Genome Assembly pipeline

This pipeline is intended to be a standard procedure to assembly genomes. We developed this pipeline to be optimal while using two kinds of data:

- PacBio sequel V2 genomic data
- Illumina genomic data

Step 0: Genome size prediction

Kmergenie estimates the best k-mer length for genome de novo assembly and it predicts the genome size. This is a useful tool in de novo genome assemblies when the genome size is unknown.

Installation:

```
module load anaconda2
conda create -n kmergenie
source activate kmergenie
conda install r-base
conda install -c bioconda kmergenie
source deactivate
```

Kmergenie needs a list with the link to illumina fasta files:

```
cd /data/SBCS-MartinDuranLab/03-Giacomo/data/osedax/illumina/
readlink -f OSE_DAX-1_S55_L003_R* > list_for_kmergenie
```

Code to run the command:

```
#!/bin/bash
#$ -wd /data/SBCS-MartinDuranLab/03-Giacomo/logs/kmergenie/
#$ -j y
#$ -pe smp 6
#$ -l h_vmem=8G
```

```
#$ -l h_rt=240:0:0

module load anaconda2
source activate kmergenie #load the environment

kmergenie \
 /data/SBCS-MartinDuranLab/03-Giacomo/data/osedax/illumina/list_for_kmergenie \
 --diploid \
 -o osedax_histograms
```

Results will be in `/data/SBCS-MartinDuranLab/03-Giacomo/logs/kmergenie/`.
There will be an `.html` file that will contain the genome size prediction.

Step 1: Assembly

Step 1.1: Assemblers

The core of this pipeline is a genome assembler software such as **Canu** or **Flye**
We tried three different approaches and then we evaluated the best one. First we runned Flye, then we launched Canu and finally we runned Flye using the corrected reads obtained with Canu.

Flye installation:
Installing Flye de novo assembler:

```
cd
module load anaconda2
conda create -n Flye python=2.7 anaconda
conda install flye
```

Canu installation:

- Download the latest release on your computer

```
scp canu-1.8.Linux-amd64.tar.xz btx604@login.hpc.qmul.ac.uk:/data/SBCS-MartinDuranLab/03-Giacomo/src
```

```
cd /data/SBCS-MartinDuranLab/03-Giacomo/src
tar -xJf canu-1.8.Linux-amd64.tar.xz
```

Converting subreads.bam to fastq with BAM2fastx tools:
installation:

```
module load anaconda2
source activate Flye
conda install bam2fastx
```

Code to run BAM2fastx tools:

```
#!/bin/bash
#$ -wd /data/home/btx604/scripts/envs_jobs/Flye/
#$ -j y
#$ -pe smp 4
#$ -l h_vmem=1G
#$ -l h_rt=72:0:0

module load anaconda2
source activate Flye #load the environment
bam2fastq -o /data/SBCS-MartinDuranLab/03-Giacomo/raw_data/PacBio/01-Oasisia/Oasisia_pb_raw /data/SBCS-MartinDuranLab/03-Giacomo/raw_data/PacBio/01-Oasisia/data2/pb/r64044_20190812_215729/1_A01/m64044_190812_220643.subreads.bam
```

Code to run Flye:

```
#!/bin/bash
#$ -wd /data/SBCS-MartinDuranLab/03-Giacomo/raw_data/PacBio/01-Oasisia/qsub_logs/
#$ -j y
#$ -pe smp 48
#$ -l h_vmem=10G
```

```
#$ -l h_rt=240:0:0
#$ -l highmem

module load anaconda2
source activate Flye #load the environment

flye \
 --pacbio-raw /data/SBCS-MartinDuranLab/03-Giacomo/raw_data/PacBio/01-Oasisia/Oasisia_pb_raw.fastq.gz \
 --genome-size 1g \
 --threads 48 \
 --asm-coverage 40 \
 --resume \
 --out-dir /data/scratch/btx604/Oasisia/flye/
```

Code to run Canu (without using the grid):

```
#!/bin/bash
#$ -wd /data/SBCS-MartinDuranLab/03-Giacomo/raw_data/PacBio/01-Oasisia/qsub_logs/
#$ -j y
#$ -pe smp 48
#$ -l h_vmem=10G
#$ -l h_rt=240:0:0
#$ -l highmem

module load gnuplot
module load java/1.8.0_152-oracle

/data/SBCS-MartinDuranLab/03-Giacomo/bin/canu \
 -p Oasisia -d /data/scratch/btx604/Oasisia/canu \
 genomeSize=1g \
```

```
 maxMemory=480g \
 maxThreads=48 \
 useGrid=false \
 gridEngineResourceOption="-l h_vmem=MEMORY -pe smp THREADS" \
 batOptions="-dg 3 -db 3 -dr 1 -ca 500 -cp 50" \
 -pacbio-raw /data/SBCS-MartinDuranLab/03-Giacomo/raw_data/PacBio/01-Oasisia/Oasisia_pb_raw.fastq.gz
```

Code to run Canu (using the grid): ????????

- First a Canu script needs to be modified accordingly to the cluster requirements: in the script `/data/SBCS-MartinDuranLab/03-Giacomo/src/canu-1.8/Linux-amd64/lib/site_perl/canu/Grid_SGE.pm` : change `#"-pe threads THREADS");` with `#"-pe smp THREADS");` and `#"-l mem=MEMORY");` with `#"-l h_vmem=MEMORY");`

```
cd

module load gnuplot
module load java/1.8.0_152-oracle

/data/SBCS-MartinDuranLab/03-Giacomo/bin/canu \
 -p Oasisia -d /data/scratch/btx604/Oasisia/canu_V8 \
 -pacbio-raw /data/SBCS-MartinDuranLab/03-Giacomo/raw_data/PacBio/01-Oasisia/Oasisia_pb_raw.fastq.gz \
 genomeSize=1g \
 useGrid=true \
 gridEngineResourceOption="-l h_vmem=MEMORY -pe smp THREADS" \
 gridOptions="-l h_rt=72:0:0 -j y -l highmem" \
 "batOptions=-dg 3 -db 3 -dr 1 -ca 500 -cp 50" \
 "gridOptionsGFA=-l h_vmem=300G" \
 "gridOptionsovb=-l h_vmem=120G" \
 "gridOptionsovs=-l h_vmem=120G" \
```

```
 "gridOptionscorovl=-l h_vmem=30G" \
 "gridOptionscor=-l h_vmem=30G" \
 "gridOptionscns=l h_vmem=40G" \
 gnuplot=$(which gnuplot) \
 java=$(which java)
 > /data/scratch/btx604/Oasisia/canu_V8/LOG/LOG 2>&1
```

- it may happen that Canu crash before starting the assembly, in this case the assembly can be launched with a separated off grid job:

```
#!/bin/bash
#$ -wd /data/scratch/btx604/Oasisia/canu_V8
#$ -o /data/scratch/btx604/Oasisia/canu_V8
#$ -j y
#$ -pe smp 48
#$ -l h_vmem=10G
#$ -l h_rt=240:0:0
#$ -l highmem

module load gnuplot
module load java/1.8.0_152-oracle

/data/SBCS-MartinDuranLab/03-Giacomo/bin/canu \
 -p Oasisia -d /data/scratch/btx604/Oasisia/canu_V8 \
 genomeSize=750m \
 maxMemory=480g \
 maxThreads=48 \
 useGrid=false \
 gridEngineResourceOption="-l h_vmem=MEMORY -pe smp THREADS" \
 batOptions="-dg 3 -db 3 -dr 1 -ca 500 -cp 50" \
 -pacbio-raw /data/SBCS-MartinDuranLab/03-Giacomo/raw_data/PacBio/01-Oasisia/Oasisia_pb_raw.fastq.gz
```

