modPhred

Release 1.0b

Leszek Pryszcz

Apr 12, 2024

FIRST STEPS

1 What is modPhred? 1

2 What do I need to run ModPhred? 3

3 Why use modPhred? 5

4 Getting help 7

5 What does modPhred stand for? 9

6 Citation 11

6.1 Installation . . . 11
6.1.1 Manual installation . . . 11
6.1.2 Docker image . . . 12
6.2 Test datasets . . . 13
6.2.1 ModPhred pipeline with live basecalling . . . 13
6.2.2 ModPhred pipeline with basecalled Fast5 files . . . 14
6.2.3 Compare modPhred and megalodon results . . . 15
6.2.4 Test data generation . . . 16
6.3 Running the pipeline . . . 16
6.3.1 Local on-the-fly basecalling . . . 16
6.3.2 Remote on-the-fly basecalling . . . 17
6.3.3 Without basecalling . . . 17
6.3.4 Using custom-modifications models . . . 18
6.3.5 Processing (very) large datasets . . . 18
6.4 Program output . . . 18
6.4.1 Data formats . . . 19
6.4.2 Why base Y is detected as modified, while model only reports modifications for X? . . . 21
6.5 Visualisation . . . 21
6.5.1 QC plots . . . 22
6.5.2 Modification frequency in certain regions . . . 23
6.5.3 Visualisation in genome browsers (IGV) . . . 23
6.6 Correlation between positions . . . 24
6.7 Read clustering . . . 25
6.7.1 Clustering details . . . 25
6.8 Scripts . . . 26
6.8.1 mods_from_bams . . . 26
6.9 Basecalling . . . 26
6.9.1 Modification-aware models . . . 26
6.10 Encoding of modifications . . . 26
6.11 Alignments . . . 27

6.12 Detection of modifications 28

CHAPTER

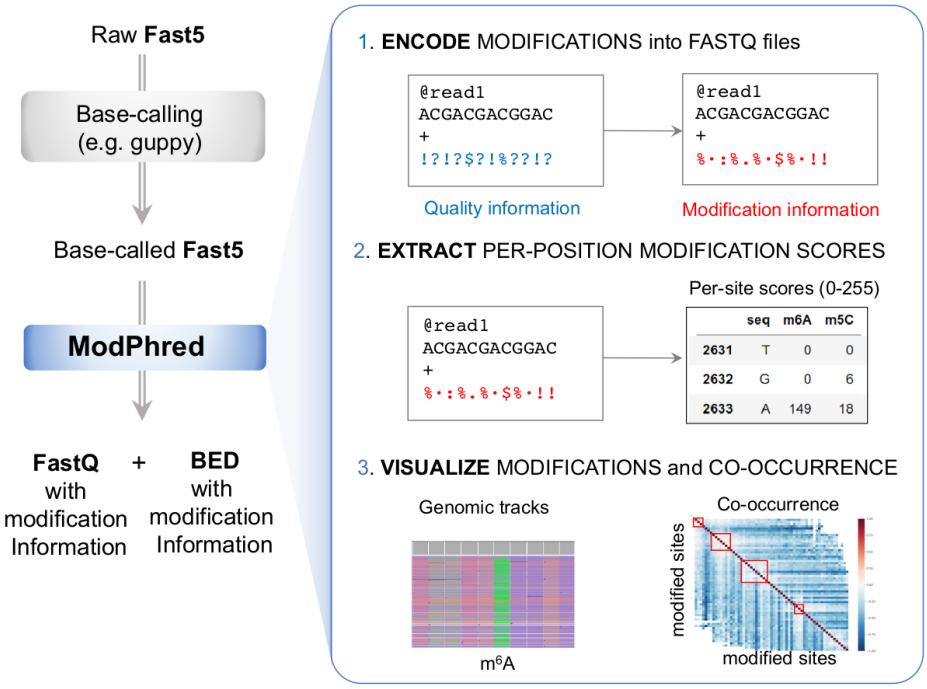
ONE

WHAT IS MODPHRED?

modPhred is a pipeline for detection, annotation and visualisation of DNA/RNA modifications. The pipeline consists of four steps / modules:

1. modEncode: encoding modification probabilities in FastQ (mod_encode.py)
2. modAlign: build alignments keepind modification information in BAMs (mod_align.py)
3. modReport: extraction of RNA modification information (bedGraph) and QC reports (mod_report.py)
4. modAnalysis:
    a. plotting QC stats, pairplots & venn diagrams (mod_plot.py),
    b. co-occurrence of modifications(mod_correlation.py)
    c. per-read clustering based on modification profiles (mod_cluster.py)

All these scripts can be run separately or as a pipeline by executing `modPhred/run`. You can find more the details in the Methods section.



CHAPTER

TWO

WHAT DO I NEED TO RUN MODPHRED?

To run modPhred, you will need:

0. *all dependencies installed*
1. reference sequence (FastA)
2. raw ONT data (Fast5)
3. modification-aware guppy_basecaller model.

Currently, there is only one modification-aware model distributed together with guppy. You can find more experimental models or you can train your own models using taiyaki.

More information about running the pipeline.

CHAPTER

THREE

WHY USE MODPHRED?

Cause why not! And seriously, it is:

- free (MIT licensed) & fast (a few times faster than other tools)
- easy-to-use & versatile: will do all for you with just one command (or at least encoding of modifications in FastQ, alignments, detection of modified positions, QC & plotting…)
- powerfull & space-optimised: it stores the modification status inside FastQ/BAM
  - no external files/DBs needed
  - you can visualise modification status of all bases of all reads in your favourite genome browser ie IGV
  - you can remove all basecalled Fast5 files to save disk space or skip separate basecalling step entirely thanks to on-the-fly basecalling!
- visually attractive: it produces nice plots (or at least not so bad so far… still working on it;))

CHAPTER

FOUR

GETTING HELP

If you have any questions, issues or doubts, first please get familiar with our documentation. This should address most common questions/problems. Then, have a look at issues users reported so far. If you don't find the solution, please open a new one.

