**Randomers Analysis**

Artificial DNA was retrieved from IDT. Two sets of oligos were designed. The first set contains Methyl-O6-Guanine (O6MeG) in the middle of the sequence, surrounded by 12 random nucleotides up and down stream and constant parts. The second set has unmodified guanine (G) in the middle and is surrounded by 30 random nucleotides up and down stream. Both sets are 103 nucleotides long.

**First set, O6MeG:**

AGC TCA CGT GAG TCA TAC TAG CGC GCT GAG ATG AGT CTA (N:25252525)(N)(N) (N)(N)(N) (N)(N)(N) (N)(N)(N) /i6OMe-dG/(N)(N) (N)(N)(N) (N)(N)(N) (N)(N)(N) (N)AC GCA CAC GCA CTA CGT CGC AAG TGG AAC CTC TCT GAC A

**Second set, unmodified guanine:**

AGC TCA CGT GAG TCA TAC TAG (N:25252525)(N)(N) (N)(N)(N) (N)(N)(N) (N)(N)(N) (N)(N)(N) (N)(N)(N) (N)(N)(N) (N)(N)(N) (N)(N)(N) (N)(N)(N) G(N)(N) (N)(N)(N) (N)(N)(N) (N)(N)(N) (N)(N)(N) (N)(N)(N) (N)(N)(N) (N)(N)(N) (N)(N)(N) (N)(N)(N) (N)GC AAG TGG AAC CTC TCT GAC A

The two sets were sequenced using Oxford Nanopore Technology, MinION. The sequencing outputs are pod5 files.

Each set yielded several pod5 files. For convenience the files of each set were united using pod5 merge command, leaving us with two files: *can.pod5* for the first set and *mod.pod5* for the second.

The pod5 files contain the raw electric signal from the MinION sequencer. We would like to translate it to nucleotides, or in other words to do base calling. The base calling was performed using Dorado package:

dorado basecaller bc\_models/dna\_r10.4.1\_e8.2\_400bps\_sup@v4.2.0/ data/can/pass/ --emit-moves > can/can.bam

dorado basecaller bc\_models/dna\_r10.4.1\_e8.2\_400bps\_sup@v4.2.0/ data/mod/pass/ --emit-moves > mod/mod.bam

This step left us with two unaligned bam files, containing the reads from the sets. Due to the random context surrounds the middle guanine and O6MeG we do not have a reference “genome” to align the reads to. Therefore, we took the reads and made them their own reference (The code is found in ‘*Random Sequences Selection for Reference.ipynb’* file).

The bam files were converted to sam files using samtools view command. The sam files were loaded into pandas DataFrame object in Jupyter Lab. Using the constant parts of the sequences and regular expression (regex) the sequences of interest were identified and filtered:

can\_sequences = can\_sequences.str.findall("TCATACTAG.............................................................GCAAGTGGA")

mod\_sequences = mod\_sequences.str.findall("ATGAGTCTA.........................ACGCACACG")

these sequences were used to build a reference for the canonical reads and for the modified reads.

The references that were built are: 1. A reference for the canonical reads, 2. a reference for the modified reads with G in the middle, 3. a reference for the modified reads with A in the middle.

These references will be used to train two models. The first will distinguish between G and O6MeG and the second between A and O6MeG.

Now that we have the references we can align the reads with dorado aligner:

dorado aligner -k 15 -w 1 -v reference/ref\_can.fa can/can.bam > can/can\_mappings.bam

dorado aligner -k 15 -w 1 -v reference/ref\_G.fa mod/mod.bam > mod/mod\_mappings\_G.bam

dorado aligner -k 15 -w 1 -v reference/ref\_A.fa mod/mod.bam > mod/mod\_mappings\_A.bam

**Quality Control**

The bam files were sorted and indexed using samtools. The unaligned files were also aligned with bowtie2 for visualization reasons.

We used samtools flagstat, fastQC, nanoPlot, pycoQC and IGV. The codes are on the cluster.

**Training a Model**

Each of the aligned bam files were split with ratio of 1:99 for test set and train set correspondingly, an example of the split:

export n=`samtools view can\_mappings.bam | wc -l`

samtools view can\_mappings.bam | shuf | split -a1 -d -l $(( $n \* 99 / 100 )) - can\_mappings.sam

samtools view -H can\_mappings.bam > can\_mappings.bam.header

cat can\_mappings.bam.header can\_mappings.sam0 | samtools view -b > can\_mappings.train.bam

cat can\_mappings.bam.header can\_mappings.sam1 | samtools view -b > can\_mappings.val.bam

rm can\_mappings.sam0 can\_mappings.sam1 can\_mappings.bam.header

than using *remora* package we ran the commands ‘prepare -> config -> train’.