Code to run Flye using Canu corrected reads:

```
#!/bin/bash
#$ -wd /data/SBCS-MartinDuranLab/03-Giacomo/raw_data/PacBio/01-Oasisia/qsub_logs/
#$ -j y
#$ -pe smp 48
#$ -l h_vmem=10G
#$ -l h_rt=240:0:0
#$ -l highmem

module load anaconda2
source activate Flye #load the environment

flye \
 --pacbio-corr /data/SBCS-MartinDuranLab/03-Giacomo/01-OasisiaData/Oasisia.correctedReads.fasta.gz \
 --genome-size 1g \
 --threads 48 \
 --asm-coverage 40 \
 --out-dir /data/scratch/btx604/Oasisia/flye_corrected_reads_v2/
```

Step 1.2: Quality check

Busco provides quantitative measures for the assessment of genome assembly, gene set, and transcriptome completeness, based on evolutionarily-informed expectations of gene content from near-universal single-copy orthologs selected from OrthoDB *v9*

Code to run Busco on the Flye assembly:

```
#!/bin/bash
#$ -wd /data/SBCS-MartinDuranLab/03-Giacomo/logs/busco
```

```
#$ -pe smp 4
#$ -l h_vmem=2G
#$ -l h_rt=10:0:0
#$ -cwd
#$ -j y

cd /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/busco/flye/

module load busco/3.0
module load augustus

export AUGUSTUS_CONFIG_PATH=/data/SBCS-MartinDuranLab/02-Chema/src/Augustus/config

BUSCO.py -i /data/scratch/btx604/Oasisia/flye/assembly.fasta -m genome -o Oasisia_Flye_Busco -c 4 -l /data/SBCS-MartinDuranLab/00-BlastDBs/metazoa_odb9
```

Code to run Busco on the Canu assembly:

```
#!/bin/bash
#$ -wd /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/busco/canu/
#$ -pe smp 4
#$ -l h_vmem=2G
#$ -l h_rt=10:0:0
#$ -cwd
#$ -j y

cd /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/busco/canu/

module load busco/3.0
```

```
module load augustus

export AUGUSTUS_CONFIG_PATH=/data/SBCS-MartinDuranLab/02-Chema/src/Augustus/config

BUSCO.py -i /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/canu/Oasisia.contigs.fasta -m genome -o Oasisia_Canu_Busco -c 4 -l /data/SBCS-MartinDuranLab/00-BlastDBs/metazoa_odb9
```

Code to run Busco on the Flye (corrected reads) assembly:

```
#!/bin/bash
#$ -wd /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/busco/canu/
#$ -pe smp 4
#$ -l h_vmem=2G
#$ -l h_rt=10:0:0
#$ -cwd
#$ -j y

cd /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/busco/canu/

module load busco/3.0
module load augustus

export AUGUSTUS_CONFIG_PATH=/data/SBCS-MartinDuranLab/02-Chema/src/Augustus/config

BUSCO.py -i /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/flye_corrected_reads_v2/assembly.fasta -m genome -o Oasisia_Flye_corrected_reads_Busco -c 4 -l /data/SBCS-MartinDuranLab/00-BlastDBs/metazoa_odb9
```

Step 1.3: Assemblies comparison

QUAST quality assessment tool evaluates and compares genome assemblies. This tool improves on leading assembly comparison software with quality metrics. QUAST can evaluate assemblies both with a reference genome, as well as without a reference. QUAST produces many reports, summary tables and plots.

Installation:

```
cd
module load anaconda2
conda create -n quast
source activate quast
conda install -c bioconda quast
source deactivate
```

Code to run Quast:

```
#!/bin/bash
#$ -wd /data/SBCS-MartinDuranLab/03-Giacomo/logs/quast
#$ -pe smp 4
#$ -l h_vmem=2G
#$ -l h_rt=10:0:0
#$ -cwd
#$ -j y

module load anaconda2
source activate quast

quast \
 /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/canu/Oasisia.contigs.fasta \
 /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/flye/assembly.fasta \
 /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/flye_correcte
```

```
d_reads_v2/assembly.fasta \
 -o /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/quast_canu_flye_flye \
 --eukaryote \
```

Step 2: Polishing

Raw PacBio reads and Illumina paired-end data will be aligned to the initial assembly using **Bwa** and **Pbmm2** and casual errors occurred during the PacBio sequencing will be corrected with **Arrow** and **Racon**. This step includes many small tasks, therefore it could be easier to use a wrapper script for all the jobs.

Step 2.1: Preparing the PacBio data

Seqtk to convert PacBio subreads from fastq.gz to fasta

Code to run Seqtk:

```
#!/bin/bash
#$ -wd /data/home/btx604/scripts
#$ -j y
#$ -o /data/SBCS-MartinDuranLab/03-Giacomo/logs/various
#$ -pe smp 1
#$ -l h_vmem=10G
#$ -l h_rt=24:0:0

module load seqtk

seqtk seq -a /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/00-pacbio/oasisia_pb_raw.fastq.gz > /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/00-pacbio/oasisia_pb_raw.fa
```

Samtools faidx to index the fasta assembly, therefore making it accessible to Arrow

Code to run Samtools:

```
#!/bin/bash
#$ -wd /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/canu/
#$ -o /data/SBCS-MartinDuranLab/03-Giacomo/logs/samtools/
#$ -j y
#$ -pe smp 1
#$ -l h_vmem=12G
#$ -l h_rt=24:0:0

module load samtools/1.9

samtools faidx /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia
/canu/Oasisia.contigs.fasta
```

Step 2.2: Polishing with PacBio data

In order to polish our assembly using PacBio data we need to run two rounds of a combination of different softwares such as pbmn2 (that aligns the raw reads to the assembly), samtools and **pbindex** (that generate all the indexes that are needed to run Arrow), and Arrow (which compare the assembly with the raw reads and fix the errors occurred during the sequencing)

First we need to install pbmn2, pbindex (pbbam) and Arrow

```
module load anaconda2
conda create -n genomePolishing python=2.7 anaconda
source activate genomePolishing
conda install -c bioconda pbmm2
conda install -c bioconda pbgcpp
conda install pbbam
```

First Round:
Align raw PB reads to the assembly in a bam file
Code to run pbmm2:

```
#!/bin/bash
#$ -wd /data/scratch/btx604/Oasisia/genomePolishing/pbalign/step1
#$ -j y
#$ -o /data/SBCS-MartinDuranLab/03-Giacomo/logs/genomePolishing
#$ -pe smp 5
#$ -l h_vmem=8G
#$ -l h_rt=72:0:0

cd /data/scratch/btx604/Oasisia/genomePolishing/pbalign/step1

module load anaconda2
source activate genomePolishing

pbmm2 align \
 /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/canu/Oasisia.contigs.fasta  \
 /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/00-pacbio/data2/pb/r64044_20190812_215729/1_A01/m64044_190812_220643.subreads.bam \
 /data/scratch/btx604/Oasisia/genomePolishing/pbalign/step1/oasisia_pbalign_step1.bam
```

- Sorted output can be generated using `--sort`

Sorting and indexing the bam alignments
Code to run pbmm2:

```
#!/bin/bash
#$ -wd /data/scratch/btx604/Oasisia/genomePolishing/pbalign/step1/
#$ -o /data/SBCS-MartinDuranLab/03-Giacomo/logs/samtools/
#$ -j y
#$ -pe smp 1
```

```
#$ -l h_vmem=12G
#$ -l h_rt=24:0:0

module load samtools/1.9

samtools sort /data/scratch/btx604/Oasisia/genomePolishing/pbalign/step1/oasisia_pbalign_step1.bam -o /data/scratch/btx604/Oasisia/genomePolishing/pbalign/step1/oasisia_sorted_step1.bam
samtools index /data/scratch/btx604/Oasisia/genomePolishing/pbalign/step1/oasisia_sorted_step1.bam
```