CHAPTER

FIVE

WHAT DOES MODPHRED STAND FOR?

The tool stores base modification status (probability of base having various types of modifications) encoded inside FastQ/BAM file instead of base quality (also called Phred scores), thus **mod** (for modification) & **Phred** (for base quality).. Or something like that :P

Initially, this tool was called **Pszczyna**, pronounced ptna, but since almost no one could pronounce or memorise it, we came up with much easier, yet so much less sexy name… There are numerous more interesting words that Pszczyna in Polish. For more information, check Wikipedia.

CHAPTER

SIX

CITATION

If you find this work useful, please cite:

Pryszcz LP and Novoa EM (2022) ModPhred: an integrative toolkit for the analysis and storage of nanopore sequencing DNA and RNA modification data. Bioinformatics, 38:257-260.

Please, consider citing *dependencies* as well.

6.1 Installation

6.1.1 Manual installation

modPhred is written in Python3 and should work in most UNIX systems.

Make sure you install all programs listed below, before runnning the pipeline.

- first the repository

```
mkdir ~/src
cd src
git clone https://github.com/novoalab/modPhred
```

- from conda (use miniconda3!)

```
conda create -y -n modPhred python=3.7
conda activate modPhred
conda install minimap2 samtools hdf5 wget
```

- from pip

```
# create & activate new virtual environment
mkdir -p ~/src/venv
python3 -m venv ~/src/venv/modPhred
source ~/src/venv/modPhred/bin/activate

pip install h5py matplotlib pysam pandas seaborn sklearn mappy pebble numba
```

- guppy_basecaller has to be obtained from Nanopore Tech. Software page Alternatively, you can try this for GPU or this for CPU version. For GPU basecalling to work, you'll need to install CUDA with NVIDIA drivers. Check my blog for instructions for Ubuntu 18.04 or NVIDIA CUDA website for other systems.
- pyguppyclient (this will work with guppy v3.6.1)

```
pip install pyguppyclient==0.0.6
```

Once you have all dependencies installed, we recommend to try running it with *test dataset*. It'll be much easier to troubleshoot all potential issues this way.

Which pyguppyclient version should I install?

If you plan to perform live basecalling, you'll need to install right version of pyguppyclient. Guppy API was changing across version and unfortunately the newer versions are not back-compatible. Therefore, you'll need to install pyguppyclient version matching the version of guppy basecaller.

Guppy version	pyguppyclient version
>= 5.0	0.1.0
>= 4.4 & <5.0	0.0.9
>= 4.0 & <4.4	0.0.7a1
>= 3.4 & <4.0	0.0.6
< 3.4	not supported!

For example, if you intend to use guppy 4.0.15, you'll need to install pyguppyclient v0.0.7a1 as follows:

```
pip install pyguppyclient==0.0.7a1
```

Note, only one version of pyguppyclient can be installed in your system. If you wish to use more than one version, you can install them using virtual environments as follows:

```
python3 -m venv ~/src/venv/pyguppyclient006
source ~/src/venv/pyguppyclient006/bin/activate
pip install pyguppyclient==0.0.6 pysam pandas seaborn
```

And whenever you wish to switch to this version, just execute:

```
source ~/src/venv/pyguppyclient006/bin/activate
```

Once you are finish with computation eihert close the terminal window or execute `deactivate`.

6.1.2 Docker image

We maintain docker image for below versions of guppy:

- 3.6.1 (with pyguppyclient v0.0.6)
- 5.0.11 (with pyguppyclient v0.1.0) - you'll need CUDA v11.1 and most recent GPU drivers installed!

If you want to use it, make sure you have Docker, GPU drivers, CUDA and nvidia-docker installed. The easiest may be to follow nvidia-docker installation tutorial.

In order to execute *test example*, all you need to do is to adjust the version of guppy in the below command:

```
cd test
acc=PRJEB22772
docker run --gpus all -u $UID:$GID -v `pwd`:/data lpryszcz/modphred-3.6.1 \
  /opt/modPhred/run -f /data/ref/ECOLI.fa -o /data/modPhred/$acc \
  -i /data/$acc/{MARC_ZFscreens_R9.4_1D-Ecoli-run_FAF05145,MARC_ZFscreens_R9.4_2D-Ecoli-
```

(continues on next page)

(continued from previous page)

```
↪run_FAF05711} \
  -t4 --host /usr/bin/guppy_basecall_server
```

As you can see, the above command got a bit complicated. This is because:

- we need to enable GPU
- define user & group (otherwise all output files will be owned by root)
- bind local directory within container
- and define all input folders (because autocompletion doesn't work inside the container)

Note, dam-dcm-cpg model has been replaced in guppy v4.5 by the new 5mc model, so additionally we'll need to specify a model name in the recent versions of guppy.

```
docker run --gpus all -u $UID:$GID -v `pwd`:/data lpryszcz/modphred-5.0.11 \
  /opt/modPhred/run -f /data/ref/ECOLI.fa -o /data/modPhred5/$acc \
  -i /data/$acc/{MARC_ZFscreens_R9.4_1D-Ecoli-run_FAF05145,MARC_ZFscreens_R9.4_2D-Ecoli-
↪run_FAF05711} \
  -t4 --host /usr/bin/guppy_basecall_server -c dna_r9.4.1_450bps_modbases_5mc_hac.cfg
```

If you wish to use the original dam-dcm-cpg model with the latest versions of guppy, you can find a copy in the /data folder in modPhred repository.