We are going to train overall 4 models. Two models to distinguish between G and O6MeG and the other two models for A and O6MeG.

One model will be used to visualize and assess its abilities and will be trained on the training set. The other for future uses and will be trained on all the data.

‘all\_G’, ‘train\_G’, ‘all\_A’, ‘train\_A’

**Prepare**

For preparation of the data, we need to supply the algorithm with locations of unmodified G’s and O6MeGs. The locations were allocated as bed files.

To create those bed files, we used the reference we built before. We iterated through the sequences and when we found G or A we appended them to the fit bed file. The code is found in *FocusBedFile.ipynb*.

Our first thought was to provide in the focus file only the middle G’s – the modified and the unmodified. Then we tried to expand our focus file and used every G/A that is really in the random context meaning at least 4 nucleotides into the random area. Not only that, we also took the unmodified G/A’s that are found in the random area in the files of the modified reads.

Now that we have the focus bed file we can use the prepare command of remora:

remora \

dataset prepare \

data/can/pass/ \

can/can\_mappings.bam \

--output-path can/can\_chunks\_all\_G \

--focus-reference-positions bed/focus\_can\_G.bed \

--refine-kmer-level-table kmer\_models/dna\_r10.4.1\_e8.2\_400bps/9mer\_levels\_v1.txt \

--refine-rough-rescale \

--motif G 0 \

--mod-base-control

The whole code is found on the cluster, dataset\_prepare.sh.

**Config**

The data is almost prepared for training, config unites the labeled files as follows:

remora \

dataset make\_config \

dataset\_all\_G.jsn \

can/can\_chunks\_all\_G \

mod/un\_mod\_chunks\_all\_G \

mod/mod\_chunks\_all\_G \

--log-filename dataset\_all\_G.log

**Train**

remora \

model train \

dataset\_all\_G.jsn \

--model /sci/backup/iris.lavon/nir.butbul/software/remora-master/models/ConvLSTM\_w\_ref.py \

--device 0 \

--output-path all\_G\_model

**Assessing the Model**

First, it is important to look at the output files of each step. We want to make sure that the chunks extracted from the reads for training make sense to us. Additionally, when looking at the output of the train command we can see details regarding the performance of the model. For visualization reasons we created a validation set, this 1% of the data we left aside beforehand. Now we can use the model we trained only on the training set to assess its performance on the validation set.

For that we use remora infer command:

remora \

infer from\_pod5\_and\_bam \

data/can/pass/ \

can/can\_mappings.val.bam \

--model train\_G\_model/model\_best.pt \

--pore dna\_r10.4.1\_e8.2 \

--basecall-model-type sup \

--modified-bases O6MeG \

--out-bam can/can\_infer\_G\_ra.val.bam \

--log-filename can/can\_infer\_G\_ra.val.log \

**--reference-anchored \**

--device 0

**[Important side note:** When inferring the model goes through the reads, when finding a G in the read it “asks” the question what are the chances that this G is modified or unmodified. However, we know that O6MeG is mostly base-called as A, meaning that its voltage read by the nanopore sequencer is more like adenine rather than guanine. Therefore, when going through the reads it is unlikely that when encountering G, it will be O6MeG and it is very likely that we will ignore O6MeG lesions. That is because O6MeG looks like A’s and the model will never ask whether a A is an O6MeG.

To get around this challenge, we trained a model to distinguish between A and O6MeG. Besides with the model of the G’s we used a tag called ‘reference-anchored’. This tag changes the default behavior of the inferring process. Now instead of going through the reads and finding G’s, it goes through the reads, asks where are the G’s suppose to be in the reference and if in the reference occurs G than it provides us with the chances of interest.

Therefore, when using this tag we not only get the modifications but we might also get the G to A mutations.**]**

After inferring we can run remora validate command. For that we need to create a ground truth bed file. We can do it in a similar way we created the focus files. Then we plotted a ROC curve and calculated the Area Under the Curve (AUC). The code is found in *ROC\_AUC.ipynb*.