PoreCamp2016 : Understanding your MinION data

0 Overview

By the end of this tutorial, you will understand when data becomes available during a MinION run, where to find the data on your laptop and/or server, what is contained in the FAST5 output files, and how to generate basic metrics on the quantity and quality of your experimental data.

Much of this tutorial is based on material produced by Mick Watson for PoreCamp2015 (`https://github.com/mw55309/PoreCamp/blob/master/Por eCamp.md`).

1 When is data produced during an experiment?

1.1 MinKNOW produces pre-basecalled FAST5 files

During a typical MinION experiment run with a single unbarcoded sample, the MinKNOW software running on the controlling laptop is used to start the sequencing script. As the experiment progress, one FAST5 file per DNA molecule in a directory called `/PATH/TO/RUNFOLDER/reads/` which contains the events (aggregated signal measurements) for the DNA molecule. These “pre-basecalled” FAST5 files are saved on the SSD of the laptop running MinKNOW.

1.2 Metrichor performs basecalling in the ONT cloud

Basecalling of the initial FAST5 files involves starting the Metrichor agent on the sequencing laptop which then transfers a copy of each pre-basecalled FAST5 file from the `/PATH/TO/RUNFOLDER/reads/` directory the ONT cloud-baed basecalling service, and moves the original FAST5 file to the `/PATH/TO/RUNFOLDER/reads/uploads` directory. The Metrichor service then returns an augmented FAST5 file containing the original information and basecalls in FASTQ format to another FAST5 file in either the `/PATH/TO/RUNFOLDER/reads/downloads/pass` or `/PATH/TO/RUNFOLDER/reads/downloads/fail` folder. The pass/fail classification, introduced in November 2014, is intended to help the user distinguish higher and lower quality read data. The classification criteria may change between versions of Metrichor, but for a 2D library, a read will “pass” if basecalling was successful, and a 2D read was produced with a mean base quality score greater than a pre-defined threshold.

1.3 Barcoding adds an extract sub-directory to the output

When sequencing a library containing multiple barcoded samples, the output from Metrichor is deposited in directories called `/PATH/TO/RUNFOLDER/reads/downloads/pass/BCNN/` where BCNN would be the barcode identifier BC01, BC02, BC03, etc. The fail FAST5 files are not split by barcode and all appear in a single directory `/PATH/TO/RUNFOLDER/reads/downloads/fail/`

1.4 The WIMP workflow for species ID

After running the usual basecalling workflow, you have the option of using Metrichor to run the "What's in my pot?" (WIMP) workflow to identify the taxonomy of the reads and the proportion of reads for each taxonomic class.

2 The ONT FAST5 data formats

To answer a biological question, all you need from the ONT data files are the basecalls in FASTQ format. If you are short of time, you are welcome to skip to section 3 now.

If you would like to understand the internal format of the FAST5 files and know what sequencing and basecalling metadata is stored there, read on.

The MinKNOW and Metrichor agents produce one file per DNA molecule. The sequencing data is stored in an HDF5 file, a "container" for storing data of a variety of types (e.g., integers, floats, strings, arrays) in a single file. HDF5 files have a hierarchical structure containing "groups" which are a bit like directories, and "datasets" which are multidimensional arrays of data elements of type integers, float, strings, etc.

For your experiments this week, you used the "**MinION Mkl**" device with "R9" flow cells with:

Nanopore kit or software	**R9 version***
Nanopore Sequencing Kit	SQK-NSK007
Native Barcoding Kit	EXP-NBD002
PCR Barcoding Kit I	EXP-PBC001 or 096
Low Input Expansion Pack	EXP-LWI001
MinION Flow Cell	FLO-MIN104 (Mk1 or 1B)
MinKNOW	0.51.3.55 NC_....._FLO_MIN104
Metrichor	Workflows ending SQK-NSK007**: - Lambda Control Experiment RNN for SQK-NSK007 (Revision 1.107) - 1D Basecalling RNN for SQK-NSK007 (Revision 1.107) - 2D Basecalling RNN for SQK-NSK007 (Revision 1.107) - 2D Basecalling RNN for SQK-NSK007 plus Human Exome (Rev 1.97) - Barcoding plus 2D Basecalling RNN for SQK-NSK007 (Revision 1.107) - WIMP plus 2D Basecalling RNN for SQK-NSK007 (Revision 1.107) Workflows ending in SQK-RAD001**: - Lambda Control Experiment RNN for SQK-RAD001 - 1D Basecalling RNN for SQK-RAD001 (Revision 1.107) - 2D Basecalling RNN for SQK-RAD001 (Revision 1.107) - WIMP plus Basecalling RNN for SQK-RAD001 (Revision 1.107)

* https://community.nanoporetech.com last updated 25-Jul-2016, accessed 13-Aug-2016.
** https://metrichor.com/workflow, accessed 13-Aug-2016.

