_heart.pbs - emacs@ln
File Edit Options Buffers Tools Help
# set the file and program paths
seqPath=~/CRI_training/Ex4
outPath=~/CRI_training/Ex4/bowtie2
seqfile1=$seqPath/heart_ERR030886.sample.1.fastq.gz
seqfile2=$seqPath/heart_ERR030886.sample.2.fastq.gz
readgroup=$outPath/heart_ERR030886.sample
referenceSeq=/group/referenceFiles/Homo_sapiens/UCSC/hg19/Sequence/IlluminaBowtie2Index/genome
bowtie2path=/software/link/bin/bowtie2
samtoolspath=/software/link/bin/samtools
# create output directory if it does not exist
if [ ! -d $outPath ]; then mkdir $outPath; fi
# align the pair-ended sequences
$bowtie2path -p 4 -x $referenceSeq -1 $seqfile1 -2 $seqfile2 -S $readgroup.sam
# convert sam file to bam file; sort and index bam file
$samtoolspath view -F 4 -bS $readgroup.sam > $readgroup.bam
$samtoolspath sort $readgroup.bam $readgroup.sorted
$samtoolspath index $readgroup.sorted.bam

# Submit a batch job

```
> qsub run_bowtie2_heart.pbs

> qstat
```

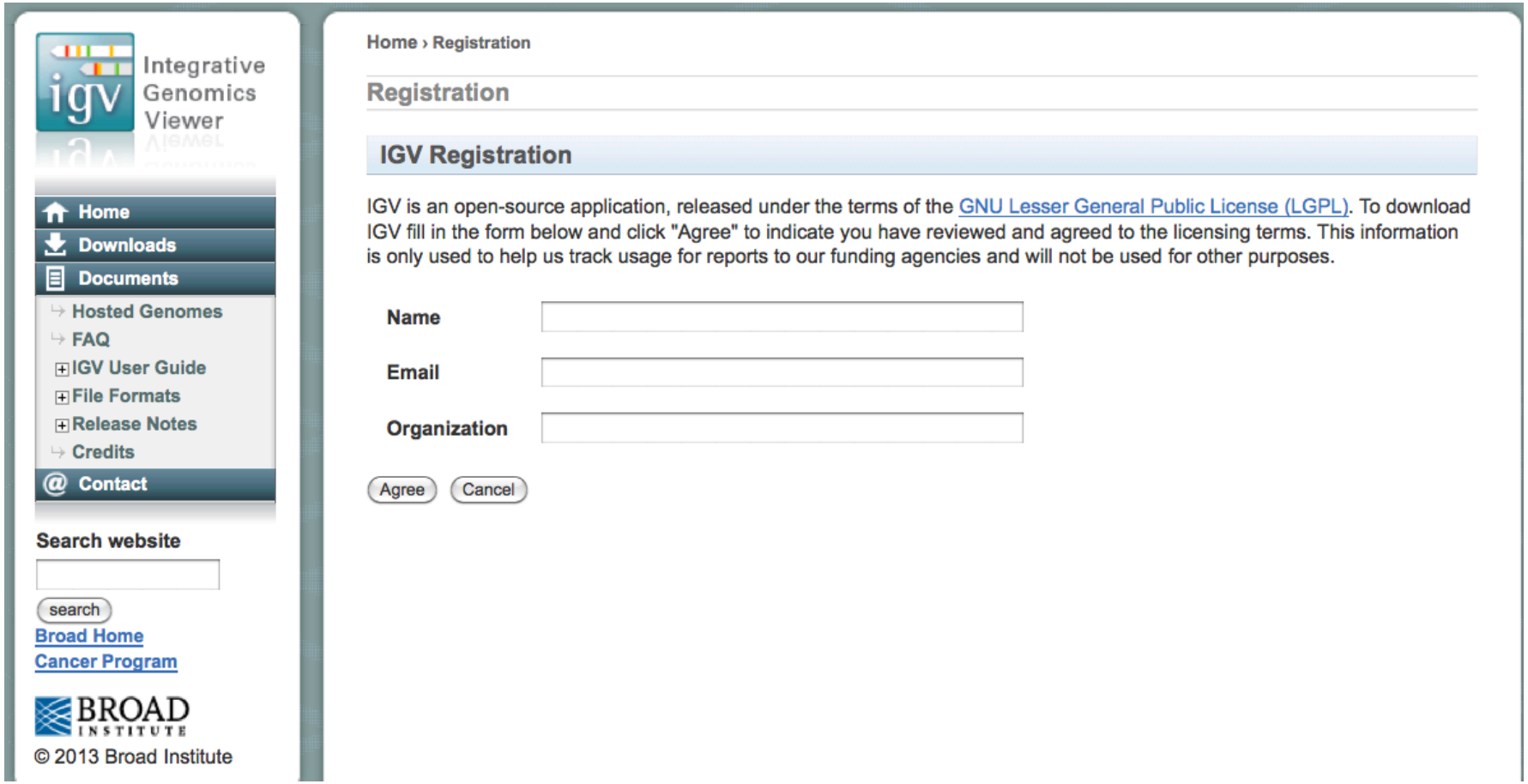
# Output files from alignment

```
> ls ~/CRI_training/Ex4/bwa
heart_ERR030886.sample.1.bwa
heart_ERR030886.sample.bam
heart_ERR030886.sample.sorted.bam
heart_ERR030886.sample.2.bwa
heart_ERR030886.sample.sam
heart_ERR030886.sample.sorted.bam.bai

> ls ~/CRI_training/Ex4/bowtie2
heart_ERR030886.sample.bam
heart_ERR030886.sample.sam
heart_ERR030886.sample.sorted.bam
heart_ERR030886.sample.sorted.bam.bai
```

