Simple Plamsid Pipeline

Run plasmid sequencing, assembly, and alignment as simple as it is.

Download and Setup

This pipeline requires a 64-bit Linux system and python (supported versions are python3: 3.2 and higher).

Download the pipeline via Git:

```
git clone simple_plasmid
```

Several dependencies such as `minimap2` and `samtools` are required. Check `utils/configs.py` for further requirements and modify them accordingly. `alignment.sh` can be found under directory `utils`.

Preparation

Before running actual time-consuming `exec` parts, the pipeline provides a `prep` mode to prepare the input data before execution as follows.

```
$python ../simple_plasmid/simp_plas.py prep -h
usage: Simple Plasmid Pipeline prep [-h] -c CSV_FILE -r ROOT_DIR --caller
{dorado,guppy}

options:
  -h, --help            show this help message and exit
  -c CSV_FILE, --csv_file CSV_FILE
                        SUMMARY CSV-format file
  -r ROOT_DIR, --root_dir ROOT_DIR
                        path to the root directory, consists of
subdirectories 1) no_sample and 2) ReferenceMaps.
  --caller {dorado,guppy}
                        base caller
```

The `CSV_FILE` must contain following columns as the minimum requirement (additional columns are permitted, **but make sure than comma `,` must be excluded from the common fields to avoid CSV file format violation**). Column headers can be case-insensitive.

- sample name
- size (bp)
- barcode
- experiment id
- flow cell id
- supplied map?
- path to map
- quality cutoff

- length cutoff

`experiment id` and `flow cell id` is optional for guppy basecalling step, but necessary for dorado basecalling step. If dorado basecalling is in use, `experiment id` must be filled with identical values across all rows.

The `ROOT_DIR` must contain at least two subdirectories, `no_sample/` and `ReferenceMaps/`. `no_sample/` contains all the un-modified samples before base calling. `ReferenceMaps/` contains all the reference map FASTA file if supplied.

An example for `CSV_FILE` is given as follows.

```
Sample name,Experiment ID,Flow Cell ID,Sample type,Size
(bp),Barcode,Analysis to run,Supplied map?,Path to map,Quality
cutoff,Length cutoff
GB1_Tox5,exp0,FLO-MIN114,Plasmid,6043,29,basecall and assemble,Y,GB1
Tox5.fasta,11,0
pTWIST_acceptor,exp0,FLO-MIN114,Plasmid,unknown,30,basecall and
assemble,Y,pTWIST acceptor.fasta,11,0
```

An example for `ROOT_DIR` hierarchy is given as follows.

```
ROOT_DIR/
  |-no_sample/
    |-YYYYMMDD_xxx1/
      |-fast5/*.fast5 (or pod5/*.pod5)
      |-*
    |-*
  |-ReferenceMaps/
    |-GB1 Tox5.fasta
    |-pTWIST acceptor.fasta
    |-*
  |-*
```

If field `Quality cutoff` is left as blank, the default cutoff would be 11. If field `Length cutoff` is left as blank, the default cutoff would be 0.

Field `Supplied map?` can be filled as either **Y** or **N**. if **Y** is filled, field `Path to map` must be filled by the relative path to the corresponding reference map FASTA file, which can be accessed via absolute path `<ROOT_DIR>/ReferenceMaps/<Path to map>`.

To obey the IGV naming convention, all the names including `<Path to map>`, the header record of `<Path to map>`, and the field `sample name` must be consistent, where only digits (0-9), alphabets (a-zA-Z), and dash `-` are permitted. Any forbiddened character be detected will be automatically corrected and replaced by `-`. The naming consistency will be ensured by the pipeline as well. i.e., simply attach anything you have, the pipeline will correct it and fix it.

After preparation, `ROOT_DIR/plas_config.csv` will be generated, do not modify it!

Execution

The `exec` mode has following usages and options.

```
$python ../simple_plasmid/simp_plas.py exec -h
usage: Simple Plasmid Pipeline exec [-h] [-f] [-a APX_RATIO] -r ROOT_DIR -
-caller {dorado,guppy}

optional arguments:
  -h, --help            show this help message and exit
  -f, --filt_first      if `-f` be set, filterred reads will be used for
assembly, (default: False)
  -a APX_RATIO, --apx_ratio APX_RATIO
                        rerun assembly (if size is given) with `-
approx_size` option when ratio between alen and size > apx_ratio.
                        (default: 1.5)
  -r ROOT_DIR, --root_dir ROOT_DIR
                        path to the root directory after `prep` step.
  --caller {dorado,guppy}
                        base caller
```

Suppose your are current in the working root directory, simply typing the following command to start the execution pipeline.

```
python /path/to/simple_plasmid/simp_plas.py exec -r .
```

First, the pipeline will run the base calling against all FAST5 (or Pod5) file found in `ROOT_DIR/calledFast5` (or `ROOT_DIR/calledPod5`) and output in `ROOT_DIR/calledFastq`. If all FASTQ file required by the barcodes from `ROOT_DIR/plas_config.csv` exist under directory `ROOT_DIR/calledFastq` before running the program, base calling step will be skipped. For each barcode `barcodeXX`, `ROOT_DIR/calledFastq/barcodeXX/` stores all the base-called FASTQ files.

Second, for each record listed in `ROOT_DIR/plas_config.csv` with barcode `barcodeXX`, the pipeline will perform read filtering via NanoFilt under certain quality and read length threshold cutoff specified, and temporarily output in `ROOT_DIR/calledFastq/barcodeXX_filt/`.

Third, if flag `-f` (or `--filt_first`) is set, filtered reads will be used to perform assembly via `epi2me-labs/wf-clone-validation`, raw reads otherwise. Each data record will be assembled by Flye once. If the Flye assembly failed, Canu assembly will be performed.

Otherwise, if the Flye assembly leads to inconsistent plasmid size compared to approximated plasmid size (if known), it will run another Flye assembly with additional assembly option `--approx_size <approx_size>`. For example, suppose the approximated size is 2000bp, and `-a 1.5` is provided to the pipeline, the former step will be performed if the assembly length is greater than 3000bp.

Utimately, either a successful assembly will be marked as final or no assembly generated.

If the reference map FASTA file is provided, Minimap2 alignment with samtools indexing will also be performed between the reference and filtered FASTQ file.

The pipeline will iterate through all the data records. When errors are produced through processing a data record, the pipeline will move to next data record immediately instead of executing subsequent commands for current failed record.

Outputs

All the logs can be found under directory: `ROOT_DIR/logs`

All the base-called fastq can be found under directory: `ROOT_DIR/calledFastq`

All the assemblies can be found under directory: `ROOT_DIR/asmOutput`

All the alignments can be found under directory: `ROOT_DIR/alnOutput`

Contacts

Feel free to contact john.luo@anu.edu.au if any bugs be experienced during execution.