The data stored in the FAST5 files depends on the software and version of the MinKNOW and Metrichor software in use when you did your experiment.

The main differences are that prior to mid-November 2015, the HMM basecaller (described as "Basecall_2D_000" or "ONT sequencing workflow" inside the FAST5 metadata) store the template and complement were stored under the "Basecall_2D" workflow:

```
/Analyses/Basecall_2D_000/BaseCalled_template/
/Analyses/Basecall_2D_000/BaseCalled_complement/
/Analyses/Basecall_2D_000/BaseCalled_2D/
```

They are now stored in separate "Basecall_1D" and "Basecall_2D" groups:

```
/Analyses/Basecall_1D_000/BaseCalled_template/
/Analyses/Basecall_1D_000/BaseCalled_complemen t/
/Analyses/Basecall_2D_000/BaseCalled_2D/
```

The newer "ONT nanonet basecaller", ONT's first RNN-based local (i.e., non-cloud) base caller released around the time that the R9 flow cells first became available ~June 2016, use additional groups:

```
/Analyses/Basecall_1D_000/BaseCalled_template/
/Analyses/Basecall_RNN_1D_000/BaseCalled_templ ate/
```

3 Software for opening FAST5 files

3.1 Extracting data with HDF5 command-line tools

List the top-level groups inside a FAST5 file:

```
$ h5ls FAST5FILE
Analyses                 Group
Sequences                Group
UniqueGlobalKey          Group
```

Recursively list all the groups in the file:

```
$ h5ls -r FAST5FILE | less

/                        Group
/Analyses                Group
/Analyses/Basecall_1D_000 Group
/Analyses/Basecall_1D_000/BaseCalled_complement Group
/Analyses/Basecall_1D_000/BaseCalled_complement/Events Dataset {678}
/Analyses/Basecall_1D_000/BaseCalled_complement/Fastq Dataset {SCALAR}
/Analyses/Basecall_1D_000/BaseCalled_complement/Model Dataset {4096}
/Analyses/Basecall_1D_000/BaseCalled_template Group
/Analyses/Basecall_1D_000/BaseCalled_template/Events Dataset {657}
...
```

If you use “h5ls -r” on a pre- and post-basecalled FAST5 file, you can see which groups were added during basecalling.

Display the entire contents of the FAST5 file:

```
$ h5dump FAST5FILE | less
GROUP "/" {
   ATTRIBUTE "file_version" {
      DATATYPE  H5T_IEEE_F64LE
      DATASPACE  SCALAR
      DATA {
      (0): 1
      }
   }
   GROUP "Analyses" {
      GROUP "Basecall_1D_000" {
         ATTRIBUTE "chimaera version" {
...
```

To generate a help message that lists all options available:

```
h5ls -h | less
h5dump -h | less
```

To open a GUI to browse the contents of a FAST5 file type the following, then left- or right-click on on groups and datasets to display them:

```
hdfview FAST5FILE
```

3.2 poretools

Poretools is a toolkit for extracting data from fast5 data, written by Nick Loman and Aaron Quinlan (http://bioinformatics.oxfordjournals.org/content/early/2014/08/19/bioinformatics.btu555.abstract).

Typing the command with no arguments gives a basic usage message:

```
$  poretools
usage: poretools [-h] [-v]
{combine,fastq,fasta,stats,hist,events,readstats,tabular,nucdist,quald
ist,winner,squiggle,times,yield_plot,occupancy}
                 ...
poretools: error: too few arguments
```

The `-v` option displays the program version:

```
$ poretools -v
poretools 0.5.1
```

