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

Zhixing Feng

December 7, 2011

Introduction

seqPatch is a handy R package that implement method described in (Zhixing Feng et al, 2012). *seqPatch* can detect DNA modifications with or without control sample. Readers can find details about model and algorithm in (Zhixing Feng et al, 2012), and readers who want to take a quick look at how seqPatch can jump to section, ”An executable example”, directly.

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.

```
> library(pbh5)
> cmpH5.native <- PacBioCmpH5(filename.native)
> alnsF.native <- getAlignmentsWithFeatures(cmpH5.native, features = "IPD")
> alnsIdx.native <- alnIndex(cmpH5.native)
> cmpH5.ctrl <- PacBioCmpH5(filename.ctrl)
> alnsF.ctrl <- getAlignmentsWithFeatures(cmpH5.ctrl, features = "IPD")
> 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 (http://www.pacbiodevnet.com/SMRT-Analysis/Software/SMRT-Pipe), 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 s-trand 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 proper historical data or you do NOT want to use historical data, you can combine IPD in control sample and IPD of homologous positions in control sample for detection by the following code

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

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.

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)
> context.effect <- getContextEffectByPos(genomeF.historical, 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

It is OK if you detect modifications that have strong signal, such as m6A, 8-oxo-G, etc, without control sample by only using IPD hologous positions in historical data. However, it does NOT work well if you want to detect modifications that have weaker signal, such m4C, m5C, etc and you need a control sample in these cases.

If you have already have a control sample, it is better to use it no matter what modifications you want to detect. A control sample is usually helpful to increase accuray.