tagMeppr

Robin H. van der Weide

2019-09-06

Hi there! Welcome!

TagMap is a very useful method for transposon mapping (Stern 2017), enabling researchers to map the insertion sites with ease and generate long sequencing reads. However, there is little to none automatisation and downstream analysis software available for these reads. TagMeppr is an easy to use, memory efficient fastq-to-figure package written in R.

Usage

The basic usage of tagMapper revolves around three clear steps:

1. index: a tagMapper-index is made once for a specific genome and protocol (e.g. hg19 and PigyBac).
2. align: a tagMapperSample-object is made and aligned to the index
3. analyse: determine and plot highly likely integraton-sites

```
library(tagMeppr)
```

tagMepprIndex

Remember: you only have to make a tagMepprIndex once per genome and protocol! This means that you can use it for however many samples for that protocol and organism (e.g. PiggyBac in human cells) you may have.

```
library("BSgenome.Hsapiens.UCSC.hg19")

reference_hg19_PB = makeIndex(indexPath = '../premadeReferences/',
                              bsgenome = BSgenome.Hsapiens.UCSC.hg19,
                              ITR = 'PiggyBac',
                              targetInsertionSite = 'TTAA', verbose = T)
```

Loading a tagMepprIndex is easy with loadIndex().

```
reference_hg19_PB = loadIndex('BSgenome.Hsapiens.UCSC.hg19_PiggyBac_tagMeppRindex.fa.gz')
```

You can use print() on this tagMepprIndex, which gives you a lot of information:

```
print(reference_hg19_PB)
#> tagMepprIndex
#>
#>  Protocol: PiggyBac
#>
#>  Directory: .
#>  Fasta: BSgenome.Hsapiens.UCSC.hg19_PiggyBac_tagMeppRindex.fa.gz
#>
#>  Target insertion site: TTAA
#>  TIS: BSgenome.Hsapiens.UCSC.hg19_PiggyBac_tagMeppRindex.tis
#>  Number of target insertion sites: 19228699
```

tagMepprSample

The core object of tagMeppr is the `tagMepprSample`. Everything, from the paths of the fastq's to the final results, are stored in this object. To kick off the analysis, such an object should be made for every sample by using `newTagMeppr()`. In this function, the paths of the four respective fastq's, the name of the sample and the transposon are asked.

```
C91 = newTagMeppr(F1 = 'clone91_FWD_R1.fq.gz',
                  F2 = 'clone91_FWD_R2.fq.gz',
                  R1 = 'clone91_REV_R1.fq.gz',
                  R2 = 'clone91_REV_R2.fq.gz',
                  name = "clone9 (minimised)",
                  protocol = 'PiggyBac')
```

When you use the `print()` method, a summary will be given about the sample and its progression trough the tagMeppr-pipeline.

```
print(C91)
#> tagMepprSample
#>
#>  Name: clone9 (minimised)
#>  Protocol: PiggyBac
#>  Primer checked: no (highly recommended!!!)
#>  Aligned: FALSE
#>  Analysed: FALSE
```

primer-checks

After this, it is a good idea to check whether your primer-design is one that tagMeppr expects: reverse- and forward-primers on different ITRs. Also, it checks if the reverse is found near the start of the transposon. This is to be able to accurately denote the orientation of the transposon. Normally, we expect the reverse-primer to be found at the start of the ITR-sequence. When running `checkPrimer()`, it will automatically update the tagMepprSample-object with the `rev5_fwd3` tag. If it appears to be to other way around, the sample-object will have a `rev5_fwd3=T` flag.