6.2 Test datasets

Below you'll find detailed information on running modPhred pipeline on test dataset. You have two options:

- download raw Fast5 files and run modPhred with live basecalling (you'll need NVIDIA GPU with CUDA installed either locally or in a remote computer)
- download pre-basecalled Fast5 and run modPhred with pre-basecalled Fast5 files (no GPU needed)

6.2.1 ModPhred pipeline with live basecalling

Download test data

Get raw Fast5 data from PRJEB22772 (subset of reads from `NC_000913.3:1-100000`):

```
mkdir -p test
cd test
wget https://public-docs.crg.es/enovoa/public/lpryszcz/src/modPhred/test/ -q --show-
↪progress -r -c -nc -np -nH --cut-dirs=6 --reject="index.html*"
```

Run modPhred (local basecalling)

Running entire modPhred pipeline with live basecalling (~6 minutes using RTX2080Ti):

```
acc=PRJEB22772
~/src/modPhred/run -f ref/ECOLI.fa -o modPhred/$acc -i $acc/* -t4 \
    -c dna_r9.4.1_450bps_modbases_dam-dcm-cpg_hac.cfg \
    --host ~/src/ont-guppy_3.6.1/bin/guppy_basecall_server
```

Note, you can enable automatic removal of intermediate files (FastQ/M) using `--cleanup`.

Instead you can run all steps one-by-one as follow:

```
~/src/modPhred/src/guppy_encode_live.py -i $acc/* -o modPhred/$acc
    -c dna_r9.4.1_450bps_modbases_dam-dcm-cpg_hac.cfg \
    --host ~/src/ont-guppy_3.6.1/bin/guppy_basecall_server
~/src/modPhred/src/guppy_align.py -f ref/ECOLI.fa -o modPhred/$acc -i modPhred/$acc/
↪reads/*
~/src/modPhred/src/mod_report.py -f ref/ECOLI.fa -o modPhred/$acc -i $acc/*
~/src/modPhred/src/mod_plot.py -i modPhred/$acc/mod.gz
```

Some examples of visualisation of the obtained results are described *here*.

Run modPhred (remote basecalling)

If you want to run live basecalling using remote instance of guppy (ie from GridION), you'll need to provide the IP and the port at which guppy_basecall_server is listening

```
acc=PRJEB22772
~/src/modPhred/run -f ref/ECOLI.fa -o modPhred/$acc -i $acc/* -t4 --host 10.46.1.65 -p␣
↪5556
```

6.2.2 ModPhred pipeline with basecalled Fast5 files

Download pre-basecalled test data

If you can't run basecalling, you can download pre-basecalled Fast5 files using:

```
wget https://public-docs.crg.es/enovoa/public/lpryszcz/src/modPhred/basecalled/ -q --
↪show-progress -r -c -nc -np -nH --cut-dirs=6 --reject="index.html*"
```

Run modPhred (pre-basecalled)

Running entire modPhred pipeline from basecalled Fast5 files (~4 minutes):

```
acc=PRJEB22772; ver=3.4.1
~/src/modPhred/run -f ref/ECOLI.fa -o modPhred/$acc -ri guppy$ver/$acc/* -t4
```

Instead you can run all steps one-by-one as follow:

```
~/src/modPhred/src/guppy_encode.py -o modPhred/$acc -ri guppy$ver/$acc/*
~/src/modPhred/src/guppy_align.py -f ref/ECOLI.fa -o modPhred/$acc -ri modPhred/$acc/
↪reads/*
~/src/modPhred/src/mod_report.py -f ref/ECOLI.fa -o modPhred/$acc -ri guppy$ver/$acc/*
~/src/modPhred/src/mod_plot.py -i modPhred/$acc/mod.gz
```

Some examples of visualisation of the obtained results are described *here*.

Note, here we separately basecalled Fast5 files and then ran modPhred. However in real-live, we **strongly recommend performing on-the-fly basecalling**. You'll find more usage information *here*.

6.2.3 Compare modPhred and megalodon results

You can download precomputed modPhred and megalodon results using

```
wget https://public-docs.crg.es/enovoa/public/lpryszcz/src/modPhred/final/ -q --show-
↪progress -r -c -nc -np -nH --cut-dirs=6 --reject="index.html*"
```

Now, you can compare predictions made by those tools.

```
# separate modPhred predictions for 6mA and 5mC and filter to those with 5% frequency␣
↪and at least 25 reads aligned
for f in modPhred/PRJEB22772/minimap2/*.bam.bed; do echo $f; for m in 6mA 5mC; do grep -
↪w $m $f | awk '$11>=5 && $10>=25' > $f.$m.flt.bed; done; done

# filter results to only sites with at least 5% of reads carrying modificationand at␣
↪least 25 reads aligned
for f in megalodon/PRJEB22772/*/modified_bases.???.bed; do echo $f; awk '$11>=5 && $10>
↪=25' $f > $f.flt.bed; done

# get number of predictions for each run
wc -l modPhred/PRJEB22772/minimap2/*.flt.bed megalodon/PRJEB22772/*/modified_bases*.flt.
↪bed

# draw Venn diagram for both mods
for m in 6mA 5mC; do
 ~/src/modPhred/src/mod_plot.py --venn {modPhred,megalodon}/PRJEB22772/*/*$m*.flt.bed -n␣
↪modPhred_1D modPhred_2D megalodon_1D megalodon_2D -o venn.$m.05.svg;
done
```

Above will produce Venn diagrams similar to these:

For more detailed comparison have a look in *modPhred paper*.

For more examples of downstream analyses, check *here*.

