1. Running FastQC on raw data

- Reminder to take notes, share info here: Etherpad
- FastQC is a program that can quickly scan your raw data to help figure out if there are adapters or low quality reads present. Create a job file to run FastQC on one of the fastq files here: `/data/genomics/dikowr/SMSC/fastq_for_fastqc`
    - **module**: `bioinformatics/fastqc/0.11.5`
    - **command**: `fastqc <FILE.fastq>`
    - after your job finishes, find the results and download some of the images, e.g. `per_base_quality.png` to your local machine using `scp` , CyberDuck, or Filezilla.

2. Trimming adapters with TrimGalore!

- PacBio data will be error-corrected solely by the assembler, but Illumina data trimming and thinning are common.
- Most assemblers these days don't want you to trim/thin for quality before assembling, but trimming is important for downstream applications. TrimGalore will auto-detect what adapters are present and remove very low quality reads (quality score <20) by default.
- Create a job file to trim adapters and very low quality reads for the Illumina data here: `/data/genomics/dikowr/SMSC/fastq_for_fastqc`
    - **command**: `trim_galore --paired --retain_unpaired <FILE_1.fastq> <FILE_2.fastq>`
    - **module**: `bioinformatics/trimgalore/0.4.0`
    - You can then run FastQC again to see if anything has changed.

3. Run Genomescope

- Genomescope can be used to estimate genome size from short read data:
    - Genomescope
- To run Genomescope, first you need to generate a Jellyfish histogram.
- You'll need two job files for Jellyfish, one to count the kmers and the second to generate a histogram to give to Genomescope:
- Here is a copy of the Red Siskin Illumina data: `/data/genomics/dikowr/SMSC/Illumina_all`

- Hint: don't copy these data to your own space.

- First job file: kmer count:
  - Module: `bioinformatics/jellyfish/2.2.3`
  - Commands: `jellyfish count -C -m 21 -t $NSLOTS -s 800000000 *.fastq -o reads.jf`
  - `-m` = kmer length
  - `-s` = RAM
  - `-C` = "canonical kmers" don't change this
  - Hint: this job needs to run on the high memory queue.

- This will take a while, so we can move on and then come back to it when it finishes.

- Second job file: histogram:
  - Module: `bioinformatics/jellyfish/2.2.3`
  - Commands: `jellyfish histo -t $NSLOTS reads.jf > reads.histo`

- Download the histogram to your computer, and put it in the Genomescope webservice: [Genomescope]

4. Run the fasta metadata parser to get statistics about the PacBio data

We use a python script to grab some statistics from assembly files, but we can also use it to look at our PacBio data. These are the stats it produces:

Total number of base pairs:
Total number of contigs:
N90:
N80:
N70:
N60:
N50:
L90:
L80:
L70:
L60:
L50:
GC content:
Median contig size:
Mean contig size:

Longest contig is:
Shortest contig is: + Module: `bioinformatics/bioinformatics/fastametadata/1.0` + Commands: `fasta_meta_data_parser.py <PACBIO.fasta> > pacbio_stats.out` + The PacBio data are here: `/data/genomics/dikowr/SMSC/PacBio/all_pacbio.fasta`

- How long is the longest read?
- What is the read N50?

5. Setting up MaSuRCA Illumina + PacBio Hybrid Assembly

- We are going to split up into three groups of 5 people each to submit whole genome assembly jobs. These will take a while and create a lot of large files.
- MaSuRCA runs with 2 job files. The first uses a configuration file to generate an sh script called assemble.sh. Then you just execute the sh script to complete the actual assembly.
- Here is a sample MaSuRCA config file that you will need to copy to your space and modify: `/data/genomics/dikowr/SMSC/masurca_config.txt`
- * Edit the file so that it points to your files and familiarize yourself with the parts.
  - Reminder, the raw data are here: `/data/genomics/dikowr/SMSC/Illumina_all` and `/data/genomics/dikowr/SMSC/PacBio`
  - Do not copy the data to your own space (look how big the files are!)

- To keep things tidy, create a directory for the MaSuRCA assembly in your space.
  - Hint: use `mkdir`

- Create a job file for this first part of MaSuRCA.

  - **Queue:** Short, high CPU
  - **Threads & RAM:** single thread, 2GB RAM
  - **Module:** `module load bioinformatics/masurca/3.2.8`
  - **Commands:** `masurca <CONFIG_FILE>`

- This job should complete in a few seconds and result in a file called assemble.sh
- Create a second job file for the second part of MaSuRCA.

  - **Queue:** Long, himem
  - **Threads & RAM:** 16 threads, 30GB RAM each
  - **Module:** `module load gcc/4.9.2`
  - **Command:** `./assemble.sh`

- Submit this second job.

6. Run the fasta metadata parser to get statistics about the assembly

- I have put a finished assembly here: `/data/genomics/dikowr/SMSC/finished_assembly`
    - Module: `bioinformatics/bioinformatics/fastametadata/1.0`
    - Commands: `fasta_meta_data_parser.py <ASSEMBLY> > assembly_stats.out`
- How long is the longest read?
- What is the read N50?