It is highly recommended to do this!

```
fwdPrimer = "CGTCAATTTTACGCAGACTATC"
revPrimer = "GTACGTCACAATATGATTATCTTTCTAG"

checkPrimer(fwdPrimer = fwdPrimer,
            revPrimer = revPrimer,
```

```
            exp = C91,
            ITR = 'PiggyBac')
```

```
print(C91)
#> tagMepprSample
#>
#>  Name: clone9 (minimised)
#>  Protocol: PiggyBac
#>  Primer checked: yes (flipped)
#>  Aligned: FALSE
#>  Analysed: FALSE
```

Align

To align the tagMeppr-sample to the index, use the `align()`-function. This will call `bwa mem` and run it with optimal parameters set. If `bwa` or `samtools` are not installed, the function will can an error. Afterwards, the chimeric reads will be filtered and placed in a `GRanges` object for easy usage later on. Running `align()` will automatically update the `tagMeppr-sample` to include the parsed reads.

```
align(exp = C91,
      ref = reference_hg19_PB,
      cores = 30)
```

The object will be updated, which keeps your evironment tidy. The summary will now also show the number of informative reads (i.e. a chimeric read to both an ITR and the reference) and the temporary folder of the .BAM-files.

```
print(C91)
#> tagMepprSample
#>
#>  Name: clone9 (minimised)
#>  Protocol: PiggyBac
#>  Primer checked: yes (flipped)
#>  Alignment-folder: /tmp/WBQRC9002Z
#>  Informative FWD-reads: 402
#>  Informative REV-reads: 712
#>  Analysed: FALSE
```

PCR-duplicates

By default, `align()` will remove suspected PCR-duplicates. This is not possible with normal tools like `samtools` due to the chimeric nature of the reads. If one doesn't want this deduplication to happen, set `dedup=F` in `align()`.

```
C91_noDedup = C91
align(exp = C91_noDedup,
      ref = reference_hg19_PB,
      dedup = F,
      cores = 30)
```

The output of the `print()`-command will now show that deduplication didn't happen.

```
print(C91_noDedup)
#> tagMepprSample
```

```
#>
#>  Name: clone9 (minimised)
#>  Protocol: PiggyBac
#>  Primer checked: yes (flipped)
#>  Alignment-folder: /tmp/TONFL1410Q
#>  Deduplicated: FALSE
#>  Informative FWD-reads: 5189
#>  Informative REV-reads: 30250
#>  Analysed: FALSE
```

Analysis

Find insertions

The `findInsertions()` function will first find all reads that overlap a TIS, which in the case of PiggyBac will be "TTAA". Next it will calculate whether there is a bias towards one side of the TIS using a binominal test. The bias, denoted as D, -1 when all reads are upstream and $+1$ when all reads are downstream of the TIS.This is done independently for the forward and reverse reads:

$$p_{fwd/rev} = \binom{reads_{D<0}}{reads}$$

Next, we filter out TISs which have the bias on the same side of the TIS:

$$sgn(D_{fwd}) \neq sgn(D_{rev})$$

To calculate a "TIS-specific" p-value, we use Edgington's sum-p method, which is very conservative in our usage. This ensures that, when $p_{combined} < \alpha$, both the fwd and the rev reads are indeed biased.

$$p_{combined} = \frac{(\sum_{i=1}^{2} p_i)^2}{2!}$$

Afterwards, a holm-correction is done to limit the Family-Wise Error Rate (FWER).

```
findInsertions(exp = C91, ref = reference_hg19_PB, padding = 2)
```

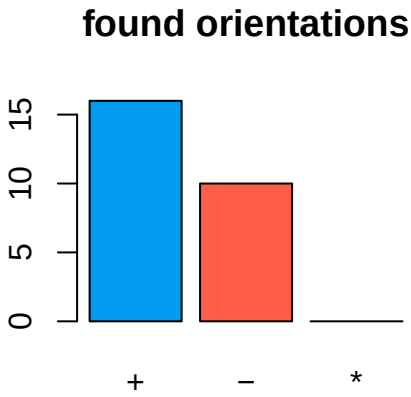
```
print(C91)
#> tagMepprSample
#>
#>  Name: clone9 (minimised)
#>  Protocol: PiggyBac
#>  Primer checked: yes (flipped)
#>  Alignment-folder: /tmp/WBQRC9002Z
#>  Informative FWD-reads: 402
#>  Informative REV-reads: 712
#>  Unique TISs covered: 26
#>      p<0.05: 26
```

```
foundInsertions = results( C91 )

head(foundInsertions)
#>   seqnames     start       end strand fwdCount revCount fwdD  revD
```

```
#> 1     chr1  36377340  36377343      +       15       24   -1  1.00
#> 2     chr1 225144936 225144939      +       18       17   -1  0.89
#> 3     chr3  75442959  75442962      +        6       23   -1  1.00
#> 4     chr3 129776748 129776751      +       11       48   -1  1.00
#> 5     chr3 194542188 194542191      +        9       25   -1  1.00
#> 6     chr3 195352648 195352651      -       21       64    1 -1.00
#>           padj
#> 1 4.300835e-08
#> 2 2.328306e-07
#> 3 2.929688e-03
#> 4 8.583069e-06
#> 5 9.155553e-05
#> 6 1.182343e-11
```