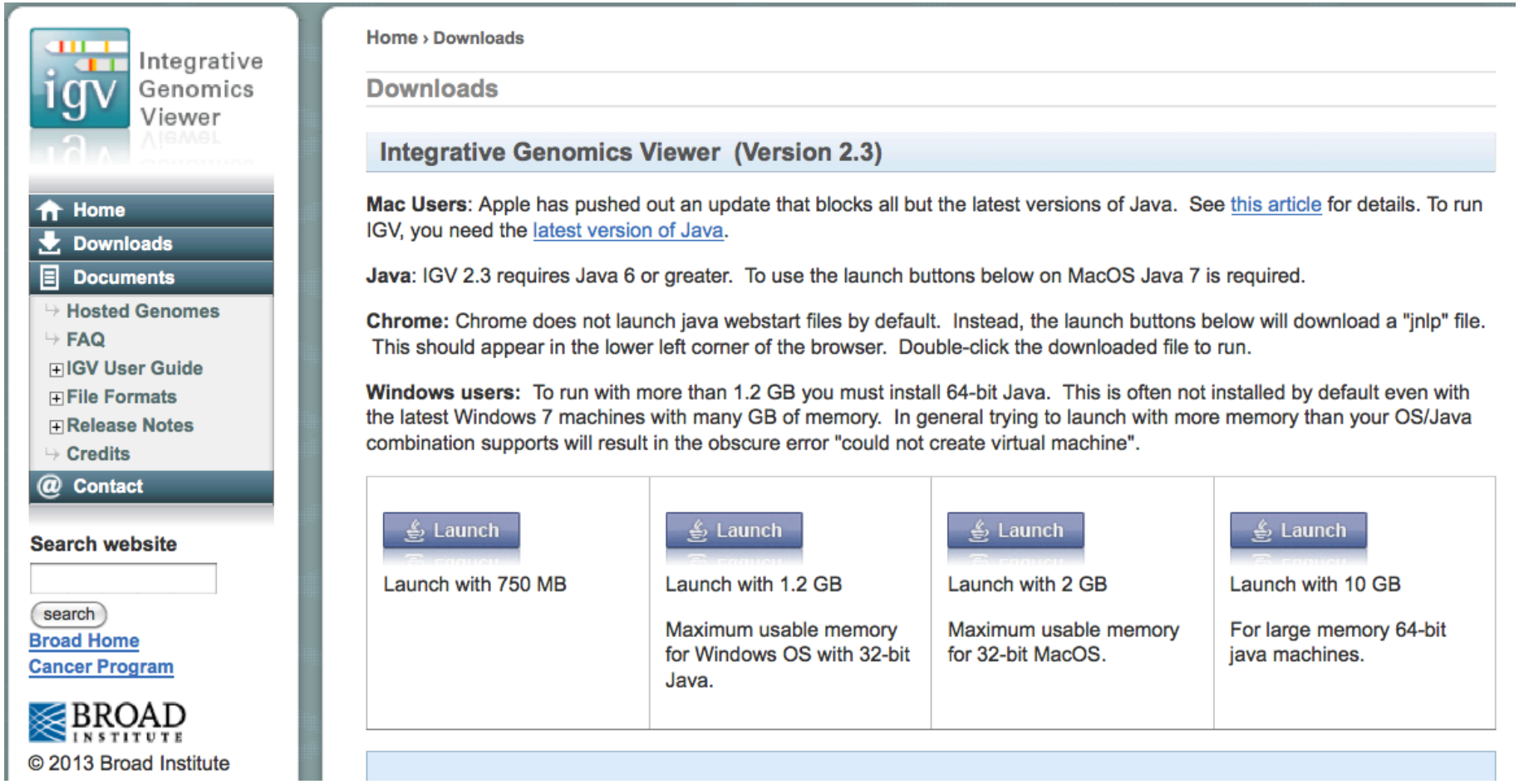
# Extra: Alignment summary statistics

```
> samtools flagstat ~/CRI_training/Ex4/bowtie2/heart_ERR030886.sample.bam
185865 + 0 in total (QC-passed reads + QC-failed reads)
0 + 0 duplicates
185865 + 0 mapped (100.00%:-nan%)
185865 + 0 paired in sequencing
93061 + 0 read1
92804 + 0 read2
153998 + 0 properly paired (82.85%:-nan%)
175828 + 0 with itself and mate mapped
10037 + 0 singletons (5.40%:-nan%)
0 + 0 with mate mapped to a different chr
0 + 0 with mate mapped to a different chr (mapQ>=5)
```

