DNA-Modification Detection with SMRT-Sequencing using R

Pacific Biosciences

October 20, 2011

Contents

Abstract

This documents describes a programing interface to Pacific Biosciences cmp.h5 files. These files provide additional data beyond basecalls and quality values obtained during a sequencing run. These data can be used to discover base modifications, e.g., DNA methylation events. In addition to demonstrating the detection of DNA methylation in a two-sample statistical testing context, we present an `R` API for generally extracting data from PacBio HDF5 files.

1 Introduction

Base modifications are important in understanding a variety of biological processes such as gene expression, host-pathogen interactions, DNA damage and DNA repair. Single-Molecule Real-Time (SMRT) sequencing has the potential to revolutionize the study of base modifications through direct detection on unamplified source material. Traditionally, it has been a challenge to study the wide variety of modifications that are seen in nature. Most high throughput techniques focus on cytosine methylation – made accessible through bisulfite treatment when sequencing or PCR techniques are

used to detect the methylation at a single-base resolution. SMRT-sequencing, in contrast, does not require genetic alterations to the source material in order to view base modifications. Instead, measurements of the kinetics of base addition are made during the normal course of sequencing. These kinetic measurements present characteristic patterns in response to a wide variety of base modifications. As a result of this relatively simple mechanism to detect base modifications, it is now possible to study more than just 5-methylcytosine in a high-throughput fashion. Bacterial modifications such as 6-methyladenine, or more recently re-discovered modifications such as 5-hydroxymethylcytosine, are accessible to study using a single sequencing method – and even a single sequencing run – on the PacBio RS. As the technology advances, direct detection of RNA modifications will also become possible. As our understanding of kinetic information grows, the analysis of base modifications using SMRT technology will continue to become easier and faster, making accessible a rich new frontier of scientific study.

In this document we attempt to demonstrate how to perform DNA modification detection using the suite of R packages developed and used at Pacific Biosciences. These APIs provide the developer with low-level access to all information collected during a sequencing run. This document attempts to serve two purposes (1) to demonstrate the use of the pbh5 R package to access low-level data produced during a SMRT-sequencing run and (2) to provide a starting point for users to conduct their own kinetic analysis.

1.1 R Packages/System Requirements

In this analysis we will make heavey use of the pbh5 and pbutils R packages. In addition, the pbh5 package depends on the h5r package. Finally, we will also make use of the ggplot2 and xtable packages available on CRAN. All of the analysis conducted here can be performed using the pbh5 package exclusively, however, the code to execute this document depends on the afformentioned packages.

```
> require(pbh5)
> require(pbls)
> require(xtable)
> require(ggplot2)
> source("utils.R")
```

In addition to R package requirements, this document requires a system with approximately 3-4 Gigabytes of memory and a recent version of R, i.e., R-2-12.

Finally, this document is a "vignette", i.e., the code and text is all contained in analysis.Rnw, the code extracted can be found in analysis.R. To "run" this document, the user can perform the following from the top-level directory:

```
make analysis-build
```

This will download the data into the Data directory and then run the analysis.Rnw document.

1.2 Experimental Setup

This experiment focuses on two different sources of input DNA (1) Synthetically methylated DNA with a few site-specific modifications per template and (2) DNA library data from lambda phage. For the synthetically modified data, we have a 5 identical (from a nucleotide sequence perspective) templates. Four of these templates have modifications at particular sites. One is a control template

	nAlignments	nMolecules	nMovies	nReferences
2x_4mC	104411	24872	1	1
2x_5hmC	33332	8752	1	1
2x_5mC	38627	9598	1	1
2x_6mA	77494	20820	1	1
control	37874	9323	1	1

Table 1: Summary of synthetically methylated datasets used in this document.

which will be used in comparison to each treatment template. For the lambda data set, we have both DAM+ and DAM- preparations as well as whole-genome amplifications for both of the DAM+ and DAM- preparations. The DAM or DNA methyl-transferase specifically methylates the adenine basee of the GATC motif in lambda. Additionally, lambda contains methyl-transferases for othe motifs, specifically the XXXX motif - this methylation occurs a lower frequency (XXX: Citations).

	nAlignments	nMolecules	nMovies	nReferences
6mA_dam-_native	149408	35829	2	1
6mA_dam+_native	146304	32134	2	1
6mA_dam-_WGA	160183	37588	2	1
6mA_dam+_WGA	97840	24469	2	1

Table 2: Summary of lambda datasets used in this document.

2 Exploring the Data

As described above, SMRT-Sequecing provides a rich set of information beyond that of traditional sequencing platforms. Specifically, here we focus on information about the kinetic behavior of the polymerase at specific positions in the reference sequence. We first examine high-level summaries of the data, such as yield, read length, and accuracy. We will focus on the Lambda data for some of the major exploratory work because it provides a larger number of sequencing context to investigate. First, we describe some of the major components of the R API which we will use throughout this document to analyze the two different modification datasets.

2.1 Working with the Compare H5 File

The *cmp.h5* file (pronounced comp H5 or compare H5) provides a rich set of data resulting from the alignment of PacBio data to a reference sequence. The cmp.h5 file may contain one or more movies (sequencing runs) and contains all of the alignments for that movie's reads to a reference fasta file.

```
> cmpH5 <- PacBioCmpH5("../Data/Lambda/6mA_dam+_native/data/aligned_reads.cmp.h5")
> cmpH5

class of: PacBioCmpH5
file: /home/UNIXHOME/jbullard/projects/analyses/11-08-04-KineticsDevNet/clean/Data/Lambda/6m
```