Further indexing of the bam file
Code to run pbindex:

```
#!/bin/bash
#$ -wd /data/scratch/btx604/Oasisia/genomePolishing/pbalign/step1
#$ -j y
#$ -o /data/SBCS-MartinDuranLab/03-Giacomo/logs/genomePolishing
#$ -pe smp 1
#$ -l h_vmem=20G
#$ -l h_rt=120:0:0

module load anaconda2
source activate genomePolishing

cd /data/scratch/btx604/Oasisia/genomePolishing/pbalign/step1/

pbindex /data/scratch/btx604/Oasisia/genomePolishing/pbalign/step1/oasisia_sorted_step1.bam
```

Finally launch Arrow, the polishing tool
Code to run Arrow:

```
#!/bin/bash
#$ -wd /data/scratch/btx604/Oasisia/genomePolishing/pbalign/step1
#$ -j y
#$ -o /data/SBCS-MartinDuranLab/03-Giacomo/logs/genomePolishing
#$ -pe smp 4
#$ -l h_vmem=20G
#$ -l h_rt=120:0:0

module load anaconda2
source activate genomePolishing

cd /data/scratch/btx604/Oasisia/genomePolishing/pbalign/step1

arrow /data/scratch/btx604/Oasisia/genomePolishing/pbalign/step1/oasisia_sorted_step1.bam \
 -r /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/canu/Oasisia.contigs.fasta \
 -o oasisia_step1_variants.gff \
 -o oasisia_step1_consensus.fasta \
 -o oasisia_step1_consensus.fastq \
 -j 4
```

Second Round:
First we need to generate another .fai index of the oasisia_step1_consensus.fasta file
Code to run Samtools:

Align raw PB reads to the fasta output of round1 Arrow
Code to run pbmm2:

Sorting and indexing the bam alignments
Code to run pbmm2:

Further indexing of the bam file
Code to run pbindex:

Launch Arrow for the second time
Code to run Arrow:

Preparing a wrapper script:
create folders: /data/scratch/btx604/Oasisia/genomePolishing/PB

```
/data/scratch/btx604/Oasisia/genomePolishing/PB/
/data/scratch/btx604/Oasisia/genomePolishing/PB/step1/
/data/scratch/btx604/Oasisia/genomePolishing/PB/step2/
/data/scratch/btx604/Oasisia/genomePolishing/PB/pacbio_reads/
/data/scratch/btx604/Oasisia/genomePolishing/PB/reference_genome/
```

Wrap script for the PacBio polishing phase (this is for osedax!)

```
#!/bin/bash
#$ -wd /data/scratch/btx604/osedax/genomePolishing/PB
#$ -j y
#$ -o /data/scratch/btx604/osedax/genomePolishing/PB
#$ -pe smp 8
#$ -l h_vmem=10G
#$ -l h_rt=240:0:0

#seqtk variables
raw_pb_reads_fq=/data/SBCS-MartinDuranLab/03-Giacomo/data/osedax/00-pacbio/osedax_pb_raw.fastq.gz
raw_pb_reads_fa=/data/scratch/btx604/osedax/genomePolishing/PB/pacbio_reads/osedax_pb_raw.fa
#faidx variables
reference_genome_step1=/data/scratch/btx604/osedax/genomePolishi
```

```
ng/PB/reference_genome/Osedax.contigs.fasta
#pbmm2 step1 variables
pb_subreads_bam=/data/SBCS-MartinDuranLab/03-Giacomo/data/osedax/00-pacbio/data2/pb/r64044_20191025_002245/2_B01/m64044_191025_155828.subreads.bam
alignment_step1=/data/scratch/btx604/osedax/genomePolishing/PB/step1/osedax_pbmm2_step1.bam
#samtools step1 variables
alignment_step1_sorted=/data/scratch/btx604/osedax/genomePolishing/PB/step1/osedax_sorted_step1.bam
#arrow step1 variables
step1_variants=/data/scratch/btx604/osedax/genomePolishing/PB/step1/osedax_step1_variants.gff
step1_consensus_fa=/data/scratch/btx604/osedax/genomePolishing/PB/step1/osedax_step1_consensus.fasta
step1_consensus_fq=/data/scratch/btx604/osedax/genomePolishing/PB/step1/osedax_step1_consensus.fastq

#pbmm2 step2 variables
alignment_step2=/data/scratch/btx604/osedax/genomePolishing/PB/step2/osedax_pbmm2_step2.bam
#samtools step2 variables
alignment_step2_sorted=/data/scratch/btx604/osedax/genomePolishing/PB/step2/osedax_sorted_step2.bam
#arrow step2 variables
step2_variants=/data/scratch/btx604/osedax/genomePolishing/PB/step2/osedax_step2_variants.gff
step2_consensus_fa=/data/scratch/btx604/osedax/genomePolishing/PB/step2/osedax_step2_consensus.fasta
step2_consensus_fq=/data/scratch/btx604/osedax/genomePolishing/PB/step2/osedax_step2_consensus.fastq


```

```
module load seqtk
module load samtools/1.9
module load anaconda2
source activate genomePolishing


echo 'SEQTK _____________________________________________________
_____________________'
if [ -e "$raw_pb_reads_fa" ]
then
  echo "$raw_pb_reads_fa found."
else
  seqtk seq -a $raw_pb_reads_fq > $raw_pb_reads_fa
fi

echo 'FAIDX STEP1_____________________________________________
_____________________'
if [ -e /data/scratch/btx604/osedax/genomePolishing/PB/reference
_genome/*.fai ]
then
  echo "/data/scratch/btx604/osedax/genomePolishing/PB/reference
_genome/*.fai found."
else
  samtools faidx $reference_genome_step1
fi

echo 'PBMM2 STEP1_____________________________________________
_____________________'
if [ -e "$alignment_step1" ]
then
  echo "$alignment_step1 found."
```

```
else
  pbmm2 align $reference_genome_step1 $pb_subreads_bam $alignment_step1
fi

echo 'SAMTOOLS STEP1_____________________________________________________'
if [ -e "$alignment_step1_sorted" ]
then
  echo "$alignment_step1_sorted found."
else
  samtools sort $alignment_step1 -o $alignment_step1_sorted
  samtools index $alignment_step1_sorted
fi

echo 'PBINDEX STEP1_____________________________________________________'
if [ -e /data/scratch/btx604/osedax/genomePolishing/PB/step1/*.pbi ]
then
  echo "/data/scratch/btx604/osedax/genomePolishing/PB/step1/*.pbi found."
else
  pbindex $alignment_step1_sorted
fi

echo 'ARROW STEP1_____________________________________________________'
if [ -e "$step1_consensus_fa" ]
then
  echo "$step1_consensus_fa found."
```

```
else
  arrow $alignment_step1_sorted -r $reference_genome_step1 -o $step1_variants -o $step1_consensus_fa -o $step1_consensus_fq -j 8
fi

echo 'FAIDX STEP2_____________________________________________________'
if [ -e /data/scratch/btx604/osedax/genomePolishing/PB/step1/*.fai ]
then
  echo "/data/scratch/btx604/osedax/genomePolishing/PB/step1/*.fai found."
else
  samtools faidx $step1_consensus_fa
fi

echo 'PBMM2 STEP2_____________________________________________________'
if [ -e "$alignment_step2" ]
then
  echo "$alignment_step2 found."
else
  pbmm2 align $step1_consensus_fa $pb_subreads_bam $alignment_step2
fi

echo 'SAMTOOLS STEP2_____________________________________________________'
if [ -e "$alignment_step2_sorted" ]
then
  echo "$alignment_step2_sorted found."
else
```

```
  samtools sort $alignment_step2 -o $alignment_step2_sorted
  samtools index $alignment_step2_sorted
fi

echo 'PBINDEX STEP2_____________________________________________
_____________________'
if [ -e /data/scratch/btx604/osedax/genomePolishing/PB/step2/*.p
bi ]
then
  echo "/data/scratch/btx604/osedax/genomePolishing/PB/step2/*.p
bi found."
else
  pbindex $alignment_step2_sorted
fi

echo 'ARROW STEP2_____________________________________________
_____________________'
if [ -e "$step2_consensus_fa" ]
then
  echo "$step2_consensus_fa found."
else
  arrow $alignment_step2_sorted -r $step1_consensus_fa -o $step2
_variants -o $step2_consensus_fa -o $step2_consensus_fq -j 8
fi
```