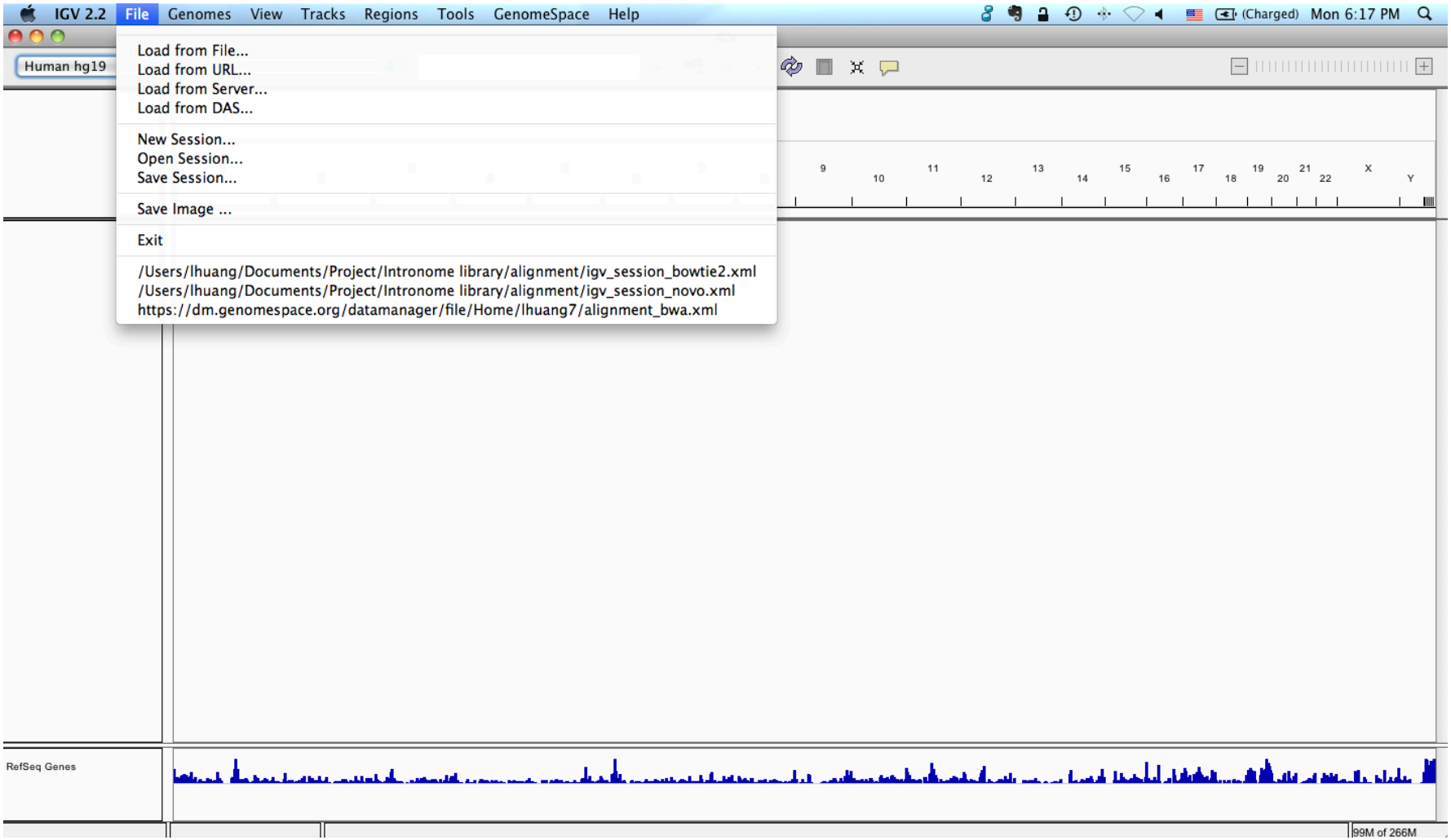
# Download and launch IGV



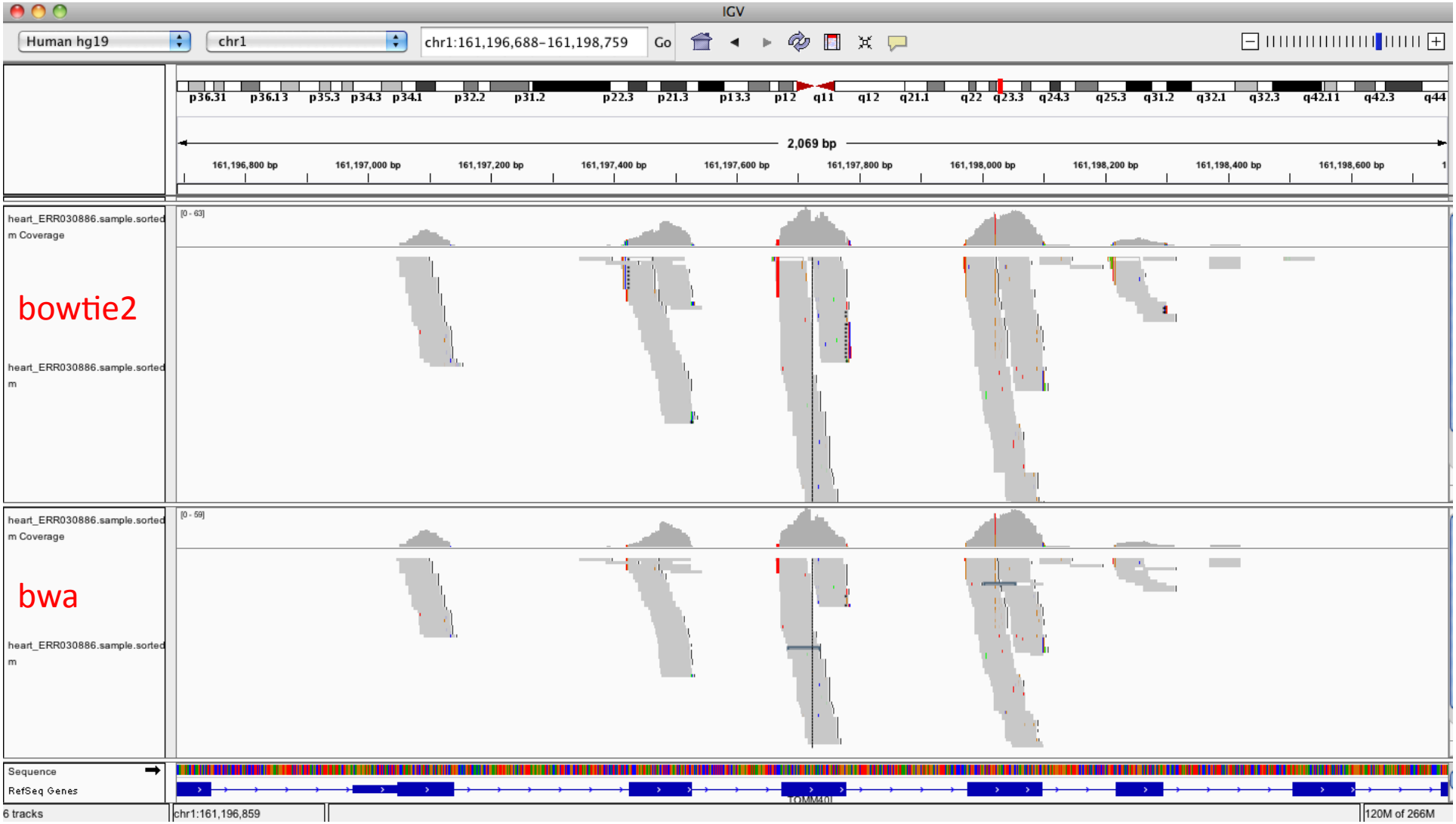
# Download and launch IGV



# Open bam files in IGV