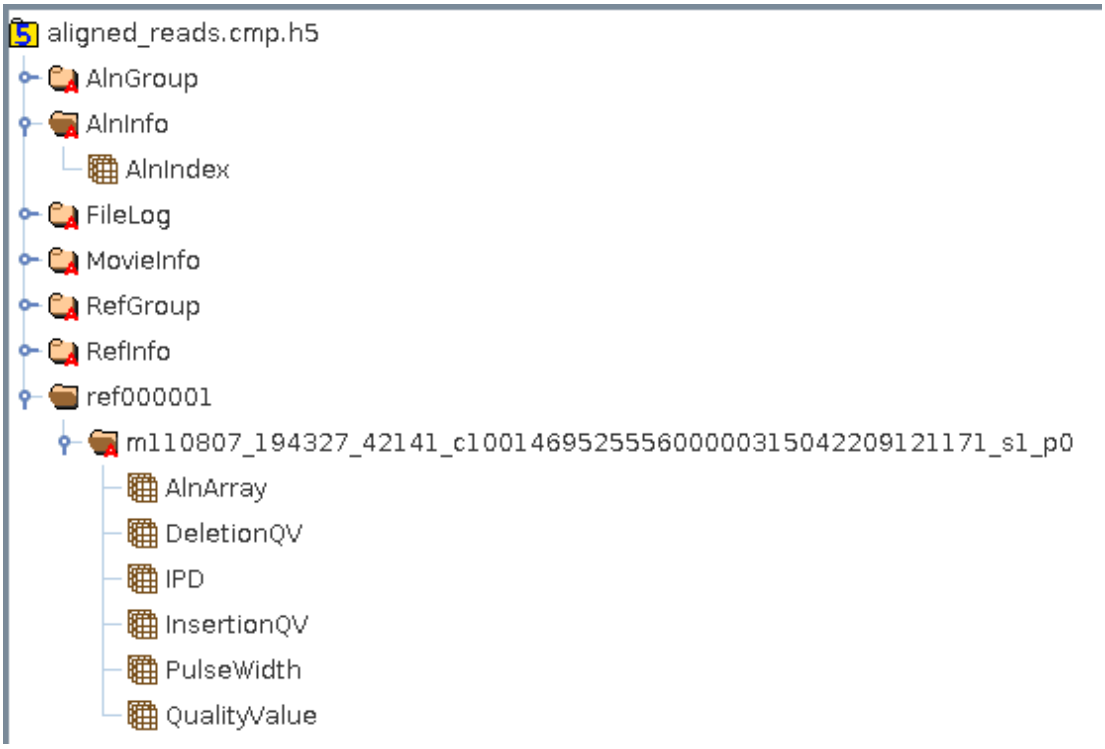


Figure 1: *cmp.h5 Structure* - A screenshot of the cmp.h5 file structure as seen through the "hdfview" tool provided by the hdfgroup.org. PacBio cmp.h5 files provide a wealth of additional information about the sequencing run. Broadly speaking, HDF5 files can be thought about as a file system for your data – allowing one to organize both metada and experimental results in consistent structure.

```
Version: 1.2.0.SF
N Alignments: 146304
N ReadGroups: 2
N RefSeqs: 1
```

The core of the file is represented by the "AlnIndex" and the corresponding "AlnArray" datasets. Alignments, QualityValues, and kinetic data concerning are stored at the "refGroup/alnGroup" level of the hierarchy, e.g.,

```
> group <- "ref000001/m110818_122604_42141_c100129202555500000315043109121114_s1_p0"
> g <- getH5Group(cmpH5, group)
> ls(g)

[1] "."            "AlnArray"     "DeletionQV"   "IPD"          "InsertionQV"
[6] "PulseWidth"   "QualityValue"
```

Each of these datasets contain all of the alignment related data for a given "alignment group" which tends to be a movie. The alignments are packed together in a compact format and the "AlnIndex" contains the relevant information on how to extract a particular alignment. The incredibly long "basename" portion of the path represents the movie name, whereas the "dirname" portion represents the reference sequence. The mapping between the canonical "ref000001" and the name in the fasta file can be found using the refGroup function. For our work, the most pertinent datasets are: AlnArray, IPD (inter-pulse duration), and PulseWidth.

As mentioned above, all of the alignments in the file are stored in a global alignment index. In addition to providing summary statisitics about alignments, e.g., the number of mismatches, deletions, etc., the index provides the offsets into the alignment datasets for fast random access to alignments.

```
> head(alnIndex(cmpH5), 2)
```

```
      ID
1 109922
2   8666
                                                                  alnGroupPath
1 /ref000001/m110818_122604_42141_c100129202555500000315043109121114_s1_p0
2 /ref000001/m110818_122604_42141_c100129202555500000315043109121114_s2_p0
                                                        movieName   refName
1 m110818_122604_42141_c100129202555500000315043109121114_s1_p0 ref000001
2 m110818_122604_42141_c100129202555500000315043109121114_s2_p0 ref000001
     fullRefName tStart tEnd alignedStrand holeNumber setNumber strobeNumber
1 lambda_NEB3011      1   98             1      47143         1            0
2 lambda_NEB3011      1  108             1      77027         2            0
  moleculeID rStart rEnd mapQV nMatches nMisMatches nInsertions nDeletions
1     367143     10  104     0       88           0           7         10
2     717027    320  435   254      104           0          12          4
  offsetBegin offsetEnd nBackRead nOverlap
1    11150415  11150519         0        0
2     2965416   2965535         1        1
```

In general, these details can be ignored and users can interact with the file via the accessor API functions. For instance, to access an alignment:

```
> alns <- getAlignments(cmpH5, idx = c(1, 200, 3))
> lapply(alns, head, n = 2)
```

```
[[1]]
     read reference
[1,] "T"  "T"
[2,] "A"  "A"

[[2]]
     read reference
[1,] "G"  "G"
[2,] "G"  "G"

[[3]]
     read reference
[1,] "G"  "G"
[2,] "G"  "G"
```

The API provides a large set of functions with the above signature, i.e., `cmpH5` and `idx`, where `idx` is an index vector which must contain values between 1 and `nrow(cmpH5)` inclusively. These naturally

refer to the rows in the `alnIndex(cmpH5)`. In addition to `getAlignments` other useful functions include: `getIPD`, `getPulseWidth`, and `getQualityValue`.

To get more information on the cmp.h5 file format refer to: www.pacbiodevnet.com/Learn/Documentation. Also, to get help on the pbh5 package, try `?pbh5`. In the remainder of the document, we will typically hide the code to not disrupt the document flow.

2.2 Visualizing Kinetic Properties of the System

In this section, we visualize pulse width and IPD (inter-pulse duration) distributions. These two, especially IPD, represent the primary source of data informing possible base modifications. In Figures 2 and 3, we have plotted a schematic of the trace signal. In this figure all pulses have the same magnitude for simplicity, and indeed we will focus about the distributions of durations for incorporation (pulse width) and translocation (IPD). In the aforementioned figures, we have drawn a red arrow to indicate the IPD at a position of interest. Each read covering a region gives us information about the incorporation events. We can compare that to a control sample where we are certain there are no modifications.

We want to examine the various sources of variation in the IPD and pulse width distributions. In our case, we will compare a function of the IPD distribution in a treatment sample (where we believe there to be modifications) to that of a control sample (where we have removed them) and therefore we can use distribution free tests. However, in general we might wish to investigate whether it is possible to determine a modification without a control sample and in general by knowing nuisance sources of variation we can decrease the size of our sample and still have as much power to reject the null.

