# seqPatch: a R package detecting DNA modifications from SMRT sequencing data by modeling sequence context dependence of polymerase kinetic

Zhixing Feng

December 8, 2012

## Introduction

*seqPatch* is a handy R package that implement SMRT sequencing based DNA modification detection method described in (Zhixing Feng et al, 2012). *seqPatch* can detect DNA modifications with or without control sample. The method use Inter-Pulse duration(IPD) generated by SMRT sequencing, which reflects kinetic of DNA polymerase and is sensitive to a wide range of DNA modifications, to detect DNA modifications.

## Prepare data

### Get alignment data by phb5

*seqPatch* depends on *pbh5* Package, and its inputs are objects returned by function *getAlignmentsWithFeatures* and *alnIndex* in *pbh5* package. We can get them by installing *phb5* and running the following code(not really an executable example).

```
> library(pbh5)
> cmpH5.native <- PacBioCmpH5(filename.native)
> alnsF.native <- getAlignmentsWithFeatures(cmpH5.native, fxs = list(IPD = getIP
```

```
> alnsIdx.native <- alnIndex(cmpH5.native)
> cmpH5.ctrl <- PacBioCmpH5(filename.ctrl)
> alnsF.ctrl <- getAlignmentsWithFeatures(cmpH5.ctrl, fxs = list(IPD = getIPD))
> alnsIdx.ctrl <- alnIndex(cmpH5.ctrl)
```

where *filename.native* is path of cmp.h5 file, which stores aligned reads as well as kinetic information of the native sample, generated by smrtpipe, and *filename.ctrl* is for the control sample. *alnsF.native* and *alnsF.ctrl* contain alignments and kinetic information for each read, and *alnsIdx.native* and *alnsIdx.ctrl* contains other alignment information for each read, such as strand and genome positions it aligned to, name of reference genome(i.e. string after > in fasta file of reference genome), etc.

### pre-processing

before detecting DNA modifications, IPD has to be Box-Cox transformed and normalized.

```
> library(seqPatch)
> load(paste(system.file(package = "seqPatch"), "/data/test.Rdata",
+     sep = ""))
> alnsF.native <- transformIPD(alnsF.native, 0.02, -0.08)
> alnsF.ctrl <- transformIPD(alnsF.ctrl, 0.02, -0.08)
> alnsF.native <- normalizeByMovie(alnsF.native, alnsIdx.native)
> alnsF.ctrl <- normalizeByMovie(alnsF.ctrl, alnsIdx.ctrl)
```

## Detect DNA modifications

DNA sequence of reference genome is needed to detect modifications, which can be got by

```
> genomeSeq <- getGenomeSeq(fastafilename)
```

For modification detection, if you have both control sample and historical data, you can use them both by the following code

```
> genomeF.native <- getFeaturesAlongGenome(alnsF.native, alnsIdx.native)
> genomeF.ctrl <- getFeaturesAlongGenome(alnsF.ctrl, alnsIdx.ctrl)
> detect.hieModel <- detectModification(genomeF.native, genomeF.ctrl,
```

```
+     genomeSeq, context.effect, method = "hieModel", left.len = 6,
+     right.len = 1)
```

This code would combine IPD in control sample (*genomeF.ctrl*) and IPD of homologous positions(positions that have the same sequence context) in historical data (*context.effect*, we will talk about how to get it in the next section) for detection. If you do NOT have a control sample, you can detect modifictions by only use IPD of homologous positions in historical data. The code is

```
> detect.hieModel <- detectModification.NC(genomeF.native, genomeSeq,
+     context.effect, method = "hieModel", left.len = 6, right.len = 1)
```

detect.hieModel$pos$refname$LR_log is vector that contains log-likelihood ratio for each position in forward strand of the genome, where larger value means the very base is more likely to be modified. "refname" means the name of reference genome, for example, chr1, chr2, etc. detect.hieModel$genome.start.pos$refname is the genome position of the first value of detect.hieModel$pos$refname$LR_log, and the positions of the left most base of a genome is 1. detect.hieModel$neg contains the information for backward strand. Note that "left.len" and "right.len" are number of upstream bases and number of downstream bases of sequence context respectively, the default values are left.len=6, right.len=1. One should make sure that they are the same with "left.len" and "right.len" in "getContextEffectByPos" described in the next section.

# Build your own "context.effect"

Now we will show how to get *context.effect* in the previous section. Firstly, we need to find a historical dataset that contains no modification(for example, whole genome amplified sample or sample lack of certain enzyme). Suppose the file name of that data is *filename.historical*, we can *context.effect* by the following code

```
> library(seqPatch)
> library(pbh5)
> cmpH5.historical <- PacBioCmpH5(filename.historical)
> alnsF.historical <- getAlignmentsWithFeatures(cmpH5.historical,
+     features = "IPD")
> alnsIdx.historical <- alnIndex(cmpH5.historical)
```

```
> genomeF.historical <- getFeaturesAlongGenome(alnsF.historical,
+     alnsIdx.historical)
> genomeSeq <- getGenomeSeq(fastafilename)
> context.effect <- getContextEffectByPos(genomeF.historical, genomeSeq,
+     left.len = 6, right.len = 1)
```

where left.len is number of upstream bases in sequence context, and right.len is for downstream bases. *context.effect* contains IPD of positions that have the context(i.e. homologous positions).

# An executable example

Here we show an executable example to demonstrate how *seqPatch* work.

```
> library(seqPatch)
> load(paste(system.file(package = "seqPatch"), "/data/test.Rdata",
+     sep = ""))
> genomeSeq <- getGenomeSeq(paste(system.file(package = "seqPatch"),
+     "/data/refgenome.fasta", sep = ""))
> alnsF.native <- transformIPD(alnsF.native, 0.02, -0.08)
> alnsF.ctrl <- transformIPD(alnsF.ctrl, 0.02, -0.08)
> alnsF.native <- normalizeByMovie(alnsF.native, alnsIdx.native)
> alnsF.ctrl <- normalizeByMovie(alnsF.ctrl, alnsIdx.ctrl)
> genomeF.native <- getFeaturesAlongGenome(alnsF.native, alnsIdx.native)
> genomeF.ctrl <- getFeaturesAlongGenome(alnsF.ctrl, alnsIdx.ctrl)
> detect.hieModel <- detectModification(genomeF.native, genomeF.ctrl,
+     genomeSeq, context.effect, method = "hieModel", left.len = 6,
+     right.len = 1)
> detect.hieModel.NC <- detectModification.NC(genomeF.native, genomeSeq,
+     context.effect, method = "hieModel", left.len = 6, right.len = 1)
```

# Practical recommendations

SMRT sequencing is senstitive a wide range of DNA modifcations including m6A, m4C m5C etc. However, m6A have much stronger signal, and m4C has weaker signal, m5C has the weakest. For m6A and 8-oxoG, one don't need a whole genome amplified(WGA) sample (control sample), and can only use

historical data to achieve good accuracy. However, one has to generate WGA sample to detect m4C accurately. Coverage has big impact on accuracy. For m6A, the recommanded coverage is 50x native sample, and for m4C, one need 100x native sample and 100x control sample. For all types of modificaitons, combining control sample and historical can always imcrease accuracy.