

# High Performance Computing for Genomics

Part II: for Genomics

# Project description

Assume this population study:

We have 2 populations, population 1 has 14 individuals, population 2 only 5.

There is no reference genome available.

Are there 'diagnostic' variations between the 2 populations.

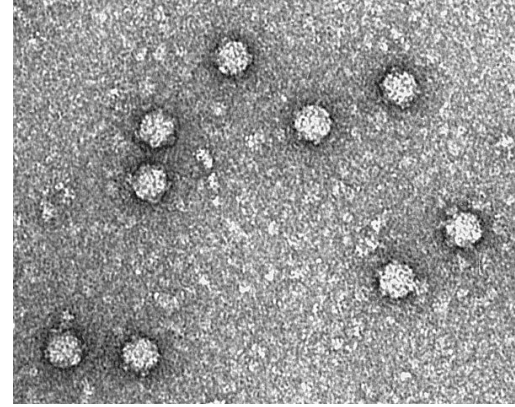
Project parts:

- Assembly
- Mapping
- Variant calling

# Actual data

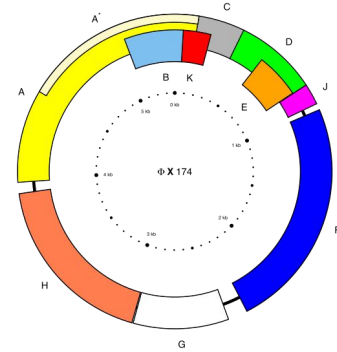
Population 1: Sequenced PhiX in the Genomics Core

Population 2: Simulated data from the ncbi reference genome



Phi X 174:

- Bacteriophage
- First sequenced DNA virus (1977)
- Circular, single stranded DNA genome of 5386bp, GCcontent 44%
- 95% are coding genes, total of 11 genes
- Used as positive control in Illumina sequencing



# Project description

Project parts:

- **Assembly**

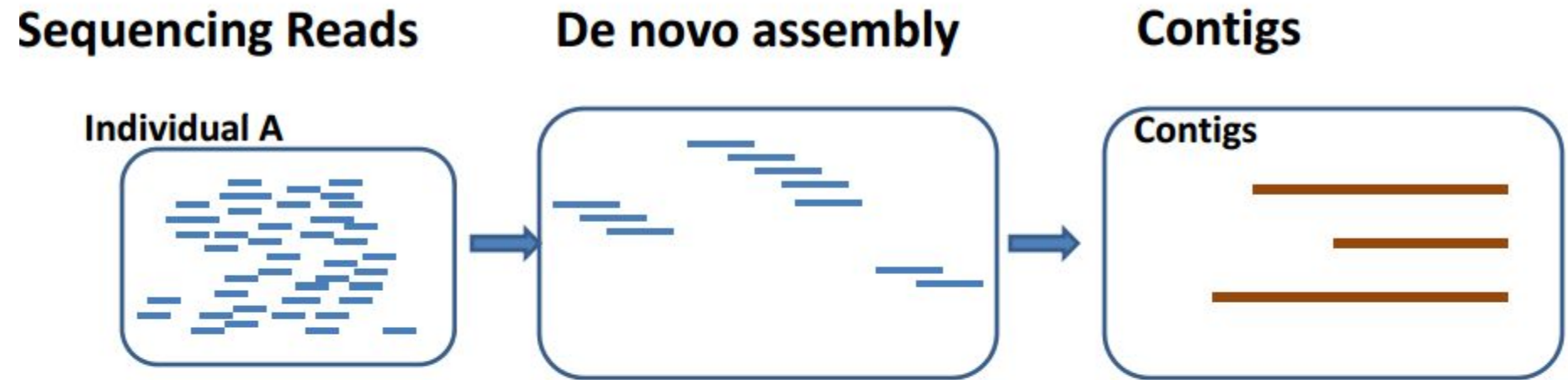
Population 1 has most individuals, so best to choose one of these individuals.

- Mapping
- Variant calling

# Assembly

Simple description: put **overlapping reads** together in order to get a **contig**

Used software: ABySS



Which partition would you use for the assembly?

# Exercise 4

- Load Cerebro
- Load ABySS (in the 2015a toolchain)
- Check the abyss.pbs file, change the header/variables if needed
- Launch the abyss.pbs script
- Check the output of the assembly

# Project description

Project parts:

- Assembly
- **Mapping**  
19 individuals need to be mapped to the newly created reference genome
- Variant calling



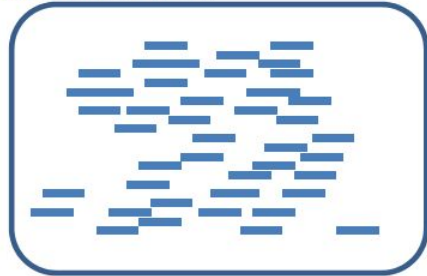
# Mapping

Compare your sequences against a reference in order to get the location

Used software: Bowtie2

## Sequencing Reads

Individual A



Reference Genome



Which partition would you use for the mapping?