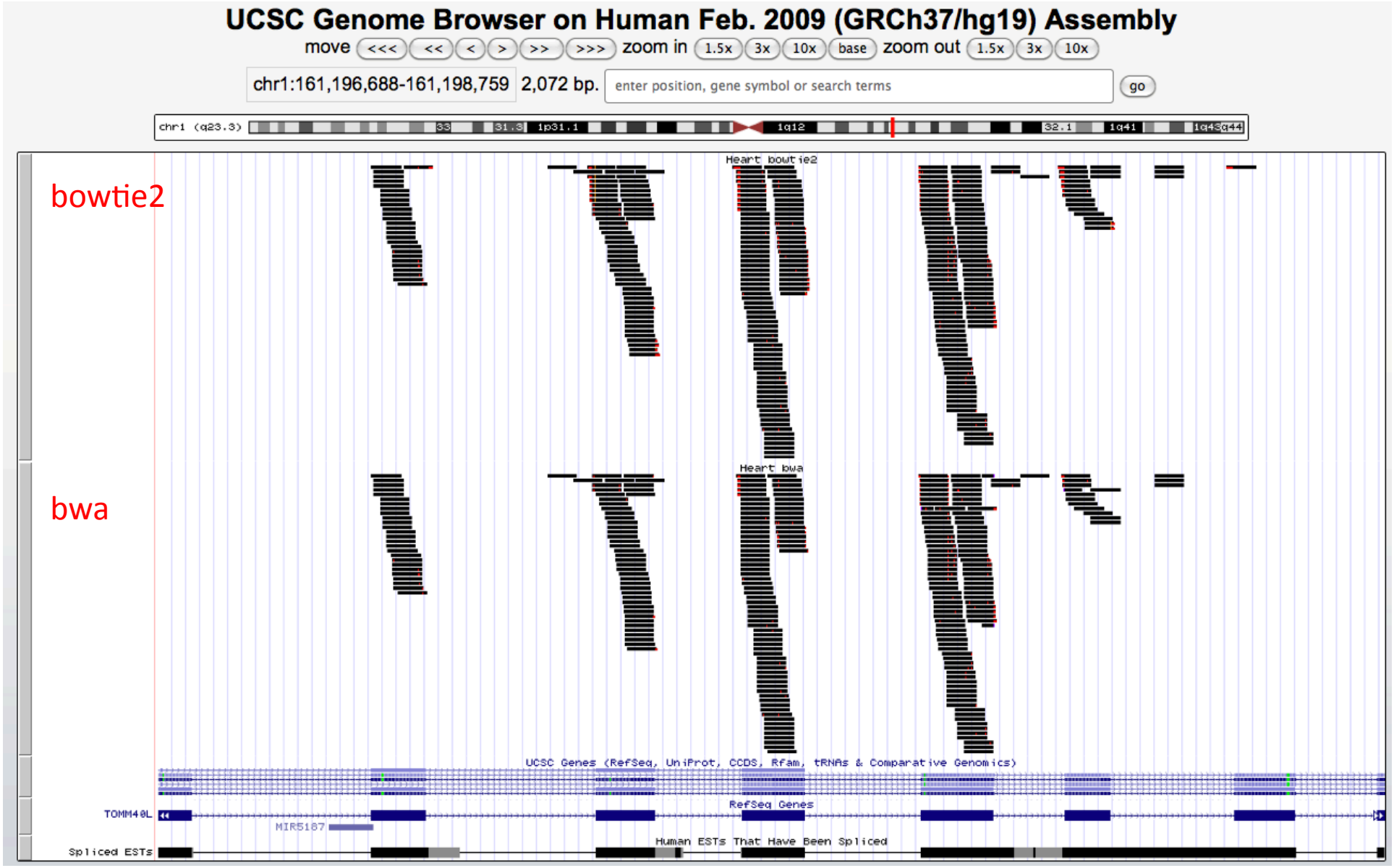


# Visualize the alignment in IGV




TOMM40L (translocase of outer mitochondrial membrane 40 homolog (yeast)-like)

# Visualize the alignment in UCSC Genome Browser



Questions and Comments?

Please fill out a short online survey using the link at the bottom of the wiki page

**Thank you!**