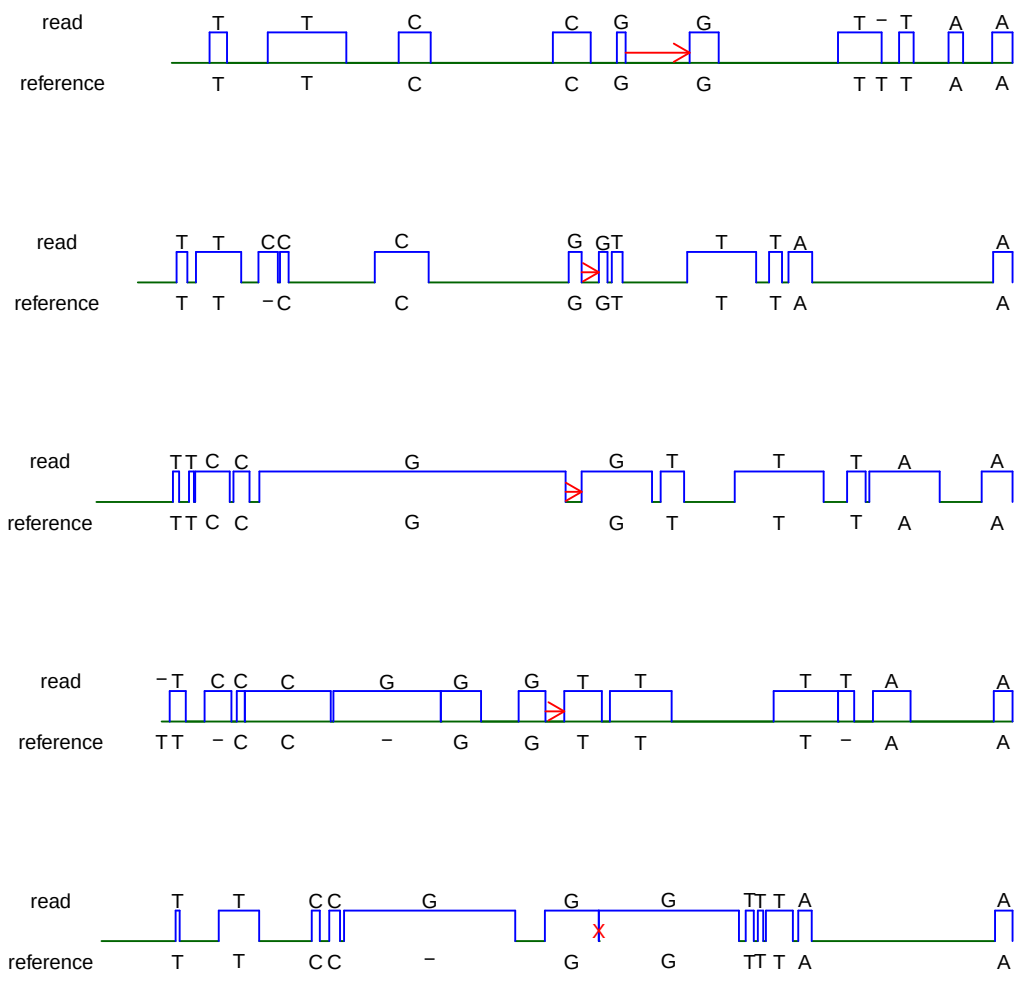


Figure 2: *Trace Cartoon* - Here we plot "trace" views directly from the cmp.h5 file. When we allow the possibility of insertions/deletions/mismatches it is more complicated to ensure that you are viewing a proper incorporation event.