6.2.4 Test data generation

The test data was generated from PRJEB22772 by selecting only reads aligned to `NC_000913.3:1-100000` as follows:

```
acc=PRJEB22772
for d in _archives/raw/$acc/*; do
 s=`echo $d|rev|cut -f1 -d"/"|rev`
 echo `date` $d $s
 if [ ! -d ~/src/modPhred/test/$acc/$s ]; then
  mkdir -p ~/src/modPhred/test/$acc/$s
  # get read IDs
  samtools view modPhred/$acc/minimap2/$s.bam "NC_000913.3:1-100000" | cut -f1 | sort |␣
↪uniq > modPhred/$acc/minimap2/$s.bam.ids
  # subset reads
  python ~/src/ont_fast5_api/ont_fast5_api/conversion_tools/multi_fast5_subset.py -t 4 --
↪recursive -l modPhred/$acc/minimap2/$s.bam.ids -i $d -s ~/src/modPhred/test/$acc/$s
 fi
done
```

6.3 Running the pipeline

ModPhred can process DNA and RNA datasets. The only difference between the two is enabling splice-aware alignments for RNA. The type of dataset is dectected automatically from provided guppy config file `-c / --config`.

The only required inputs are:

- reference FastA sequence
- path(s) containing Fast5 files

You can provide multiple input Fast5 folders - those will be treated as separate runs/samples. If you wish to treat multiple runs as one sample, place your Fast5 files in 1 folder.

ModPhred can be executed in three modes:

- local on-the-fly basecalling
- remote on-the-fly basecalling
- without basecalling (assuming the Fast5 files were basecalled before)

We strongly recommend to use on-the-fly basecalling because it's a few times faster than running basecalling and modPhred separately. Basecalling is **the slowest step** of entire process and it produces **huge intermediate files**.

In order to see detailed description of program parameters, just execute it with `-h / --help`.

6.3.1 Local on-the-fly basecalling

Here, we assume, that you have guppy already installed in you system. ModPhred will start guppy_basecall_server in the background and stop it when it isn't needed anymore.

All you need to do is to provide path to `guppy_basecall_server` using `--host`

```
~/src/modPhred/run -f reference.fasta -o modPhred/projectName \
    -i input_fast5_folder1 [input_fast5_folder2 ... input_fast5_folderN] \
```

(continues on next page)

(continued from previous page)

```
    -c dna_r9.4.1_450bps_modbases_dam-dcm-cpg_hac.cfg \
    --host ~/src/ont-guppy_4.0.15/bin/guppy_basecall_server
```

Alternatively, if guppy_basecall_server is already running in your machine, you can provide just its port using –host localhost –port.

6.3.2 Remote on-the-fly basecalling

Here, we assume the guppy_basecall_server is already running in the remote machine. All you need to do is to provide IP address `--host` and port `--port`

```
~/src/modPhred/run -f reference.fasta -o modPhred/projectName \
    -i input_fast5_folder1 [input_fast5_folder2 ... input_fast5_folderN] \
    -c dna_r9.4.1_450bps_modbases_dam-dcm-cpg_hac.cfg \
    --host 172.21.11.186 --port 5555
```

6.3.3 Without basecalling

Make sure your Fast5 files are basecalled with guppy v3.2+ with models trained to detect modifications.

Running modPhred pipeline is as easy as:

```
~/src/modPhred/run -f reference.fasta -o modPhred/projectName \
    -i input_fast5_folder1 [input_fast5_folder2 ... input_fast5_folderN]
```

For more usage examples, please have a look in *test dataset*.

How to check if my Fast5 files are basecalled with modifications?

modPhred will fail if you don't enable on-the-fly basecalling and try to run it on Fast5 files that are not basecalled or that are basecalled withouth modifications. You can check that quickly using `h5ls`: * If you see `*/Basecall_*` entries, it means your Fast5 is basecalled. * If you see `*/Basecall_*/BaseCalled_template/ModBaseProbs` enries, it means your Fast5 is basecalled with modificatoins.

- If you don't see any of the above, your Fast5 files are not basecalled at all.

```
> h5ls -r project/sample/workspace/batch_0.fast5 | less
/                        Group
/read_XXXXXX Group
/read_XXXXXX/Analyses Group
/read_XXXXXX/Analyses/Basecall_1D_000 Group
/read_XXXXXX/Analyses/Basecall_1D_000/BaseCalled_template Group
/read_XXXXXX/Analyses/Basecall_1D_000/BaseCalled_template/Fastq Dataset {SCALAR}
/read_XXXXXX/Analyses/Basecall_1D_000/BaseCalled_template/ModBaseProbs Dataset {10527, 6}
/read_XXXXXX/Analyses/Basecall_1D_000/BaseCalled_template/Move Dataset {54990}
/read_XXXXXX/Analyses/Basecall_1D_000/BaseCalled_template/Trace Dataset {54990, 8}
/read_XXXXXX/Analyses/Basecall_1D_000/Summary Group
...
```

6.3.4 Using custom-modifications models

ModPhred supports DNA and RNA modification-aware guppy models. If you wish to use `my_custom_model.cfg` model, first copy `.cfg` and `.jsn` files to guppy `/data` directory (ie. `~/src/ont-guppy_4.0.15/data`) along other basecalling models. Then execute modPhred as follows:

```
~/src/modPhred/run -f reference.fasta -o modPhred/projectName \
    -i input_fast5_folder1 [input_fast5_folder2 ... input_fast5_folderN] \
    -c my_custom_model.cfg \
    --host ~/src/ont-guppy_4.0.15/bin/guppy_basecall_server
```

6.3.5 Processing (very) large datasets

There are several ways of speeding up entire analysis for very large datasets.