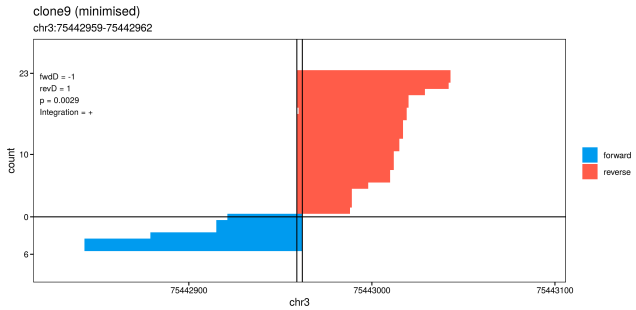
```
barplot(table(foundInsertions$strand), horiz = F,
        col = tagMepprCol(),
        main = 'found orientations')
```



Plot

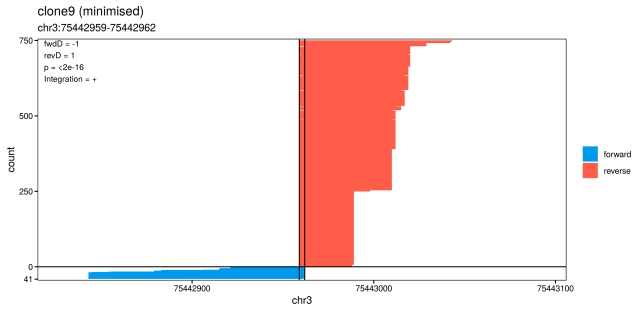
Single insertions

```
plotSite(C91,site = 3)
```



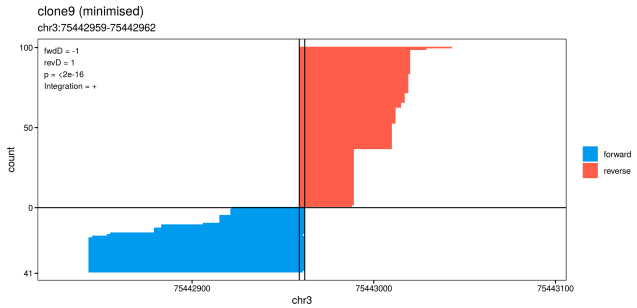
Returning to the sample witout deduplication, one can now see what would happen:

```
plotSite(C91_noDedup,site = 3)
```



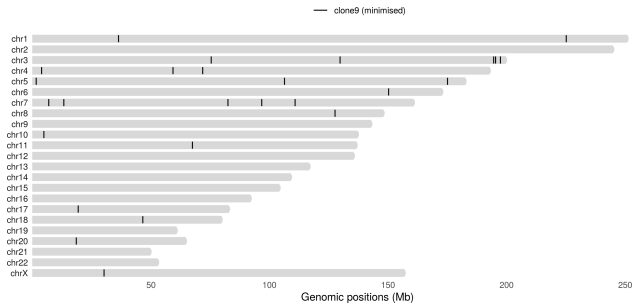
You can also set a limit on the maximum number of reads shown:

```
plotSite(C91_noDedup,site = 3, maxReads = 100)
```