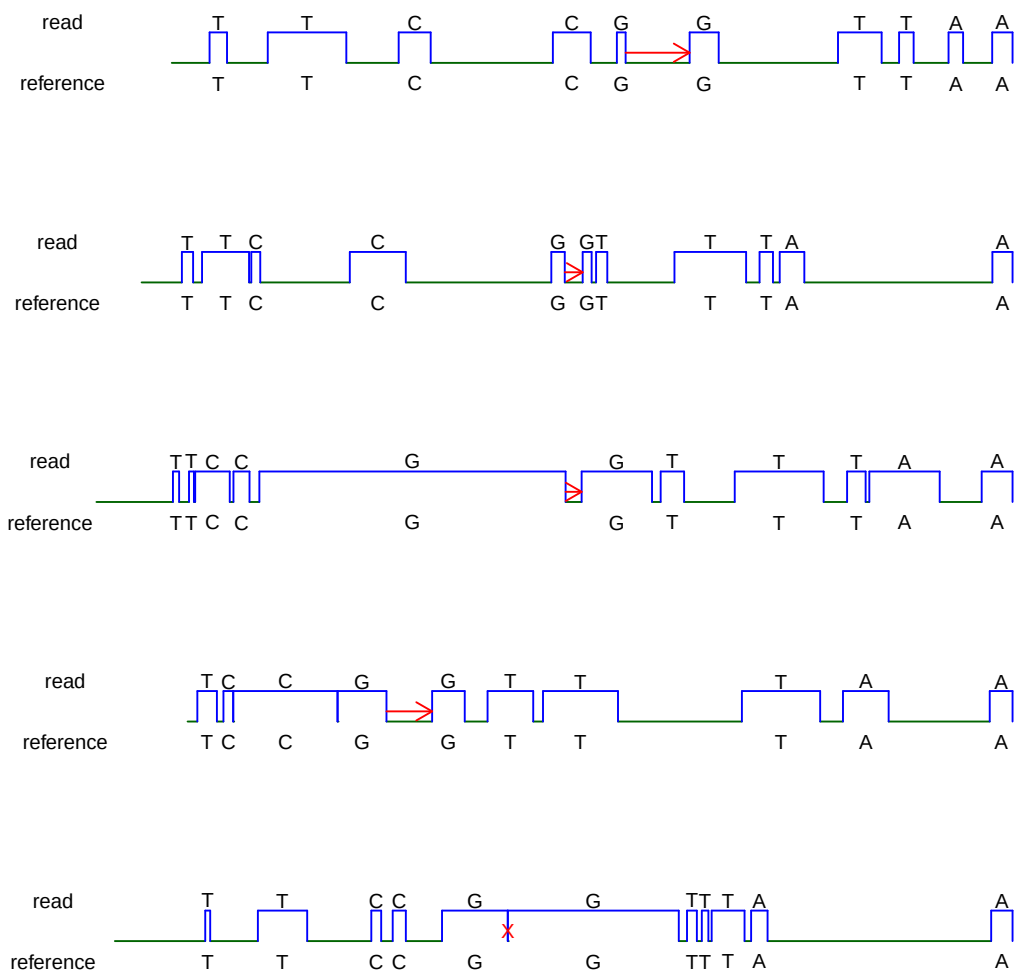
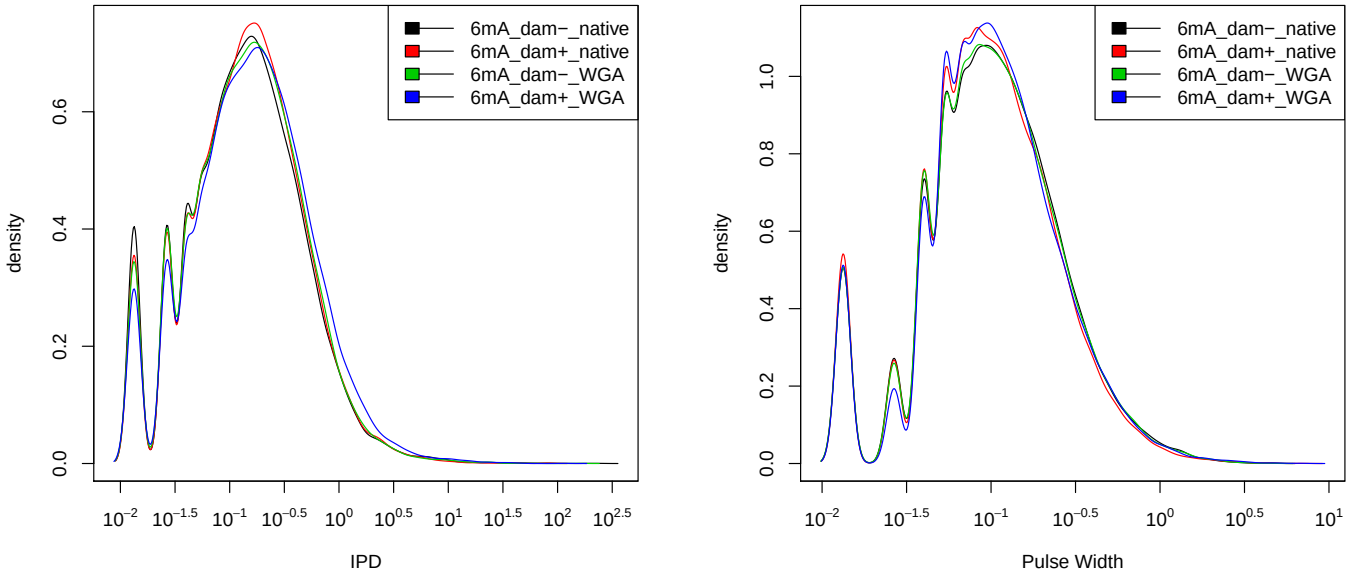
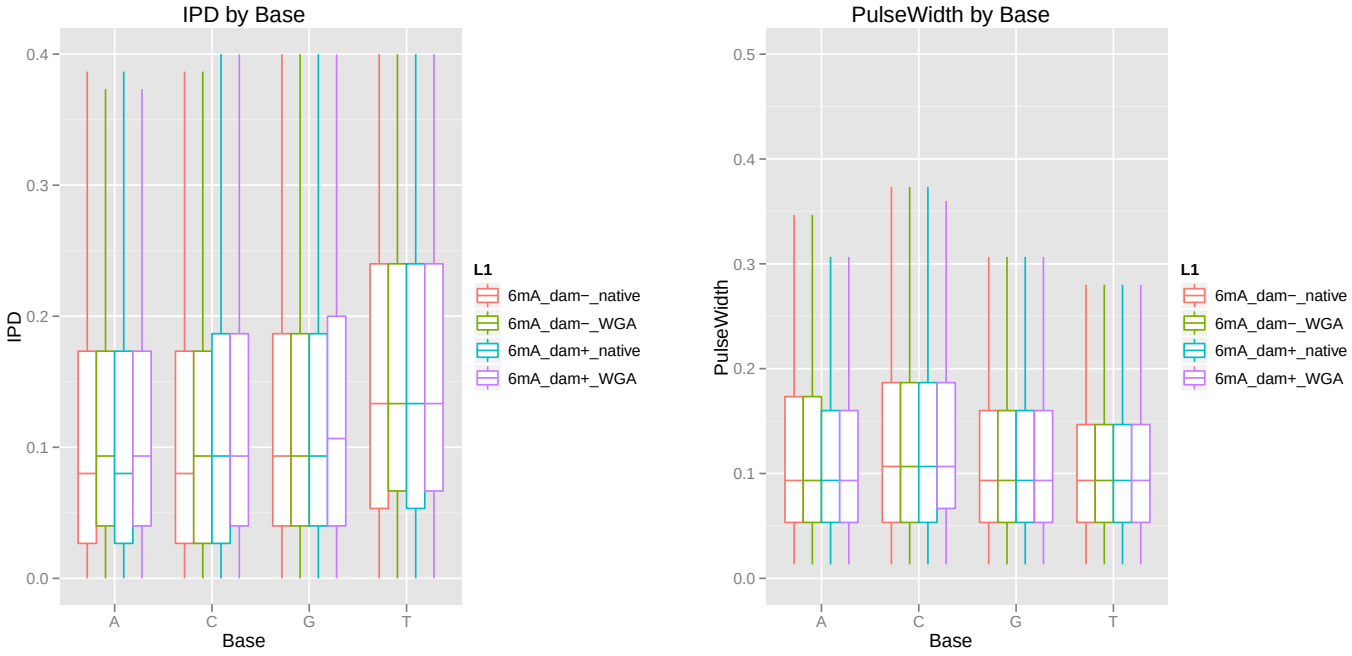


Figure 3: *Trace Cartoon* - Here we plot "trace" views directly from the cmp.h5 file. This is the same plot as above, however we are restricting things to those bases which match. In practice, such a filtering will be too stringent.



(a) (b)

Figure 4: *Global IPD and PulseWidth Densities* - Here we plot the global IPD and PulseWidth distributions. This distribution is a mix of incorporations of the 4 nucleotides. Both IPD and PulseWidth are currently stored in the file in seconds rather than frames. The spikes occur because in fact the distribution is not continuous.



(a) (b)

Figure 5: *IPD and PulseWidth By Base* - Here we plot the IPD and PulseWidth distributions stratified by the base being incorporated. We see that there is a base effect on IPD, i.e., which base we are incorporating changes the kinetic behavior of the enzyme.

The getByTemplatePosition function retrieves data for `idx` reads. It takes a function, f, which returns a list of vectors or matrices where the length or number of rows is equal to the alignment length for alignment i. Typically, one just passes in an existing function, such as getIPD or getPulseWidth.

```
> head(getByTemplatePosition(cmpH5, idx = 1:2, f = getIPD))
```

```
  position read ref idx strand        elt
1        1    C   C   1      1 0.17333291
2        2    C   C   1      1 0.06666651
3        3    C   C   1      1 0.17333291
4        4    G   G   1      1 0.22666611
5        5    C   C   1      1 0.13333301
6        6    C   C   1      1 0.39999902
```

Additionally, there are a number of high-level data access functions related to retrieving the information in the cmp.h5 file by position and context. As mentioned before, these functions take a vector of indices which refer to the reads in the alignment index to be used, e.g.,

```
> head(makeContextDataTable(cmpH5, idx = 1:2, up = 2, down = 2))
```

```
    elt.ipd    elt.pw elt.tpos context.P01 context.P02 context.P03 context.P04
1 0.3999990 0.0533332       96           T           A           T           T
2 0.4266656 0.1066664       95           A           T           T           T
```

```
3 0.1066664 0.0799998       94          T          T          T          T
4 0.7866647 0.0799998       93          T          T          T          A
5 0.0399999 0.0533332       92          T          T          A          A
6 0.0399999 0.1466663       91          T          A          A          A
  context.P05
1           T
2           T
3           A
4           A
5           A
6           T
```