- modEncode: process each sample or (or even subsets of each run) separately using guppy_encove_live.py. Ideally, each subset will be processed on dedicated GPU (local or remote). Here, providing more than 6 cores per job brings no improvement, since modEncode is primarily GPU-bound.
- modAlign: no much can be done, since every sample has to produce one BAM file. Beside, modAlign is by far the fastest step.
- modReport: process each chromsome (or even subsets of chromosome) as separate job. Make sure to provide as many cores as possible to each job.

```
# run mod_encode
~/src/modPhred/src/guppy_encode_live.py

# run mod_report
~/src/modPhred/run -o outdir [...] --chr chr1
~/src/modPhred/run [...] --chr chr2
...
~/src/modPhred/run [...] --chr chrN

# combine the results for individual chromosomes
~/src/modPhred/src/merge_chr.py outdir/mod.gz.chr*.gz

# rerun modPhred without --chr to generate missing result files
~/src/modPhred/run -o outdir [...]
```

6.4 Program output

If the program was execute command like

```
~/src/modPhred/run -f reference.fasta -o modPhred/projectName -i guppy_out/projectName/
↪sample*/workspace
```

modPhred will generate in the output directory `modPhred/projectName`:

- mod.gz - internal format with all predicted positions with likely modifications
- minimap2/*.bam - alignments in BAM format and with encoded modifications. One BAM file will be generated for every sample (directory) provided as input `-i`. Modification probabilities can be *viewed directly in IGV*.

- .bed - annotated positions with modifications as bedMethyl-formatted files
  - mod.bed - combined report of positions with detected modifications in any of the samples
  - minimap2/*.bam.bed - modified sites reported for each run separetely
- mod.gz.svg - QC plots
  - and additional plots in plots/ directory
- reads/sampleName/**.fastq.gz - basecalled reads in FastQ format
- reads/sampleName/**.fastm.gz - basecalled reads with modifications encoded in FastM format

6.4.1 Data formats

While using modPhred it'll be good to familirised yourself with couple of data formats.

bedMethyl

modPhred reports information about modified positions in bedMethyl format. This is tab-delimited files, compatible with BED (positions are 0-based, half-open) with several additional fields:

1. Reference chromosome or scaffold
2. Start position in chromosome
3. End position in chromosome
4. Name of item - short modification name
5. Score - median modification probability scaled from 0-1000.
6. Strandedness, plus (+), minus (-), or unknown (.)
7. Start of where display should be thick (start codon) - same as 2.
8. End of where display should be thick (stop codon) - same as 3.
9. Color value (RGB) - different color to various modifications, although if more than 7 mods the colors may repeat. Color intensity depends on modification frequency (darker means modification is more frequent).
10. Coverage - number of reads at this position
11. Percentage of reads that show given modification at this position in the genome/transcriptome

For example, the output for `NC_000913.3:1,061-1,253` looks like this:

```
NC_000913.3     1089    1090    5mC     903     +       1089    1090    0,139,0 454     ␣
↪55
NC_000913.3     1091    1092    5mC     806     -       1091    1092    0,97,0  354     ␣
↪38
NC_000913.3     1167    1168    6mA     839     +       1167    1168    0,0,155 468     ␣
↪61
NC_000913.3     1168    1169    6mA     871     -       1168    1169    0,0,187 464     ␣
↪74
NC_000913.3     1207    1208    5mC     806     +       1207    1208    0,108,0 431     ␣
↪42
NC_000913.3     1209    1210    5mC     968     -       1209    1210    0,175,0 407     ␣
↪69
```

bedMethyl files can be visualised in many genome browsers ie IGV.

mod.gz

This is internal, tab-delimited format that contain information about all predicted positions with likely modifications. Position are 1-based (similar to VCF format).

For example, the mod.gz for `NC_000913.3:1,061-1,253` region will look like that:

```
###
# Welcome to modPhred (ver. 1.0b)!
#
# Executed with: ~/src/modPhred/run -f ref/ECOLI.fa -o modPhred/PRJEB22772 -i guppy3.4.1/
↪PRJEB22772/MARC_ZFscreens_R9.4_1D-Ecoli-run_FAF05145/workspace guppy3.4.1/PRJEB22772/
↪MARC_ZFscreens_R9.4_2D-Ecoli-run_FAF05711/workspace -t3
#
# For each bam file 4 values are stored for every position:
# - depth of coverage (only positions with >=25 X in at least one sample are reported)
# - accuracy of basecalling (fraction of reads having same base as reference, ignoring␣
↪indels)
# - frequency of modification (fraction of reads with modification above given threshold)
# - median modification probability of modified bases (0-1 scaled).
#
# If you have any questions, suggestions or want to report bugs,
# please use https://github.com/lpryszcz/modPhred/issues.
#
# Let's begin the fun-time with Nanopore modifications...
###
chr     pos     ref_base        strand  mod     modPhred/PRJEB22772/minimap2/MARC_
↪ZFscreens_R9.4_1D-Ecoli-run_FAF05145.bam depth        modPhred/PRJEB22772/minimap2/
↪MARC_ZFscreens_R9.4_1D-Ecoli-run_FAF05145.bam basecall_accuracy    modPhred/PRJEB22772/
↪minimap2/MARC_ZFscreens_R9.4_1D-Ecoli-run_FAF05145.bam mod_frequency        modPhred/
↪PRJEB22772/minimap2/MARC_ZFscreens_R9.4_1D-Ecoli-run_FAF05145.bam median_mod_prob      ␣
↪modPhred/PRJEB22772/minimap2/MARC_ZFscreens_R9.4_2D-Ecoli-run_FAF05711.bam depth       ␣
↪ modPhred/PRJEB22772/minimap2/MARC_ZFscreens_R9.4_2D-Ecoli-run_FAF05711.bam basecall_
↪accuracy    modPhred/PRJEB22772/minimap2/MARC_ZFscreens_R9.4_2D-Ecoli-run_FAF05711.bam␣
↪mod_frequency        modPhred/PRJEB22772/minimap2/MARC_ZFscreens_R9.4_2D-Ecoli-run_
↪FAF05711.bam median_mod_prob
NC_000913.3     244     C       -       5mC     444     0.910   0.014   0.806   120     ␣
↪0.958   0.050   0.581
NC_000913.3     420     C       +       5mC     464     0.978   0.713   0.935   132     ␣
↪0.962   0.644   0.935
NC_000913.3     422     C       -       5mC     351     0.604   0.328   0.806   103     ␣
↪0.621   0.369   0.839
...
NC_000913.3     1090    C       +       5mC     454     0.941   0.520   0.903   134     ␣
↪0.970   0.545   0.871
NC_000913.3     1092    C       -       5mC     354     0.833   0.379   0.806   103     ␣
↪0.854   0.320   0.806
NC_000913.3     1168    A       +       6mA     468     0.998   0.607   0.839   143     ␣
↪1.000   0.573   0.806
NC_000913.3     1169    A       -       6mA     464     0.996   0.735   0.871   131     ␣
↪1.000   0.557   0.806
NC_000913.3     1208    C       +       5mC     431     0.910   0.297   0.806   135     ␣
↪0.963   0.422   0.806
NC_000913.3     1210    C       -       5mC     407     0.865   0.686   0.935   119     ␣
```