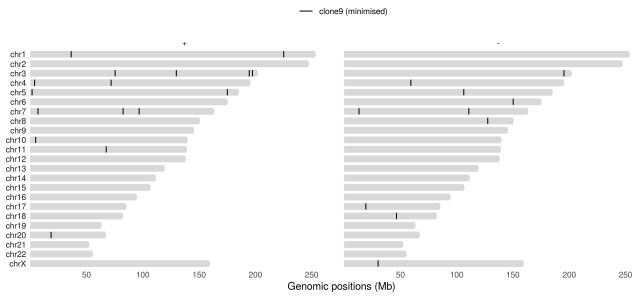
All insertions

```
plotInsertions(exp = C91)
```



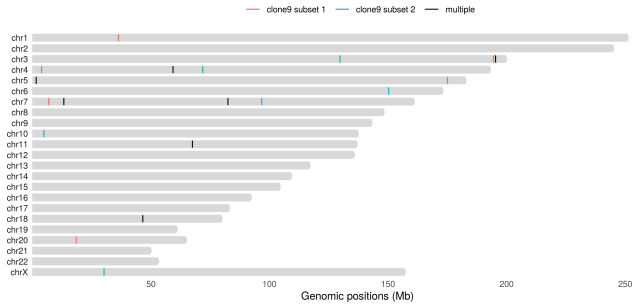
show orientation:

```
plotInsertions(exp = C91, showOrientation = T)
```

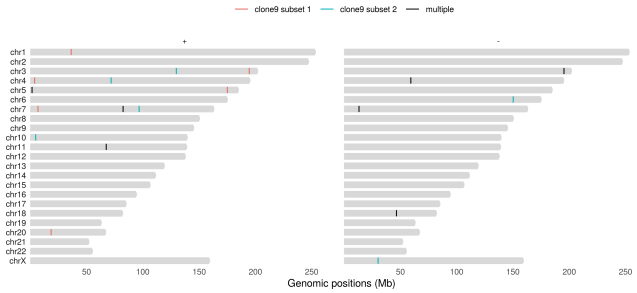


Plot muliple samples

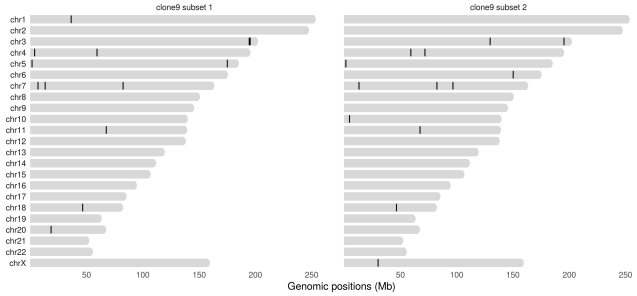
```
plotInsertions(tagMepprSampleList)
```



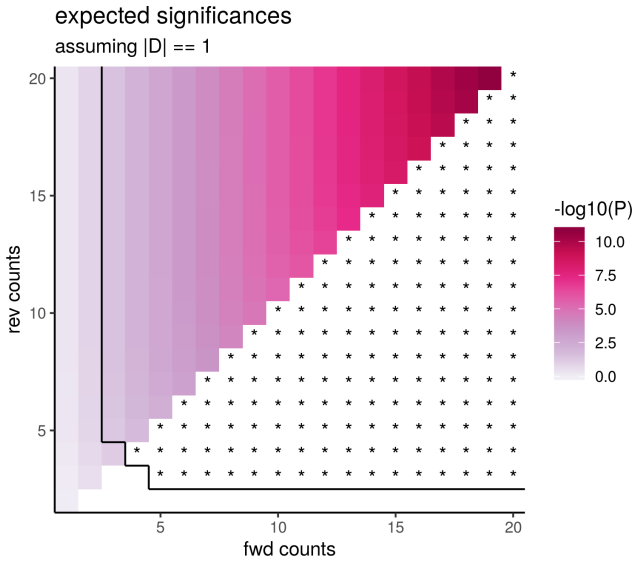

```
plotInsertions(tagMepprSampleList, showOrientation = T)
```




```
plotInsertions(tagMepprSampleList, sideBySide = T)
```



significance



Biography

Stern, David L. 2017. "Tagmentation-Based Mapping (Tagmap) of Mobile Dna Genomic Insertion Sites." *bioRxiv*. Cold Spring Harbor Laboratory. https://doi.org/10.1101/037762.