And the -h option gives a full help message:

```
$ poretools -h
```

```
usage: poretools [-h] [-v]
{combine,fastq,fasta,stats,hist,events,readstats,tabular,nucdist,qualdist,winner,
squiggle,times,yield_plot,occupancy}
                 ...

optional arguments:
  -h, --help            show this help message and exit
  -v, --version         Installed poretools version

[sub-commands]:

{combine,fastq,fasta,stats,hist,events,readstats,tabular,nucdist,qualdist,winner,
squiggle,times,yield_plot,occupancy}
    combine             Combine a set of FAST5 files in a TAR achive
    fastq               Extract FASTQ sequences from a set of FAST5 files
    fasta               Extract FASTA sequences from a set of FAST5 files
    stats               Get read size stats for a set of FAST5 files
    hist                Plot read size histogram for a set of FAST5 files
    events              Extract each nanopore event for each read.
    readstats           Extract signal information for each read over time.
    tabular             Extract the lengths and name/seq/quals from a set of
                        FAST5 files in TAB delimited format
    nucdist             Get the nucl. composition of a set of FAST5 files
    qualdist            Get the qual score composition of a set of FAST5 files
    winner              Get the longest read from a set of FAST5 files
    squiggle            Plot the observed signals for FAST5 reads.
    times               Return the start times from a set of FAST5 files in
                        tabular format
    yield_plot          Plot the yield over time for a set of FAST5 files
    occupancy           Inspect pore activity over time for a set of FAST5
                        files
```

Why don't you try out each of the poretools sub-commands?

3.3 poRe

poRe is a library for R written by Mick Watson and colleagues, available from SourceForge (https://sourceforge.net/projects/rpore/) and published in bioinformatics (http://bioinformatics.oxfordjournals.org/content/31/1/114). poRe is simple to install, requiring only R 3.0 or above and a few additional libraries.

The poRe library is set up to read the newer FAST5 group names by default (see section 2), but offers users parameters to open older data as well.

To use poRe, you need to start R at the command-line, or start the RStudio graphical interface to R, load the poRe library, they type in R statements that open, manipulate and close the FAST5 file(s).

You could also type all the commands into a file called `SOMETHING.R`, then execute the commands using `Rscript SOMETHING.R`

To view some example scripts, view the documentation and examples at

`https://github.com/mw55309/poRe_docs`

Of particular interest is the “Run QC” example which explains how to use the pore_rt() function to generate a tab-separated table of statistics with columns:

```
"filename","channel_num","read_num","read_start_time","status","tlen","clen","len
2d","run_id","read_id","barcode","exp_start"
```

3.4 Python h5py module

Another way to write your own scripts to interrogate FAST5 files is to use the h5py module in Python.

```
$ cat extractattr.py
#!/usr/bin/env python

import h5py, numpy as np, os, sys

if len(sys.argv) < 3:
    print('Usage:   extractattr.py runname fast5_path')
    print('         Extract a table of attributes and values from a fast5 file.')
    print('')
    sys.exit(1)
runname, fast5_path = sys.argv[1:]

def Process(fast5_path):
  # Collate the attribute list
    hdf = h5py.File(fast5_path, 'r')
  # Get the names of all groups and subgroups in the file
    list_of_names = []
    hdf.visit(list_of_names.append)
    attribute = []
    for name in list_of_names:
      # Get all the attribute name and value pairs
        itemL = hdf[name].attrs.items()
        for item in itemL:
            attr, val = item
            if type(hdf[name].attrs[attr]) == np.ndarray:
                val = ''.join(hdf[name].attrs[attr])
            val = str(val).replace('\n', '')
            attribute.append([runname, name+'/'+attr, val])
    hdf.close()
  # Print the header
    print('{0}'.format('\t'.join(['runname', 'attribute', 'value'])))
  # Print the attribute list
    print('{0}'.format('\n'.join(
        ['\t'.join([str(x) for x in item]) for item in attribute])))
```

```
if __name__ == '__main__':
    Process(fast5_path)
$
$./extractattr.py PC2016-EXPT1 FAST5FILE
runname attribute       value
PC2016-EXPT1    Analyses/Basecall_1D_000/time_stamp     2016-Jun-03 22:00:16
PC2016-EXPT1    Analyses/Basecall_1D_000/dragonet version       1.22.2
PC2016-EXPT1    Analyses/Basecall_1D_000/name   ONT Sequencing Workflow
PC2016-EXPT1    Analyses/Basecall_1D_000/chimaera version       1.22.6
...
```

6 More information

- Nanopore MAP Community Forum : `https://community.nanoporetech.com`
- HDF5 containers : `https://www.hdfgroup.org/HDF5/Tutor/introductory.html`
- FAST5 format : `http://f1000research.com/articles/4-1075/v1` (Table S2)
- Glossary of terms : `http://f1000research.com/articles/4-1075/v1` (File S2)
- poretools : `https://github.com/arq5x/poretools`
- poRe : `https://github.com/mw55309/poRe_docs`
- h5py : `http://www.h5py.org`