(continues on next page)

(continued from previous page)

```
↪0.899   0.681   0.968
```

FastQ

A text-based format for storing a nucleotide sequence and its corresponding quality scores, both encoded with a single ASCII character. You can find more details here.

FastM

A variation of FastQ format, in which instead of quality scores, we store probability of the base being modified. You can find more details on modification encoding *here*.

BAM

A binary format for storing raw genomic data. Reads in BAM files are typically aligned to reference, compressed and sorted by reference position. **Note, we store modification probability for every base instead of base qualities.** More information about modification probability encoding can be found *here*.

Original base qualities can be saved in BAM under OQ tag using `-s / --storeQuals`. You can find more information about SAM/BAM format at htslib websites.

6.4.2 Why base Y is detected as modified, while model only reports modifications for X?

Let's assume your model detects 5mC. Sometimes non-C reference bases may be detected as modified. This may happend for several reasons:

- mis-alignment - apparent 5mC bases were incorrectly aligned to A, G or T reference
- mis-calling - apparent A, G or T bases were mispredicted as 5mC
- true biological variation - for example: * genotype of your sample may be different that this of your reference genome, thus true base will be C instead of A, G or T * heterozygous positions - a variant can have alternative allel being modified, thus 5mC may be true * variability in population - if you sequence pooled/mixed/tumor sample, some fraction of the cells may carry alternative alleles

6.5 Visualisation

Instead of looking at text files described below, you can visualies some aspects of the DNA/RNA modifications.

6.5.1 QC plots

QC plots are generated automatically at the end of modPhred pipeline run. For all detected positions with likely modifications from all samples, those plots inform about various distribution for all modifications:

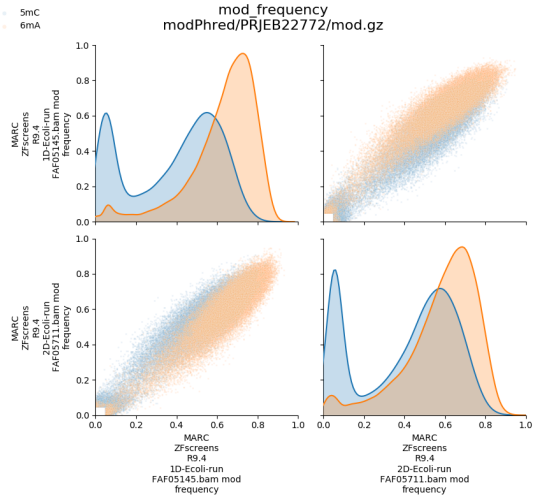
- depth of coverage
- basecall accuracy (taking into account substitutions, but ignoring indels)
- modification frequency
- and median modification probability of bases with modifications

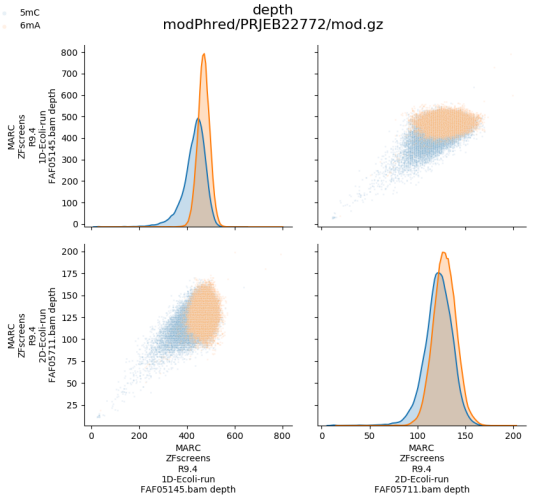
Compare samples

In addition, modPhred can generate pairwise plots of above measurements for all modifications, for all modified positions between all samples.

```
~/src/modPhred/src/mod_plot.py --scatter -i modPhred/PRJEB22772/mod.gz
```

Above will produce plots similar to these:





6.5.2 Modification frequency in certain regions

You can plot type and frequency of modifications for selected regions as follows:

```
~/src/modPhred/src/mod_plot.py -i modPhred/PRJEB22772/mod.gz -b "NC_000913.3:1,061-1,253"
```

This will create directory modPhred/PRJEB22772/plots with separate figures for each region that was provided either via command line of BED-file. Every figure will plot frequency of modification for all modifications (in different colours) and for every run/sample in different panel separately for + (positive values of frequency) and - strand (negative values of frequency).

From above plot it's very easy to conclude that enzymes depositing 5mC and 5mA in DNA are likely acting on sequence motifs that are palindromic because modification on + strand is always followed by modification on - strand.

6.5.3 Visualisation in genome browsers (IGV)

Modification probabilities for individual reads and bases can be viewed directly in genome browsers (such as IGV) given you have model with 1 modification per base. If you have more than 1 modification per base the probabilities of 2nd and further modificaiton for given base will be rendered improperly.