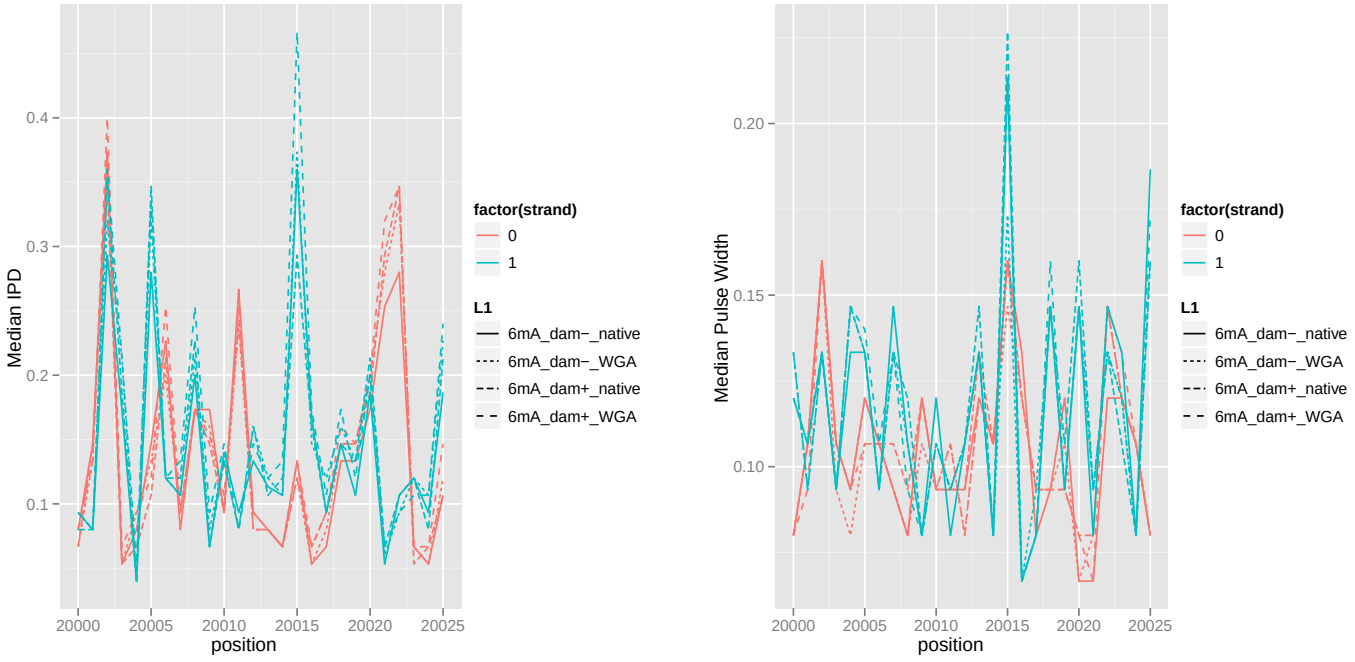
Another useful function for summarizing things by context is:

```
> s <- summarizeByContext(cmpH5, idx = 1:100, up = 1, down = 1,
+     statF = getPulseWidth)
> head(s)

    count     value
AAA  1486 0.1066664
AAC   520 0.0933331
AAG   465 0.0799998
AAT   353 0.0933331
ACA   277 0.1199997
ACC   264 0.1266664
```

Throughout this document we will be using these two or three functions for data access. Below we define a convenience function which takes a range along the genome and then retrieves the results of f for those reads. An important point to notice is that the getReadsInRange function returns any read that overlaps either the start or the end of the range. Therefore, portions of reads will not be within $[s, e]$, hence the subset below.

```
> getByPositionAndStrand <- function(f = getIPD, s = 20000, e = 20025) {
+     pbutils::collapse(lapply(cmpH5s, function(cmpH5) {
+         x <- getByTemplatePosition(cmpH5, idx = getReadsInRange(cmpH5,
+             1, s, e), f = f)
+         x <- subset(x, position >= s & position <= e)
+         ddply(x, c("strand", "position"), function(a) {
+             median(a$elt, na.rm = T)
+         })
+     }))
+ }
> byPositionAndStrandIPD <- getByPositionAndStrand()
> byPositionAndStrandPW <- getByPositionAndStrand(f = getPulseWidth)
```



(a) (b)

Figure 6: *IPD and PulseWidth by Strand and Position* - Here we plot the median of the IPD distribution conditioned on both strand and position. We can see the presence of a very strong position and strand effect.

2.3 Context-specific Efffects

Finally, we investigate the effect of sequence context on IPD and pulse width distributions. Alignment-level data from a cmp.h5 file is always stored with respect to the bases being incorporated. Therefore, when one retrieves an alignment from the cmp.h5 file, if that alignment is labeled as a reverse strand alignment: `getTemplateStrand(cmpH5) == 1`, then the reference sequence is reverse complemented rather than the read. The importance of this representation is that we always store the data (e.g., alignments, IPDs, pulse widths, etc.) in the direction in which the bases are incorporated.

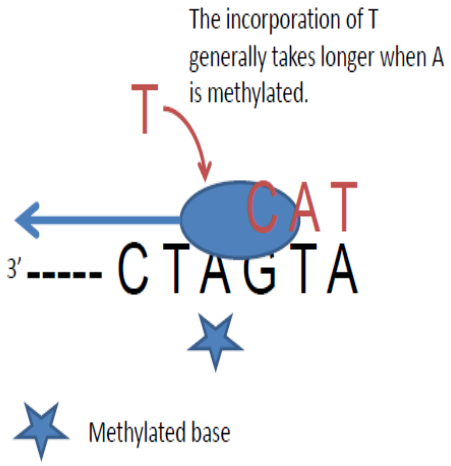


Figure 7: *GATC Cartoon* - A cartoon depicting the incorporation of "T" which will be "delayed" when the complement A base is methylated. The "T" base is what is stored in the "AlnArray" data structures.

```
> getTemplateStrand(cmpH5)[1:10]

 [1] 1 1 0 0 1 1 0 1 0 1
```

```
> tmp <- getByTemplatePosition(cmpH5, idx = 1:2)
> head(tmp[order(tmp$position, tmp$strand), ])
```

```
    position read ref idx strand        elt
1          1    C   C   1      1 0.17333291
106        1    C   C   2      1 0.22666611
2          2    C   C   1      1 0.06666651
107        2    C   C   2      1 0.29333261
3          3    C   C   1      1 0.17333291
108        3    C   C   2      1 0.17333291
```