Step 2.3: Preparing the Illumina data

Cutadapt is used to remove the adaptors employed during the Illumina sequencing from the raw data

Installation:

```
module load python/3.6.3
pip3 install --user --upgrade cutadapt
```

`cutadapt` will get installed in `~/.local` and can be run like this `~/.local/bin/cutadapt`

Code to run Cutadapt:

```
#!/bin/bash
#$ -cwd
#$ -j y
#$ -o /data/SBCS-MartinDuranLab/03-Giacomo/logs/cutadapt
#$ -pe smp 1
#$ -l h_vmem=12G
#$ -l h_rt=24:0:0

module load python/3.6.3
~/.local/bin/cutadapt \
 -a CTGTCTCTTATACACATCT \
 -A CTGTCTCTTATACACATCT \
 -m 30 \
 -q 15,10 \
 -o /data/scratch/btx604/Oasisia/cutadapt/02/Oasisia_illumina_R1.fastq.gz \
 -p /data/scratch/btx604/Oasisia/cutadapt/02/Oasisia_illumina_R2.fastq.gz \
 /data/SBCS-MartinDuranLab/03-Giacomo/01-OasisiaData/OT_S71_L002_R1_001.fastq.gz \
 /data/SBCS-MartinDuranLab/03-Giacomo/01-OasisiaData/OT_S71_L002_R2_001.fastq.gz
```

- -q 15,10 \ will quality-trim the 5’ end with a cutoff of 15 and the 3’ end with a cutoff of 10.
- -m 30 \ Discard processed reads that are shorter than 30. Reads that are too short even before adapter removal are also discarded. Without this option, reads that have a length of zero (empty reads) are kept in the output.

Quality check of the cutadapted sequences with **FastQC**
Code to run fastQC:

```
#!/bin/bash
#$ -cwd
#$ -j y
#$ -o /data/SBCS-MartinDuranLab/03-Giacomo/logs/fastQC
#$ -pe smp 2
#$ -l h_vmem=1G
#$ -l h_rt=2:0:0

cd /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/fastQC/
module load  fastqc/0.11.5

fastqc /data/scratch/btx604/Oasisia/cutadapt/02/Oasisia_illumina_R1.fastq.gz
fastqc /data/scratch/btx604/Oasisia/cutadapt/02/Oasisia_illumina_R2.fastq.gz
```

Step 2.4: Polishing with Illumina data

Unique wrapper for illumina polishing. needed folders:

```
/data/scratch/btx604/$1/genomePolishing/illumina/raw_reads
/data/scratch/btx604/$1/genomePolishing/illumina/step1
/data/scratch/btx604/$1/genomePolishing/illumina/step2
```

- $1 refers to the species name
- step2_consensus_fa (from PB wrapper) should be in step1
- illumina cutadapted reads named R1.fastq.gz and R2.fastq.gz should be in raw_reads

this script will take an input of the species name as the $1 and should be launched like this:

```
qsub illumina_polishing_wrapper_v1.2_highmem.sh species
```

Before running the following script you need to ask the permission to use a high memory node to the IT service.

illumina_polishing_wrapper_v1.2_highmem.sh

```
#!/bin/bash
#$ -wd /data/scratch/btx604/
#$ -j y
#$ -o /data/scratch/btx604/
#$ -pe smp 24
#$ -l h_vmem=30G
#$ -l h_rt=240:0:0
#$ -l highmem

#bwa index step1 variables
reference_genome_step1_prefix="$1"_step2_consensus.fasta
reference_genome_step1=/data/scratch/btx604/$1/genomePolishing/illumina/step1/$reference_genome_step1_prefix
#bwa mem step1 variables
R1=/data/scratch/btx604/$1/genomePolishing/illumina/raw_reads/R1.fastq.gz
R2=/data/scratch/btx604/$1/genomePolishing/illumina/raw_reads/R2.fastq.gz
alignment_step1_sam=/data/scratch/btx604/$1/genomePolishing/illumina/step1/"$1"_alignment_step1.sam
#samtools view step1 variables
alignment_step1_bam=/data/scratch/btx604/$1/genomePolishing/illumina/step1/"$1"_alignment_step1.bam
#samtools sort index step1 variables
alignment_step1_sorted=/data/scratch/btx604/$1/genomePolishing/illumina/step1/"$1"_sorted_step1.bam
#pilon step1 variables
step1_pilon_prefix="$1"_pilon_step1
```

```
step1_pilon_fa=/data/scratch/btx604/$1/genomePolishing/illumina/step1/"$1"_pilon_step1.fasta

#bwa index step2 variables
reference_genome_step2_prefix="$1"_pilon_step1.fasta
reference_genome_step2=/data/scratch/btx604/$1/genomePolishing/illumina/step2/$reference_genome_step2_prefix
#bwa mem step2 variables
alignment_step2_sam=/data/scratch/btx604/$1/genomePolishing/illumina/step2/"$1"_alignment_step2.sam
#samtools view step2 variables
alignment_step2_bam=/data/scratch/btx604/$1/genomePolishing/illumina/step2/"$1"_alignment_step2.bam
#samtools sort index step2 variables
alignment_step2_sorted=/data/scratch/btx604/$1/genomePolishing/illumina/step2/"$1"_sorted_step2.bam
#pilon step2 variables
step2_pilon_prefix="$1"_pilon_step2
step2_pilon_fa=/data/scratch/btx604/$1/genomePolishing/illumina/step2/"$1"_pilon_step2.fasta

cd /data/scratch/btx604/$1/genomePolishing/illumina/step1

module load bwa
module load samtools/1.9
module load anaconda2
source activate genomePolishing
source activate pilon

echo 'working on '$1