# Go to ThinkKing

- List all loaded modules
- Purge
- Load thinking

# Job efficiency

Run every task **separately**, using all available cores in a node

- Multithreading  
Using 20 cores (ivybridge)
- Needs 1 pbs per task
- Is applicable for all tasks,  
but each pbs file needs adjustments

You can do this on your own now :)

Run all tasks in **parallel**

- Single threads  
19 processes + master process (worker)
- Needs 1 pbs in total
- Tasks must be exact the same,  
except for some parameters

This requires some more theory :(

# Parallel Jobs

Assume following pbs script

```
#!/bin/bash -l
#PBS -l nodes=1:ppn=1
#PBS -l walltime=00:15:00
cd $VSC_SCRATCH
map -s sample1 -r homo_sapiens -l 100
```

Now you want to use *'map'* for 100 different samples => 100 tasks

# Parallel Jobs: parameters

The script will be changed to:

```
#!/bin/bash -l
#PBS -l nodes=1:ppn=8
#PBS -l walltime=04:00:00
cd $VSC_SCRATCH
map -s $sample -r $reference -l $length
```

The '*map*' parameters are now variables, these will be listed in a separate file:

```
sample,reference,length
sample1,homo_sapiens,100
sample2,homo_sapiens,50
```

# Parallel Jobs: Parameter file

The parameter file is a simple comma separated value file (CSV)  
Which can be created in any text editor (NOT word)  
or in a spreadsheet program (as excel)

```
sample,reference,length  
sample1,homo_sapiens,100  
sample2,homo_sapiens,50  
sample2,mus_musculus,50  
sample3,homo_sapiens,100  
sample3,homo_sapiens,50
```

# Parallel Jobs: PBS header

## Original file

```
#!/bin/bash -l
#PBS -l nodes=1:ppn=1
#PBS -l walltime=00:15:00
```

- Uses 1 core
- Walltime is 15 minutes  
100 tasks, 15 minutes each  
=> 1500 minutes or 25 hours

## Parallel file

```
#!/bin/bash -l
#PBS -l nodes=1:ppn=8
#PBS -l walltime=04:00:00
```

- 8 cores in use: 7 for the job, 1 for delegating the work
- Walltime is 4 hours  
15 minutes per job, 7 jobs at the time  
=> 215 minutes (1500/7) or 3.56 hours  
So 4 hours is a safe time



# Parallel Jobs: start a job

Load the worker module:

```
$ source switch_to_2015a
$ module load worker
$ wsub -batch run.pbs -data data.csv
```

Use wsub instead of qsub

-batch	The pbs script (.pbs)
-data	The parameter file (.csv)

# Parallel Jobs: Map Reduce

Parallel computations can be abstracted into 3 steps:

1. Preparation
2. The work items
3. Aggregating results, clean-up, ...

The worker framework also supports this:

-prolog	Preparation step	e.g.: copying the reference genome to scratch
-batch	The parallel jobs	e.g.: mapping the reads to the genome
-epilog	Clean-up step	e.g.: cleaning the scratch storage

```
$ wsub -prolog split-data.sh -batch run.pbs -epilog distr.sh -  
data data.csv
```

# Exercise 5a

- Open bowtie\_batch.pbs
- Change script
- Variable name?
- input/output path?
- Reference genome?

# Exercise 5b

- Open prolog script
- What happens?
- Check genome path
- Where is the genome stored?

# Exercise 5c

- Open epilog script
- What happens?

# Exercise 5d

- Start job on thinking
- Check mapping statistics (especially the data for the assembly)

# Project description

Project parts:

- Assembly
- Mapping
- **Variant calling**

The 2 populations have to be called together in order to find common variants (if no variant found, we want to know if it was reference, or not seen)

# Homework: do population variant calling

```
source switch_to_2015a
module load freebayes/1.0.2-33-foss-2015a

#set some variables (see other scripts)

GENOME="$GIT_DIR/results/denovo/genome_k31-unitigs.fa";
BAM_DIR="$GIT_DIR/results/mapping";
FREEBAYES_OPTIONS="-m 20 -q 15 --ploidy 2 ";
FREEBAYES_OUTPUT_FILE_NAME="freebayes.m20.q15.ploidy2.vcf";

#copy the GENOME to $VSC_SCRATCH_NODE/genome/genome.fa
#copy the bam files to the $VSC_SCRATCH_NODE/bams directory
bamfiles="";
for i in `ls -1 -d $VSC_SCRATCH_NODE/bams/*bam`;
do
    bamfiles="$bamfiles $i";
done

#do the variant calling
freebayes --fasta-reference $SCRATCH_DIR/genome/genome.fa $FREEBAYES_OPTIONS $bamfiles > $SCRATCH_DIR/$FREEBAYES_OUTPUT_FILE_NAME;

#copy the data
```