We can use the associateWithContext to get a data element by context. There are a couple of relevant options to consider. First, context can either be determined by the read bases or by the reference bases. In either case, gaps are removed from either the read or the reference and then context is computed.

```
> tmp <- associateWithContext(cmpH5, idx = 1:2, f = getTemplatePosition,
+     collapse = T, useReference = T)
> head(tmp[order(tmp$elt), ])
```

```
    elt context
94    3   CGCCC
198   3   CGCCC
93    4   CCGCC
197   4   CCGCC
92    5   GCCGC
196   5   GCCGC
```

Here we used the reference context to group the results of the function call f and you can see that there are two different contexts for the same position – this occurs because we still maintain the orientation of the alignments in terms of read space, so for the reference context of 'GGGCG' there are the set of reverse strand reads with the context 'CGCCC'.

```
> contextTable <- associateWithContext(cmpH5, idx = sample(1:nrow(cmpH5),
+     size = 1000), f = getIPD, collapse = T, useReference = T,
+     up = 1, down = 1)
> par(cex.axis = 0.65)
> boxplot(split(contextTable$elt, contextTable$context), ylim = c(0,
+     0.5), las = 2, main = "Context-specific IPD distributions",
+     ylab = "IPD", outline = FALSE, col = rep(1:4, each = 4))
```

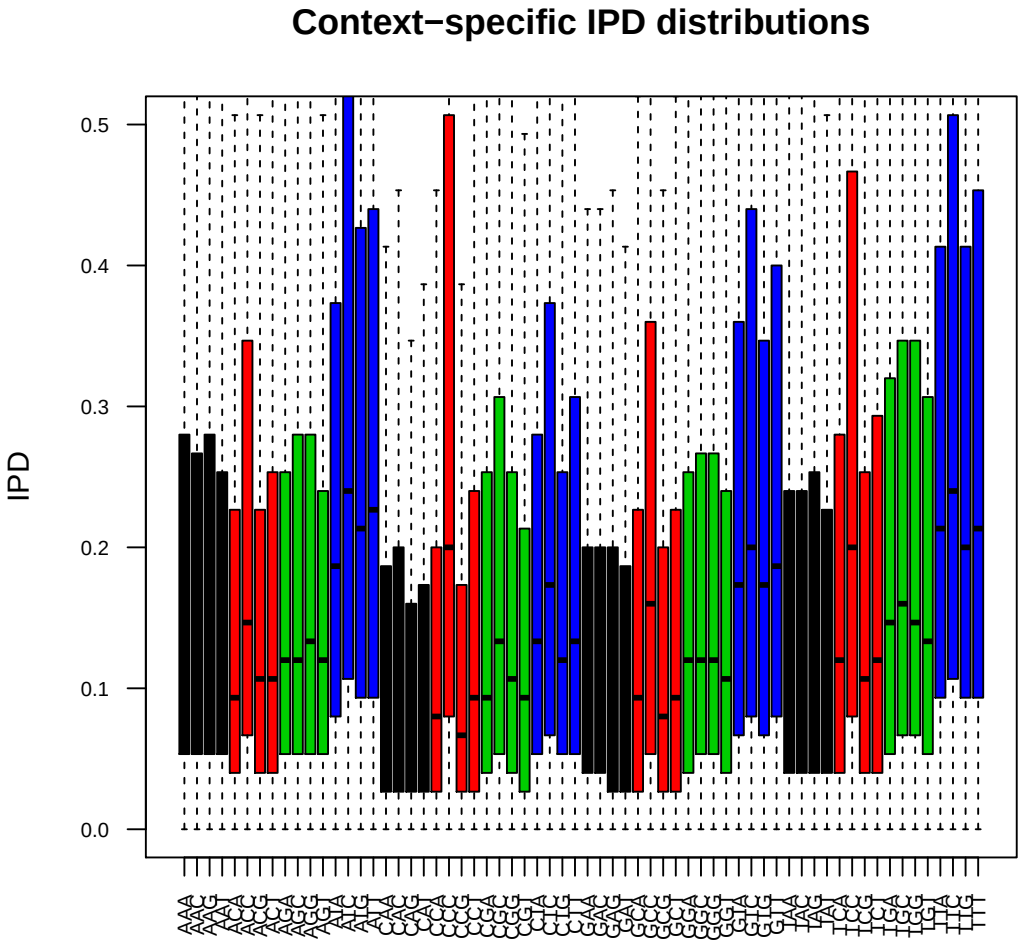


Figure 8: *IPD by Context* - Plots of IPD by context. We can see that the IPD distribution depends on context. Here the boxplots have been colored by the base being incorporated.

We can use the associateWithContext to see modification patterns which might follow motifs, rather than specific positions. In this case, we focus on the DAM+ condition of the Lambda dataset as GATC

motif is mostly modified.