```

```
48 echo 'BWA INDEX STEP1_____________________________________________
   ____________________'
49 if [ -e /data/scratch/btx604/$1/genomePolishing/illumina/step1/
   *.ann ]
50 then
51   echo "/data/scratch/btx604/$1/genomePolishing/illumina/step1/
   *.ann found."
52 else
53   bwa index -p $reference_genome_step1_prefix -a bwtsw $referenc
   e_genome_step1
54 fi
55
56 echo 'BWA MEM STEP1_____________________________________________
   ____________________'
57 if [ -e "$alignment_step1_sam" ]
58 then
59   echo "$alignment_step1_sam found."
60 else
61   bwa mem -t 24 -M $reference_genome_step1 $R1 $R2 > $alignment_
   step1_sam
62 fi
63
64 echo 'SAMTOOLS VIEW STEP1_____________________________________
   ____________________'
65 if [ -e "$alignment_step1_bam" ]
66 then
67   echo "$alignment_step1_bam found."
68 else
69   samtools view -S -b -h $alignment_step1_sam -o $alignment_step
   1_bam
70 fi
71
```

```
echo 'SAMTOOLS SORT INDEX STEP1_____________________________________
_____________________'
if [ -e "$alignment_step1_sorted" ]
then
  echo "$alignment_step1_sorted found."
else
  samtools sort $alignment_step1_bam -o $alignment_step1_sorted
  samtools index $alignment_step1_sorted
fi

echo 'PILON STEP1_____________________________________________________
_____________________'
if [ -e "$step1_pilon_fa" ]
then
  echo "$step1_pilon_fa found."
else
  java -Xmx700G -jar /data/home/btx604/.conda/envs/pilon/share/p
ilon-1.23-2/pilon-1.23.jar --genome $reference_genome_step1 --fr
ags $alignment_step1_sorted --diploid --outdir /data/scratch/btx
604/$1/genomePolishing/illumina/step1 --output $step1_pilon_pref
ix --threads 4
fi

cp $step1_pilon_fa /data/scratch/btx604/$1/genomePolishing/illum
ina/step2/
cd /data/scratch/btx604/$1/genomePolishing/illumina/step2

echo 'BWA INDEX STEP2_____________________________________________
_____________________'
if [ -e /data/scratch/btx604/$1/genomePolishing/illumina/step2/
*.ann ]
then
```

```
  echo "/data/scratch/btx604/$1/genomePolishing/illumina/step2/*.ann found."
else
  bwa index -p $reference_genome_step2_prefix -a bwtsw $reference_genome_step2
fi

echo 'BWA MEM STEP2_____________________________________________________'
if [ -e "$alignment_step2_sam" ]
then
  echo "$alignment_step2_sam found."
else
  bwa mem -t 24 -M $reference_genome_step2 $R1 $R2 > $alignment_step2_sam
fi

echo 'SAMTOOLS VIEW STEP2_____________________________________________'
if [ -e "$alignment_step2_bam" ]
then
  echo "$alignment_step2_bam found."
else
  samtools view -S -b -h $alignment_step2_sam -o $alignment_step2_bam
fi

echo 'SAMTOOLS SORT INDEX STEP2_____________________________________'
if [ -e "$alignment_step2_sorted" ]
then
  echo "$alignment_step2_sorted found."
```

```
else
  samtools sort $alignment_step2_bam -o $alignment_step2_sorted
  samtools index $alignment_step2_sorted
fi

echo 'PILON STEP2_____________________________________________________'
if [ -e "$step2_pilon_fa" ]
then
  echo "$step2_pilon_fa found."
else
  java -Xmx700G -jar /data/home/btx604/.conda/envs/pilon/share/pilon-1.23-2/pilon-1.23.jar --genome $reference_genome_step2 --frags $alignment_step2_sorted --diploid --outdir /data/scratch/btx604/$1/genomePolishing/illumina/step2 --output $step2_pilon_prefix --threads 4
fi
```

Step 2.5: Quality check

Busco and Quast will be used again here. The first software is used to obtain info about the polished version of our genome, while the second is used to compare the assembly at each stage of the polishing providing a detailed review about the changes occurred during this process.

Code to run Busco on the polished assembly:

```
#!/bin/bash
#$ -wd /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/busco/genomePolishing
#$ -o /data/SBCS-MartinDuranLab/03-Giacomo/logs/busco
#$ -pe smp 4
#$ -l h_vmem=2G
#$ -l h_rt=10:0:0
```

```
#$ -j y

cd /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/busco/genomePolishing

module load busco/3.0
module load augustus

export AUGUSTUS_CONFIG_PATH=/data/SBCS-MartinDuranLab/02-Chema/src/Augustus/config

BUSCO.py -i /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/genomePolishing/illumina/oasisia_pilon_step2.fasta -m genome -o oasisia_polished_busco -c 4 -l /data/home/btx604/datasets/metazoa_odb9
```

Code to run Quast:

```
#!/bin/bash
#$ -wd /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/quast/genomePolishing
#$ -o /data/SBCS-MartinDuranLab/03-Giacomo/logs/quast
#$ -pe smp 8
#$ -l h_vmem=10G
#$ -l h_rt=10:0:0
#$ -cwd
#$ -j y

module load anaconda2
source activate quast

quast \
 /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/canu/Oasisia.
```

```
contigs.fasta \
 /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/genomePolishing/PB/oasisia_step1_consensus.fasta \
 /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/genomePolishing/PB/oasisia_step2_consensus.fasta \
 /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/genomePolishing/illumina/oasisia_pilon_step1.fasta \
 /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/genomePolishing/illumina/oasisia_pilon_step2.fasta \
 -o /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/quast/genomePolishing/ \
 --eukaryote \
```

Step 3: Decontamination

BlobTools (1 not 2) is a tool able to determine the taxonomy of each contig in our assembled genome and flag those portions accordingly. This is very useful to identify which sequences truly derive from the target genome and which are are derived from associated microbiome or contaminant organisms.

Step 3.1: Preparing the data

In order to run Blobtools we need to obtain 3 other files: 1. An alignment file, mapping Illumina cutadapted reads against the polished genome, 2. a BlastN dataset, 3. a BlastX dataset.

1-Alignment file

To do first (create a new working directory, copy and rename the polished assembly):

```
mkdir /data/scratch/btx604/oasisia/genomePolishing/illumina/blobtools
cp oasisia_pilon_step2.fasta /data/scratch/btx604/oasisia/genomePolishing/illumina/blobtools
mv oasisia_pilon_step2.fasta oasisia_step2_consensus.fasta
```

Cutadapted R1 and R2 are in /data/scratch/btx604/$1/genomePolishing/illumina

/raw_reads/

Now we can run the following steps all together in one script:
mapping_polished_oasisia_illumina_reads_v1.sh

```
#!/bin/bash
#$ -wd /data/scratch/btx604
#$ -j y
#$ -o /data/scratch/btx604
#$ -pe smp 24
#$ -l h_vmem=10G
#$ -l h_rt=24:0:0
#$ -l highmem

#bwa index step1 variables
reference_genome_step1_prefix="$1"_step2_consensus.fasta
reference_genome_step1=/data/scratch/btx604/$1/genomePolishing/illumina/blobtools/$reference_genome_step1_prefix
#bwa mem step1 variables
R1=/data/scratch/btx604/$1/genomePolishing/illumina/raw_reads/R1.fastq.gz
R2=/data/scratch/btx604/$1/genomePolishing/illumina/raw_reads/R2.fastq.gz
alignment_step1_sam=/data/scratch/btx604/$1/genomePolishing/illumina/blobtools/"$1"_alignment_step1.sam
#samtools view step1 variables
alignment_step1_bam=/data/scratch/btx604/$1/genomePolishing/illumina/blobtools/"$1"_alignment_step1.bam
#samtools sort index step1 variables
alignment_step1_sorted=/data/scratch/btx604/$1/genomePolishing/illumina/blobtools/"$1"_sorted_step1.bam

module load bwa
```

```
module load samtools/1.9

cd /data/scratch/btx604/oasisia/genomePolishing/illumina/blobtools

echo 'BWA INDEX STEP1_____________________________________________'
if [ -e /data/scratch/btx604/$1/genomePolishing/illumina/blobtools/*.ann ]
then
  echo "/data/scratch/btx604/$1/genomePolishing/illumina/blobtools/*.ann found."
else
  bwa index -p $reference_genome_step1_prefix -a bwtsw $reference_genome_step1
fi

echo 'BWA MEM STEP1_____________________________________________'
if [ -e "$alignment_step1_sam" ]
then
  echo "$alignment_step1_sam found."
else
  bwa mem -t 24 -M $reference_genome_step1 $R1 $R2 > $alignment_step1_sam
fi

echo 'SAMTOOLS VIEW STEP1_____________________________________________'
if [ -e "$alignment_step1_bam" ]
then
  echo "$alignment_step1_bam found."
```

```
else
  samtools view -S -b -h $alignment_step1_sam -o $alignment_step1_bam
fi

echo 'SAMTOOLS SORT INDEX STEP1_____________________________________'
if [ -e "$alignment_step1_sorted" ]
then
  echo "$alignment_step1_sorted found."
else
  samtools sort $alignment_step1_bam -o $alignment_step1_sorted
  samtools index $alignment_step1_sorted
fi
```

2-BlastN dataset

Install the NCBI nucleotide database (nt). First we need to create a conda env in order to avoid problems with perl dependencies

```
conda create -n blast
source activate blast
conda install -c bioconda blast
```

