pbcore_BAM_toy.md

Using the opener.py gateway to access PacBio BAM files

Objectives

- explore some of the functionality of pbcore
- point out that finding certain pieces of information can be difficult because it's not obvious it's there
- demonstrate mismatch between IPD list depending on how it's accessed. Ask Dave why

```
from pbcore.io import opener

# open up the dummy aligned_subreads.bam file
# using the indexing in from aligned_subreads.bam.pbi
a = opener.openIndexedAlignmentFile( '~/Projects/PacBio/test_pbcore/m140905_042212_sidney_c1005
```

```
# take a look at the first alignment, just as an example
aln = a[0]

# step up the chain to check out what's inside the pysam object
p = aln.peer

# check out the custom goodies that are attached w/ pacbio alignments
aln_tags = dict( p.tags )
print aln_tags.keys()
```

```
['zm', 'qs', 'NM', 'iq', 'ip', 'sq', 'cx', 'rq', 'AS', 'RG', 'mq', 'sn', 'np', 'qe', 'dt', 'dq'
```

These two-letter tags represent different pieces of information we can access. For example, 'zm' points to ZMW holenumber, 'ip' points to IPD, and 'NM'. A lot of these I don't know what they are...

We can get a list of the first 20 IPDs with:

```
print aln_tags['ip'][0:20]
```

```
[10, 5, 1, 9, 10, 5, 0, 0, 6, 0, 11, 29, 2, 37, 28, 74, 19, 3, 4, 2]
```

Useful aside about getting a full glimpse inside alignment object instances

The tags are where specific information such as IPDs are stored. For many things, you don't actually need to go back to the pySAM alignment-object instance to access specific information; the pbcore api provides a lot of this functionality.

However, some of pbcore functionality is currently hard to find. For example, if you take an alignment object instance, aln (see line 7 in the first coding cell), and try to see what's inside, you don't see that you can get the IPD.

```
# use dir introspection to see what's inside
print dir( aln )
```

```
['_unrolledCigar', 'aEnd', 'aEnd', 'aStart', 'aStart', 'bam', 'contextFlag', 'holeNumber', 'isR
```

There's no IPD in that list, but if you look at the source you can see that there's a couple builtin ways to do it: 1. Through a pulseFeature function 2. Through an IPD alias accessor

```
# 1.
print 'Using pulseFeature() to get IPDs: ' + str( aln.pulseFeature( 'Ipd', aligned=False )[0:10

# 2.
print 'Using IPD() to get IPDs: ' + str( aln.IPD( aligned=False )[0:10] )
```

```
Using pulseFeature() to get IPDs: [ 6  0 11 29  2 37 28 84 19  3]
Using IPD() to get IPDs: [ 6  0 11 29  2 37 28 84 19  3]
```

The IPDs are in pbcore!

Sidenote: why does the pySAM-accessed list of IPDs start earlier than the pbcore-accessed list? Also, what's the deal with the pySAM-accessed list having an IPD of 74 frames when the pbcore-accessed list has an IPD of 84 frames?