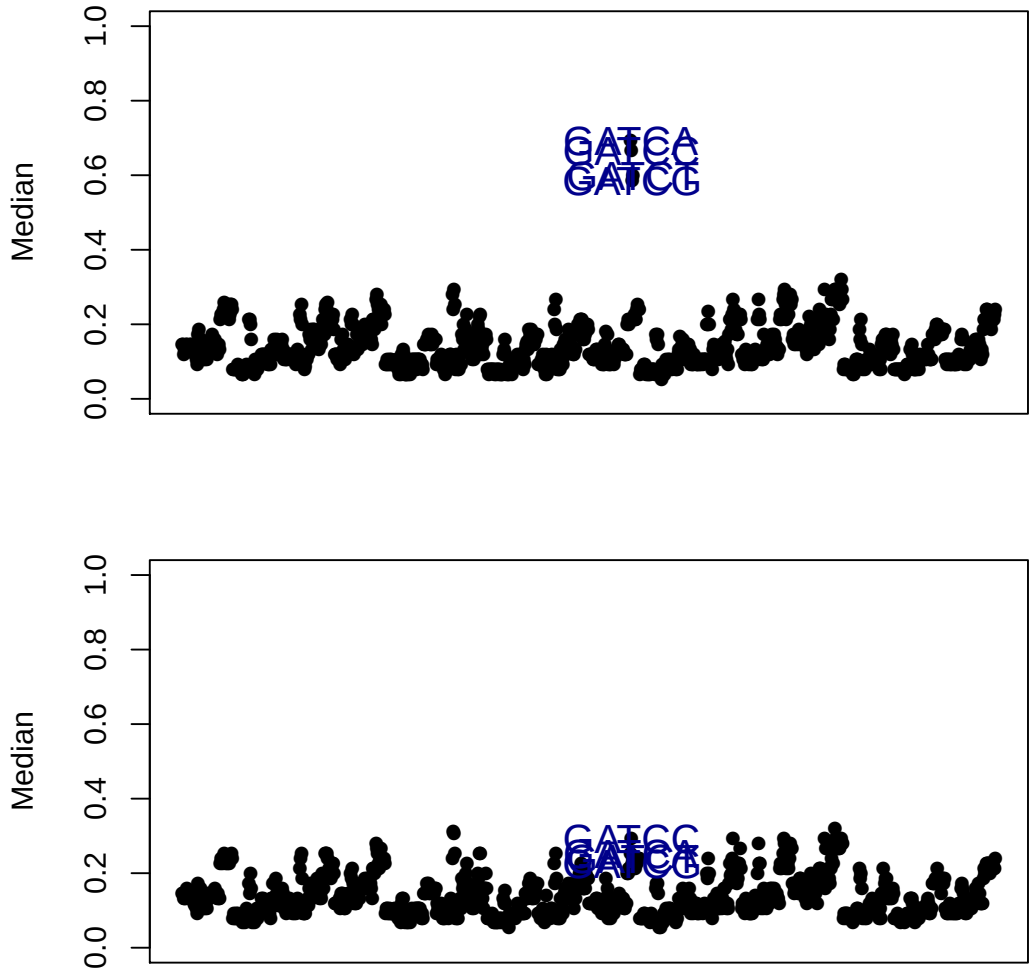


Figure 9: *Context-specific Modificationse* – Here we plot the median IPD for 5 based contexts for both the DAM+ and DAM- Lambda strains. First, the range of IPDs is quite similar for all non-modified motifs, i.e., the motif effect is larger than the the strain effect.

3 Statistical Testing

In this section we focus on two-sample statistical tests comparing the IPD distribution in a control sample to a treatment. Each particular DNA modification has a different signature at or around the modified base and more sophisticated methods will take that into account. In this section, we will first focus on the Synthetic data sets where the modified positions are known. Here, we will look at detection as a function of coverage. In general, with sufficient coverage the difference between IPD distributions can be detected, however, certain modifications do not have a large effect on the kinetics of the polymerase and therefore to detect these smaller effects we need to observe the incorporation event many times. Additionally, the effects of a modified base might occur around the actual modifications.

We can view this as a simple two-sample statistical testing problem where IPD measurements obtained from our native sample are compared to IPD measurements obtained from an unmodified sample. As in many high-throughput sequencing experiments, some of the canonical assumptions, e.g., independence, normality, etc. might not be satisfied. In addition, one necessarily cares about multiple testing as there are many sites to test. At the end of this section we will discuss natural enhancements to the simple procedures demonstrated here.

Before we begin to analyze the modification signal for the Synthetic template data, it is important to understand the topology of the synthetic molecule. The sequence which we align to is a reverse complement of itself with a hairpin at one end and a Smrtbell at the other end. It the default pipeline, the Smrbell is found and the reads are partitioned into individual "subreads" based on that adapter location. Due to the shortness of the reference sequence which we align to there are some "edge effects" - we can see this in Figure 10, where the coverage tails off dramatically at the ends of the sequence. We will generally ignore the regions towards the end of the reference sequence as they are an artifact of the alignment method.

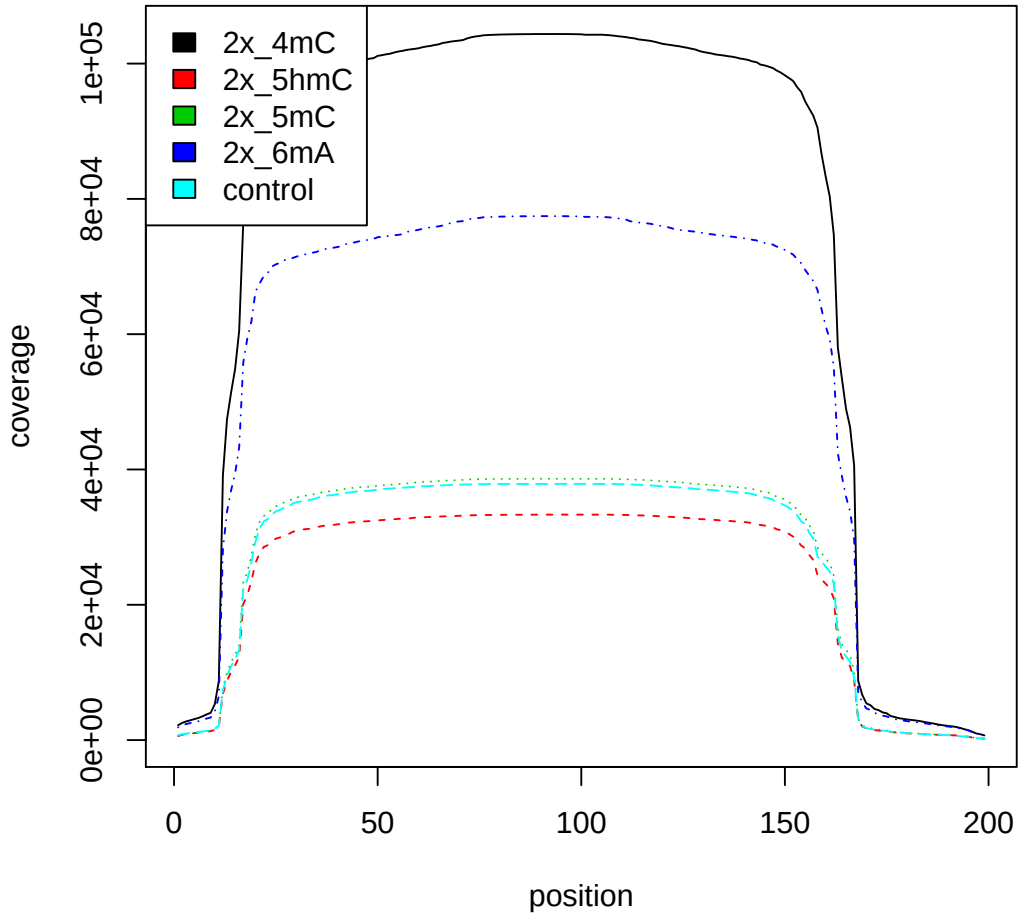


Figure 10: *Coverage Across Synthetic Reference* - Here we plot the "pileup" coverage across the synthetic template. We can see the extreme dropoff of coverage towards the ends of the sequence.