Now we can run the code to download the last version of the NCBI nt database (V5-10/02/20)

update_blastdb_5_v1.sh

```
#!/bin/bash
#$ -wd /data/SBCS-MartinDuranLab/03-Giacomo/db/blast_nt_v5
#$ -j y
#$ -o /data/SBCS-MartinDuranLab/03-Giacomo/db/blast_nt_v5
#$ -pe smp 2
#$ -l h_vmem=10G
#$ -l h_rt=24:0:0
```

```
module load anaconda2
source activate blast

cd /data/SBCS-MartinDuranLab/03-Giacomo/db/blast_nt_v5
update_blastdb.pl --blastdb_version 5 nt --decompress
```

- the directory “blast_nt_v5” should be created previously

Finally we can run the code to obtain the blast nt dataset using the polished genome as query:
blastn_oasisia_v2.sh

```
#!/bin/bash
#$ -wd /data/SBCS-MartinDuranLab/03-Giacomo/db/blast_nt_v5/
#$ -o /data/scratch/btx604/oasisia/blobtools/nt_v5
#$ -j y
#$ -pe smp 16
#$ -l h_vmem=20G
#$ -l h_rt=120:0:0
#$ -l highmem

module load anaconda2
source activate blobtools2

cd /data/SBCS-MartinDuranLab/03-Giacomo/db/blast_nt_v5/

export BLASTDB=/data/SBCS-MartinDuranLab/03-Giacomo/db/blast_nt_v5/

blastn -db nt \
 -query /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/genomePolishing/illumina/oasisia_pilon_step2.fasta \
```

```
 -outfmt "6 qseqid staxids bitscore std" \
 -max_target_seqs 10 \
 -max_hsps 1 \
 -evalue 1e-25 \
 -num_threads 16 \
 -out /data/scratch/btx604/oasisia/blobtools/nt_v5/blast.out
```

3-BlastX dataset

Create conda environment

```
module load anaconda2
conda config --add channels bioconda
conda config --add channels conda-forge
conda config --add channels r
cd /data/SBCS-MartinDuranLab/03-Giacomo/src/blobtools2
conda create -n blobtools2 -y python=3.6 docopt pyyaml ujson pysam tqdm nodejs seqtk
source activate blobtools2
conda install -c bioconda -y pysam seqtk
conda install -c bioconda -y blast=2.9 busco diamond minimap2
```

- I think this can be avoided, what I need here is just diamond that can also be loaded with `module load diamond`

Download and format the UniProt reference_proteomes as a diamond database:

```
#!/bin/bash
#$ -wd /data/SBCS-MartinDuranLab/03-Giacomo/src/blobtools2
#$ -o /data/SBCS-MartinDuranLab/03-Giacomo/src/blobtools2
#$ -j y
#$ -pe smp 16
#$ -l h_vmem=3G
#$ -l h_rt=24:0:0

```

```
module load anaconda2
source activate blobtools2

cd /data/SBCS-MartinDuranLab/03-Giacomo/src/blobtools2

mkdir -p uniprot

# fetch latest reference proteome database
wget -q -O uniprot/reference_proteomes.tar.gz \
  ftp.ebi.ac.uk/pub/databases/uniprot/current_release/knowledgebase/reference_proteomes/$(curl \
    -vs ftp.ebi.ac.uk/pub/databases/uniprot/current_release/knowledgebase/reference_proteomes/ 2>&1 | \
    awk '/tar.gz/ {print $9}')

cd uniprot

tar xf reference_proteomes.tar.gz

# extract and concatenate protein FASTA files
touch reference_proteomes.fasta.gz
find . -mindepth 2 | grep "fasta.gz" | grep -v 'DNA' | grep -v 'additional' | xargs cat >> reference_proteomes.fasta.gz

# extract and concatenate taxid mapping files
echo "accession\taccession.version\ttaxid\tgi" > reference_proteomes.taxid_map
zcat */*.idmapping.gz | grep "NCBI_TaxID" | awk '{print $1 "\t" $1 "\t" $3 "\t" 0}' >> reference_proteomes.taxid_map

# make diamond blast db with taxonomic information included
```

```
diamond makedb -p 16 --in reference_proteomes.fasta.gz --taxonmap reference_proteomes.taxid_map --taxonnodes ../taxdump/nodes.dmp -d reference_proteomes.dmnd
```

Code to obtain the blastX dataset using the polished genome as query:
blastx_oasisia_v1.sh

```
#!/bin/bash
#$ -wd /data/scratch/btx604/oasisia/blobtools
#$ -o /data/scratch/btx604/oasisia/blobtools
#$ -j y
#$ -pe smp 16
#$ -l h_vmem=20G
#$ -l h_rt=120:0:0
#$ -l highmem

module load anaconda2
source activate blobtools2

cd /data/scratch/btx604/oasisia/blobtools

diamond blastx \
 --query /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/genomePolishing/illumina/oasisia_pilon_step2.fasta \
 --db /data/SBCS-MartinDuranLab/03-Giacomo/src/blobtools2/uniprot/reference_proteomes.dmnd \
 --outfmt 6 qseqid staxids bitscore qseqid sseqid pident length mismatch gapopen qstart qend sstart send evalue bitscore \
 --sensitive \
 --max-target-seqs 1 \
 --evalue 1e-25 \
 --threads 16 \
 > /data/scratch/btx604/oasisia/blobtools/diamond.out
```

Step 3.2: Blobtools

Blobtools is a command line tool designed to aid genome assembly QC and contaminant/cobiont detection and filtering.

First we need to create a BlobDB data structure based on input files that we have obtained in 3.1

blobtools1_create_oasisia_v2.sh

```
#!/bin/bash
#$ -wd /data/scratch/btx604/oasisia/blobtools/nt_v5
#$ -o /data/scratch/btx604/oasisia/blobtools/nt_v5
#$ -j y
#$ -pe smp 4
#$ -l h_vmem=5G
#$ -l h_rt=24:0:0

module load anaconda2
source activate blobtools

cd /data/scratch/btx604/oasisia/blobtools/nt_v5

blobtools create \
 -i /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/genomePolishing/illumina/oasisia_pilon_step2.fasta \
 -b /data/scratch/btx604/oasisia/genomePolishing/illumina/blobtools/oasisia_sorted_step1.bam \
 -t /data/scratch/btx604/oasisia/blobtools/nt_v5/blast.out \
 -t /data/scratch/btx604/oasisia/blobtools/diamond.out \
 -o /data/scratch/btx604/oasisia/blobtools/nt_v5/oasisia_blobplot
```

Then we need to generate a tabular output containing the taxonomic rank

informations. This table contains information about the sequences in the order they appear in the assembly file and it will be used in 3.3. The following command is also generating two plots that can help to better understand your assembly.
blobtools1_view_plot_oasisia_v2.sh

```
#!/bin/bash
#$ -wd /data/scratch/btx604/oasisia/blobtools/nt_v5
#$ -o /data/scratch/btx604/oasisia/blobtools/nt_v5
#$ -j y
#$ -pe smp 4
#$ -l h_vmem=5G
#$ -l h_rt=24:0:0

module load anaconda2
source activate blobtools

cd /data/scratch/btx604/oasisia/blobtools/nt_v5

blobtools view \
 -i /data/scratch/btx604/oasisia/blobtools/nt_v5/oasisia_blobplot.blobDB.json \
 -o /data/scratch/btx604/oasisia/blobtools/nt_v5/

blobtools plot \
 -i /data/scratch/btx604/oasisia/blobtools/nt_v5/oasisia_blobplot.blobDB.json \
 -o /data/scratch/btx604/oasisia/blobtools/nt_v5/
```

Step 3.3: Filter the assembly

Now we need to extract fasta sequences from the polished assembly based on taxonomy. In order to accomplish this, 3 steps will be required:

1-Divide blobtools table

This will generate five tables containing only the sequences from a particular group of organisms

PAY ATTENTION! the last line of oasisia_blobplot.blobDB.bestsum.table.txt is not well formatted, in fact wc -l is giving one line more, the best is to delete the last empty line. Furthermore, the first lines of host table containing comments and info can be removed.