To do so:

1. open IGV,
2. load your reference genome (Genomes > Load genome from File… and select file `ref/ECOLI.fa`),
3. load BAM files generated by modPhred (File > Load from File… and select `.bam` files inside `/minimap2` folder),
4. load BED files generated by modPhred (File > Load from File… and select `.bed` files inside `/minimap2` folder),
5. select some region ie `NC_000913.3:1,061-1,253`
6. collapse alignments (mouse right-click and tick Collapsed)
7. make all bases visible (mouse right-click and tick Show all bases)
8. and shade by quality (mouse right-click and tick Shade base by quality)

You should see something like this:

Now you can easily see positions that contain modified bases:

- bedMethyl tracks (BED files) show different modifications in different colours, plus the frequency of modification is depicted as the colour intensity (darker means more reads are modified at given position)
- and when you zoom-in you can clearly see the probability of this base being modified for every read and every base in the read - the intensity of base color is proportional to the probability of that base being modified.

Cool, right?

6.6 Correlation between positions

You can plot correlation between modified positions within certain regions. This is as easy as:

```
~/src/modPhred/src/mod_correlation.py -i modPhred/PRJEB22772/mod.gz -r NC_000913.3:1-5000
```

This will generate something like that:

In addition, using below you can narrow those plots to only: * modifications from particular strand * and one modification from particular strand

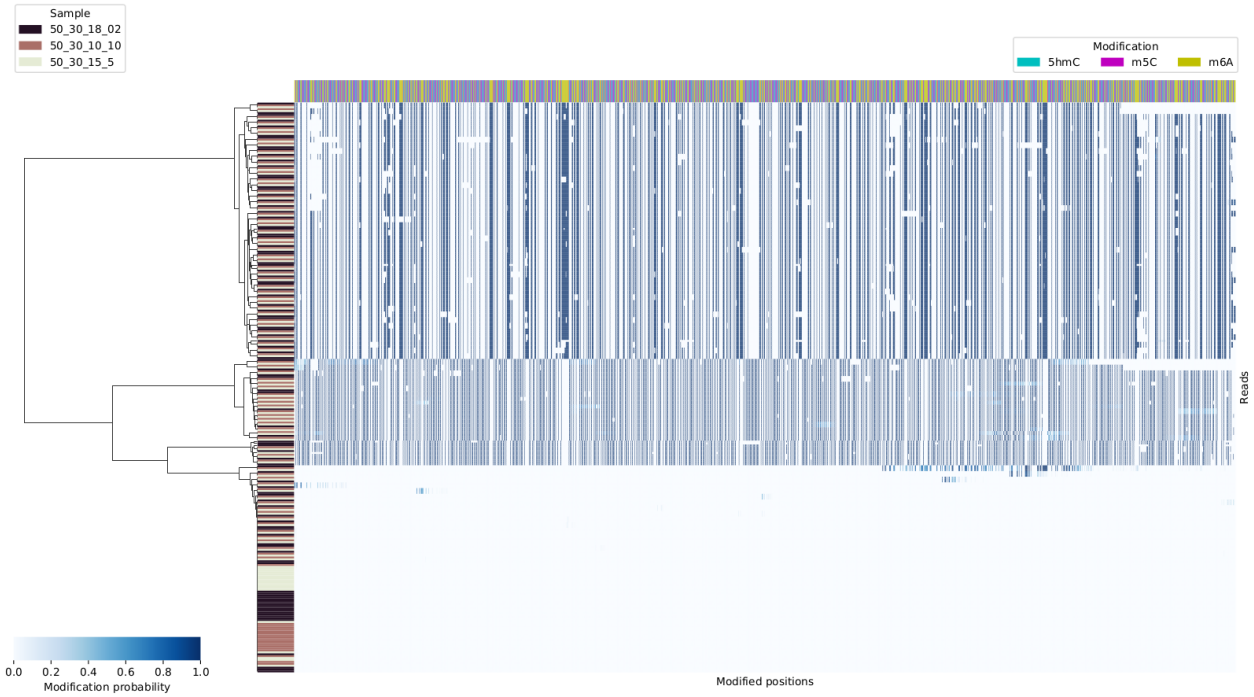
```
~/src/modPhred/src/mod_correlation.py -i modPhred/PRJEB22772/mod.gz -r NC_000913.3:1-
↪5000+
~/src/modPhred/src/mod_correlation.py -i modPhred/PRJEB22772/mod.gz -r NC_000913.3:1-
↪5000+ --mod 6mA
```

6.7 Read clustering

You can also cluster reads by modification status. Note, this currently cluster entire reference sequences, so it's reasonable to run it only with individual transcritps. Definitely, avoid clustering reads from entire chromosomes.

```
~/src/modPhred/src/mod_cluster.py --minfreq 0.01 -i mod.gz -e pdf
```

This will produce plots similar to this one



You can specify regions of interest using `-r`, for example if you want to cluster reads that aligned to chr1 at positions between 100 and 200 on the + strand, you'd run it as follows:

```
~/src/modPhred/src/mod_cluster.py --minfreq 0.01 -i mod.gz -e pdf -r chr1:100-200+
```

6.7.1 Clustering details

We perform clustering of reads as follows:

- user selects a region for read clustering ie. certain transcript or stranded genomic region
- identify a set of positions that are modified in that region ie. more than 20% of reads are modified (`--minfreq 0.2`) with least 25 reads aligned (`-d / --mindepth 25`) in at least one sample. Importantly, we include all types of modifications in the analysis by default. Analysis can be limited to single modification using `--mod`.
- extract modification probabilities for a set of modified positions from all reads from all samples. Note, the reads are filtered by mapping quality `-m / --mapq 15` and their overlap with a selected region: by default a read has to cover at least 80% of the selected region `--minAlgFrac 0.8`.
- perform hierarchical clustering & save a figure. Only reads are clustered (modified positions are not clustered). We're using implementation from seaborn.clustermap.