To get all the phyla present in my assembly do this:

```
awk -F '\t' '{print $6}' oasisia_blobplot.blobDB.bestsum.table.t
xt | sort | uniq
```

And then grep phyla by phyla with the following:
new_filter_blobtools_v1.sh

```
#!/bin/bash
#$ -wd /data/scratch/btx604/
#$ -o /data/scratch/btx604/
#$ -j y
#$ -pe smp 2
#$ -l h_vmem=5G
#$ -l h_rt=24:0:0

#variables:
blobplot_original=/data/SBCS-MartinDuranLab/03-Giacomo/data/$1/b
lobtools/"$1"_blobplot.blobDB.bestsum.table.txt
blobplot=/data/SBCS-MartinDuranLab/03-Giacomo/data/$1/blobtools
/"$1"_blobplot_tail_13.blobDB.bestsum.table.txt
host_table=/data/SBCS-MartinDuranLab/03-Giacomo/data/$1/blobtool
s/new_filter/new_filter_"$1"_blobplot_host.table.txt
proteobacteria_table=/data/SBCS-MartinDuranLab/03-Giacomo/data
/$1/blobtools/new_filter/new_filter_"$1"_blobplot_proteobacteri
a.table.txt
actinobacteria_table=/data/SBCS-MartinDuranLab/03-Giacomo/data
/$1/blobtools/new_filter/new_filter_"$1"_blobplot_actinobacteri
```

```
a.table.txt
other_bacteria_table=/data/SBCS-MartinDuranLab/03-Giacomo/data/$1/blobtools/new_filter/new_filter_"$1"_blobplot_other_bacteria.table.txt
virus_table=/data/SBCS-MartinDuranLab/03-Giacomo/data/$1/blobtools/new_filter/new_filter_"$1"_blobplot_virus.table.txt

echo 'working on '$1

cd /data/SBCS-MartinDuranLab/03-Giacomo/data/$1/blobtools
mkdir new_filter

tail -n +13 $blobplot_original > $blobplot

grep -wi Arthropoda $blobplot >> $host_table
grep -wi Annelida $blobplot >> $host_table
grep -wi Ascomycota $blobplot >> $host_table
grep -wi Brachiopoda $blobplot >> $host_table
grep -wi Chordata $blobplot >> $host_table
grep -wi Cnidaria $blobplot >> $host_table
grep -wi Echinodermata $blobplot >> $host_table
grep -wi Hemichordata $blobplot >> $host_table
grep -wi Mollusca $blobplot >> $host_table
grep -wi Nematoda $blobplot >> $host_table
grep -wi Nemertea $blobplot >> $host_table
grep -wi Platyhelminthes $blobplot >> $host_table
grep -wi Porifera $blobplot >> $host_table
grep -wi Priapulida $blobplot >> $host_table
grep -wi Streptophyta $blobplot >> $host_table
grep -wi Chytridiomycota $blobplot >> $host_table
grep -wi Ciliophora $blobplot >> $host_table
```

```
grep -wi Euglenozoa $blobplot >> $host_table
grep -wi Apicomplexa $blobplot >> $host_table
grep -wi Basidiomycota $blobplot >> $host_table
grep -wi Chlorophyta $blobplot >> $host_table
grep -wi Cryptophyta $blobplot >> $host_table
grep -wi Eukaryota-undef $blobplot >> $host_table
grep -wi Evosea $blobplot >> $host_table
grep -wi Haptista $blobplot >> $host_table
grep -wi Mucoromycota $blobplot >> $host_table
grep -wi Placozoa $blobplot >> $host_table
grep -wi Rotifera $blobplot >> $host_table

grep -wi unresolved $blobplot >> $host_table
grep -wi phylum.t.6%s $blobplot >> $host_table
grep -wi no-hit $blobplot >> $host_table

grep -wi Proteobacteria $blobplot >> $proteobacteria_table

grep -wi Actinobacteria $blobplot >> $actinobacteria_table


grep -wi Bacteria-undef $blobplot >> $other_bacteria_table
grep -wi Bacteroidetes $blobplot >> $other_bacteria_table
grep -wi Calditrichaeota $blobplot >> $other_bacteria_table
grep -wi Chlorobi $blobplot >> $other_bacteria_table
grep -wi Cyanobacteria $blobplot >> $other_bacteria_table
grep -wi Firmicutes $blobplot >> $other_bacteria_table
grep -wi Fusobacteria $blobplot >> $other_bacteria_table
grep -wi Ignavibacteriae $blobplot >> $other_bacteria_table
grep -wi Kiritimatiellaeota $blobplot >> $other_bacteria_table
```

```
grep -wi Lentisphaerae $blobplot >> $other_bacteria_table
grep -wi Nitrospirae $blobplot >> $other_bacteria_table
grep -wi Planctomycetes $blobplot >> $other_bacteria_table
grep -wi Verrucomicrobia $blobplot >> $other_bacteria_table
grep -wi Chlamydiae $blobplot >> $other_bacteria_table
grep -wi Nitrospinae $blobplot >> $other_bacteria_table
grep -wi Spirochaetes $blobplot >> $other_bacteria_table
grep -wi Thermodesulfobacteria $blobplot >> $other_bacteria_tabl
e

grep -wi Viruses-undef $blobplot >> $virus_table
```

```
qsub new_filter_blobtools_v1.sh oasisia
qsub new_filter_blobtools_v1.sh riftia
qsub new_filter_blobtools_v1.sh osedax
```

Quick check:

```
wc -l $blobplot #this is the total
wc -l $host_table #sum
wc -l $proteobacteria_table #sum
wc -l $actinobacteria_table #sum
wc -l $other_bacteria_table #sum
wc -l $virus_table #sum
```

2-Obtain headers

The next script will create a list of contig names (one per line) that will then be used to extract fasta sequences. It requires 3 inputs from the command line: $1 is the name to one of the tables we obtained before, $2 is the taxa of the organisms contained in the table (e.g. Actinobacteria, host...) and $3 is the path of the directory containing the input file and where you want to place the output.

create_headers_list_v1.sh

```
#!/bin/bash
```

```
#$ -wd /data/scratch/btx604/
#$ -o /data/scratch/btx604/
#$ -j y
#$ -pe smp 1
#$ -l h_vmem=1G
#$ -l h_rt=1:0:0

#This script is working on subsets depending on taxa of blobtools tables (e.g. oasisia_blobplot_host.table.txt)

#variables
input=$1
taxa=$2
working_directory=$3
output=headers_"$2".txt

cd $working_directory
touch $output

while IFS= read -r line
do
 echo $line | awk '{print $1;}' >> $output
done < $input
```

```
qsub create_headers_list_v1.sh oasisia_blobplot_Proteobacteria.table.txt proteobacteria /data/scratch/btx604/oasisia/blobtools/nt_v5/
```

- This example is showing how to launch the script from the command line

Now you have four/five files containing the headers divided by taxa

3-Obtain sequences

Let's create a directory "fasta" where we want to collect our fasta files, one for each taxa of interest.

```
mkdir fasta
```

Samtools as a function (faidx) that can extract subsequence from indexed reference sequence when they are matching with a specific pattern, or as in this case with the headers.