At this point we don't report read clusters.

6.8 Scripts

6.8.1 mods_from_bams

If you happen to have BAM files with modifications already encoded in them and just want to process them with modPhred all you need to do is:

- make new directory ie `mods_from_bams/CC.raw_1x`
- copy `modPhred.pkl` there (it stores info about mods that are present in BAMs)

```
rsync -av modPhred/raw_1x.m6A/modPhred.pkl mods_from_bams/CC.raw_1x
```

- prepare your BAM files in this output dir - BAM files have to be in `mods_from_bams/CC.raw_1x/minimap2`
- and run the script

```
~/src/modPhred/src/mods_from_bams.py -o mods_from_bams/CC.raw_1x \
   -i mods_from_bams/CC.raw_1x/minimap2/*.bam -f ~/cluster/rna_mods/ref/curlcake.fa
```

6.9 Basecalling

Basecalling is performed using guppy basecaller. You can choose to either:

- basecall Fast5 before (guppy ver. >3.2 is supported)
- or run live basecalling (guppy ver. >3.6 is supported)

In the live mode, basecalling is performed with default settings. By default, CpG/dam/dcm model (`dna_r9.4.1_450bps_modbases_dam-dcm-cpg_hac.cfg`) is used. This can be changed using `-c / --config` parameter.

Note, if you choose to run live basecalling, you'll need to install matching version of pyguppyclient.

6.9.1 Modification-aware models

Modification aware-model should be used for basecalling. At this point, there are no publically available models for RNA modifications. The RNA model provided in this repository has been trained with *in vitro* transcribed molecules in which either all bases are modified or all are unmodified, and because of that it isn't applicable to biological samples in which only some bases are expected to be modified. You can find more details in the supplementary information of the manuscript.

6.10 Encoding of modifications

Since basecall qualities are not very informative for Nanopore sequencing (typically valued between 7-12), we decided to store modification probabilities as FastQ base qualities. This is achieved as follows.

Guppy basecaller reports probability of base being modified in Fast5 files under `/Analyses/Basecall_1D_000/BaseCalled_template/ModBaseProbs`. Those probabilities are stored as 8-bit integers scaled from 0 (no modification) to 255 (modification) and are reported separately for all modifications that given model has been trained for. If we wanted to store just one modification per each base, we could simply rescale those values to ASCII scale (0-93) and store it as is (base qualities in FastQ files are stored as ASCII characters). However, especially for RNA modifications, it would be beneficial to store information about multiple possible modifications for given base (ie. m5C and 5hmC are quite common for C) as there are over 170 known modifications in RNA. If you want to store information for 3

modifications of every base, this is up to 12 modifications in total, one can rescale modification probabilities into 31 values (instead of 255 or 93) and store the probability of the modification with the highest probability for given base. For example, if you want to store information of about 3 modifications of C (m5C, 5hmC and m3C), you can assign values of:

1. 0-30 to the first (m5C, PHRED 33-63),
2. 31-61 to the second (5hmC, PHRED 64-94)
3. and 62-92 to the third (m3C, PHRED 95-125) modification of C.

Now imagine in some read the base C (position 2635 below) has probabilities of 12, 247, 9 for m5C, 5hmC and m3C, respectively. Since only one probability can be stored for every base in FastQ, the most informative would be to store probability of 5hmC since it has the highest probability. We would rescale the probability of 247 to 31-unit scale as follows:

$$Q = 255 * maxp1, p2, p3/31 + 31 * Mi$$

where *Mi* (modification index) is

- 0 if first (m5C),
- 1 if second (5hmC)
- or 2 if third (m3C) modification of C
- has the highest probability of modification (p).

Such storage of information assures simplicity and versatility. Since probability of modification is stored inside FastQ no external databases or additional files are needed for calculation of modifications as the per base modification probabilities for all reads will be stored also in BAM files that are derived from FastQ. What's more, our encoding system is flexible - it can be easily adjusted to encode more modifications. For example 9 modifications per each base, this is up to 36 modifications in total, can be encoded given 10-value scale per modification is enough (instead of 31 in default 3 values per modifications).

Theoretically our system allow encoding information about up to 368 modifications (92 per base) directly in FastQ format given we limit the information about modifications status to binary form, meaning either base is carrying one of the 368 modifications or not.

6.11 Alignments

Alignments are performed using minimap2 and sorted BAM files are stored in `out_dir/minimap2` directory.

We use recommended settings for mapping ONT reads: `-ax map-ont`. For RNA samples, splice-aware mapping is automatically performed: `-ax splice`.

Since the modification status is encoded within FastQ, it'll be propagated through downstream analyses such as alignment and stored in BAM files. And having modification status encoded in BAM allows visualisation of modification probabilities directly in genome browsers.

You can find more info about data formats here.

6.12 Detection of modifications

In order to detect and annotated modified positions, modPhred scans read alignments and for every position of reference genome/transcriptome it calculates

- depth of coverage,
- basecall accuracy,
- modification frequency
- and median modification probability.

Those values are reported into `mod.gz` file.

By default, modPhred reports only sites with modifications fulfilling following criteria:

- minimum mapping quality of 15 - only reads with mapping quality of 15 or more are considered
- minimum sequencing depth of 25 - at least 25 reads for given positions
- at least 0.05 frequency of modification - at least 5% of reads being modified for this position
- at least 0.5 modification probability - only bases with modification probability of 50% or more are considered as truly modified.

The default values can be adjusted with several parameters:

```
./run -h
...
-m --mapq               min mapping quality [15]
-d --minDepth           min depth of coverage [25]
```

(continues on next page)

(continued from previous page)

```
--minModFreq            min modification frequency per position [0.05]
--minModProb            min modification probability per base [0.5]
...
```