samtools_faidx_oasisia_blob_fastas_v1.sh

```
#!/bin/bash
#$ -wd /data/scratch/btx604/oasisia/blobtools/nt_v5/fasta/
#$ -o /data/scratch/btx604/oasisia/blobtools/nt_v5/fasta/
#$ -j y
#$ -pe smp 1
#$ -l h_vmem=5G
#$ -l h_rt=3:0:0

module load samtools/1.9

samtools faidx /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/genomePolishing/illumina/oasisia_pilon_step2.fasta -r /data/scratch/btx604/oasisia/blobtools/nt_v5/headers_proteobacteria.txt > /data/scratch/btx604/oasisia/blobtools/nt_v5/fasta/oasisia_proteobacteria.fasta

samtools faidx /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/genomePolishing/illumina/oasisia_pilon_step2.fasta -r /data/scratch/btx604/oasisia/blobtools/nt_v5/headers_actinobacteria.txt > /data/scratch/btx604/oasisia/blobtools/nt_v5/fasta/oasisia_actinobacteria.fasta

samtools faidx /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/genomePolishing/illumina/oasisia_pilon_step2.fasta -r /data/scratch/btx604/oasisia/blobtools/nt_v5/headers_bacteria.txt > /data/scratch/btx604/oasisia/blobtools/nt_v5/fasta/oasisia_bacteria.f
```

```
asta


samtools faidx /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/genomePolishing/illumina/oasisia_pilon_step2.fasta -r /data/scratch/btx604/oasisia/blobtools/nt_v5/headers_host.txt > /data/scratch/btx604/oasisia/blobtools/nt_v5/fasta/oasisia_host.fasta
```

- it works very quickly

Check if the files look alright:

```
wc -l /data/scratch/btx604/oasisia/blobtools/nt_v5/headers_proteobacteria.txt #5
grep -c "^>" /data/scratch/btx604/oasisia/blobtools/nt_v5/fasta/oasisia_proteobacteria.fasta #5

wc -l /data/scratch/btx604/oasisia/blobtools/nt_v5/headers_bacteria.txt
grep -c "^>" /data/scratch/btx604/oasisia/blobtools/nt_v5/fasta/oasisia_bacteria.fasta

wc -l /data/scratch/btx604/oasisia/blobtools/nt_v5/headers_actinobacteria.txt
grep -c "^>" /data/scratch/btx604/oasisia/blobtools/nt_v5/fasta/oasisia_actinobacteria.fasta

wc -l /data/scratch/btx604/oasisia/blobtools/nt_v5/headers_host.txt
grep -c "^>" /data/scratch/btx604/oasisia/blobtools/nt_v5/fasta/oasisia_host.fasta
```

- they should give the same value

Step 4: Haploidization

This is the final step of the assembly. The fasta obtained during the previous step

(oasisia_host.fasta) has high level of duplication level, in fact that version of the genome can be considered the diploid version. However, we are interested in the haploid version that will be easier to annotate and will give better and more precise results. **Purge_Dups** is a software designed for this purpouse. Minimap2, an aligner tool, Busco and Quast will also be used during this phase.

Step 4.1: Installation

Purge_Dups installation:

```
03g
cd src/
git clone https://github.com/dfguan/purge_dups.git
cd purge_dups/src && make
git clone https://github.com/dfguan/runner.git
module load python
cd runner && python setup.py install --user
```

Minimap2 installation:

```
module load anaconda2
conda create -n Minimap2
source activate Minimap2
conda install -c bioconda minimap2
```

Step 4.2: Purge_Dups and quality checks

All the programs that we need (Busco and Quast as well) are inluded in the following unique and universal script. It just needs a directory "purge_dups" containing the output of blobtools (e.g. oasisia_host.fasta) and it should be submitted like this:

```
qsub purge_dups_v2.sh $1 $2
```

- $1 is the species name (e.g. oasisia)
- $2 is the absolute path to the PacBio raw reads obtained transforming subreads.bam into fq.gz with BAM2fastx tools 19/08/19 (e.g. /data/SBCS-MartinDuranLab/03-Giacomo/data/oasisia/00-pacbio/oasisia_pb_raw.fastq.gz)

Code to run the pipeline:

```
#!/bin/bash
#$ -wd /data/scratch/btx604/
#$ -j y
#$ -o /data/scratch/btx604/
#$ -pe smp 12
#$ -l h_vmem=30G
#$ -l h_rt=240:0:0
#$ -l highmem

#variables
ref_genome=/data/scratch/btx604/$1/purge_dups/"$1"_host.fasta
pb_fasta=$2
ref_genome_split=/data/scratch/btx604/$1/purge_dups/"$1"_host.split.fasta
self_aln_genome=/data/scratch/btx604/$1/purge_dups/"$1"_host.split.self.paf.gz



cd /data/scratch/btx604/$1/purge_dups/

echo 'working on '$1

echo 'MINIMAP2 _____'if [ -e ./purge_"$1".paf.gz ]
then
  echo "purge_$1.paf.gz found"
else
  module load anaconda2
  source activate Minimap2
```

```
27
28    minimap2 -x map-pb $ref_genome $pb_fasta | gzip -c - > purge
      _$1.paf.gz
29
30    source deactivate
31    module unload anaconda2
32  fi
33
34  echo 'PBCSTAT ___________________________________________________
      ____________________'
35  if [ -e ./PB.base.cov ]
36  then
37    echo "PB.base.cov found"
38  else
39    module load python
40
41    /data/SBCS-MartinDuranLab/03-Giacomo/src/purge_dups/bin/pbcsta
      t *.paf.gz #(produces PB.base.cov and PB.stat files)
42
43    module unload python
44  fi
45
46  echo 'CALCUTS ___________________________________________________
      ____________________'if [ -e ./calcults.log ]
47  then
48    echo "calcults.log found"
49  else
50    module load python
51
52    /data/SBCS-MartinDuranLab/03-Giacomo/src/purge_dups/bin/calcut
      s PB.stat > cutoffs 2> calcults.log
```

```
53
54    module unload python
55  fi
56
57  echo 'SPLIT_FA ____________________________________________________
    ____________________'
58  if [ -e "$ref_genome_split" ]
59  then
60    echo "$ref_genome_split found"
61  else
62    module load python
63
64    /data/SBCS-MartinDuranLab/03-Giacomo/src/purge_dups/bin/split_
    fa $ref_genome > $ref_genome_split
65
66    module unload python
67  fi
68
69  echo 'MINIMAP2 ____________________________________________________
    ____________________'if [ -e "$self_aln_genome" ]
70  then
71    echo "$self_aln_genome found"
72  else
73    module load anaconda2
74    source activate Minimap2
75
76    minimap2 -x asm5 -DP $ref_genome_split $ref_genome_split | gzi
    p -c - > $self_aln_genome
77
78    source deactivate
79    module unload anaconda2
```

```
fi

echo 'PURGE_DUPS ____________________________________________________
_____________________'
if [ -e ./purge_dups.log ]
then
  echo "purge_dups.log found"
else
  module load python

  /data/SBCS-MartinDuranLab/03-Giacomo/src/purge_dups/bin/purge_
dups -2 -T cutoffs -c PB.base.cov $self_aln_genome > dups.bed 2>
purge_dups.log

  module unload python
fi

echo 'GET_SEQS _____________________________________________________
_____________________'
if [ -e ./hap.fa ]
then
  echo "hap.fa found"
else
  module load python

  /data/SBCS-MartinDuranLab/03-Giacomo/src/purge_dups/bin/get_se
qs dups.bed $ref_genome > purged.fa 2> hap.fa

  module unload python
fi

```

```
106 echo 'BUSCO _____________________________________________________
    ____________________'
107 if [ -e ./busco/ ]
108 then
109   echo "hap.fa found"
110 else
111   module load busco/3.0
112   module load augustus
113   export AUGUSTUS_CONFIG_PATH=/data/SBCS-MartinDuranLab/02-Chema
    /src/Augustus/config
114
115   mkdir busco
116   cd ./busco/
117
118   BUSCO.py -i ../purged.fa -m genome -o busco -f -c 4 -l /data/h
    ome/btx604/datasets/metazoa_odb9
119
120   cd ..
121
122   module unload busco/3.0
123   module unload augustus
124 fi
125
126 echo 'QUAST _____________________________________________________
    ____________________'
127 if [ -e ./quast/ ]
128 then
129   echo "quast found"
130 else
131   module load anaconda2
132   source activate quast
```

```
133
134   mkdir quast
135   quast \
136    ./purged.fa \
137    -o ./quast \
138    --eukaryote \
139
140   source deactivate
141   module unload anaconda2
142 fi
```

- purged.fa is the haploid version of the genome