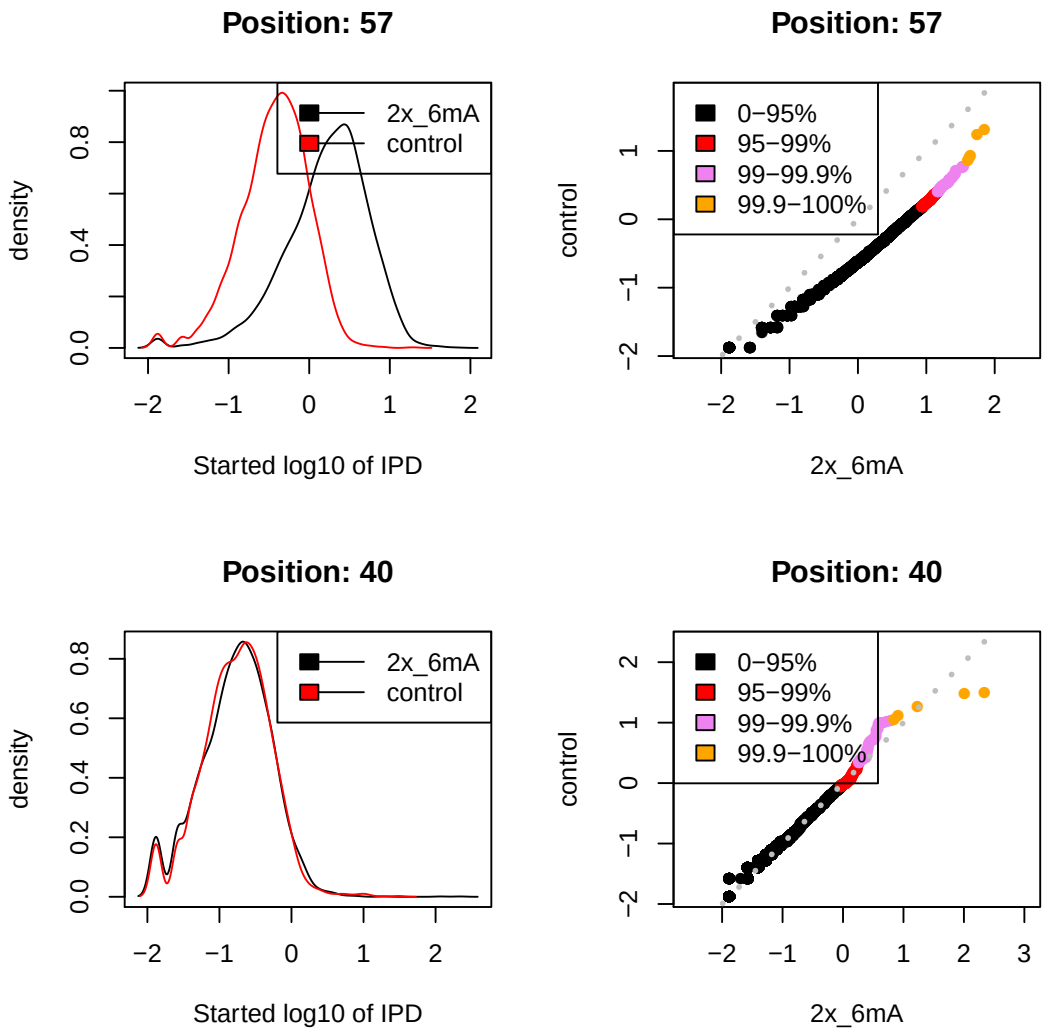


Figure 11: *Density at position and Strand* - Here we plot the IPD distribution for both a trully modified site as well as a non-modified site. We can see that there is a very large effect on the distribution when the modification is a methyl-A.

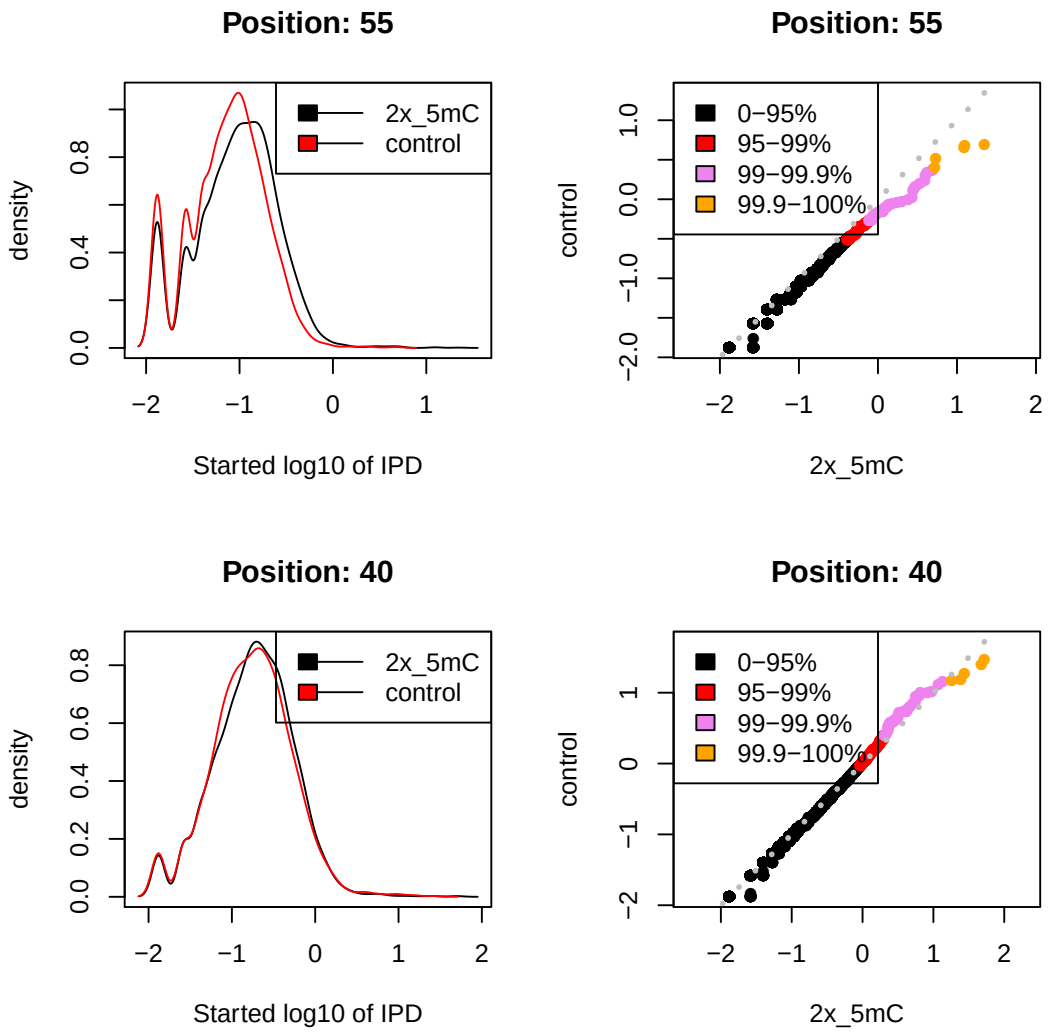


Figure 12: *Density at position and Strand* - Here we plot the IPD distribution for both a trully modified site as well as a non-modified site. We can see that the effect on IPD is much smaller when the modification is a methyl-C indicating that we will need larger sample sizes to determine a true effect.

As Figures 9, 11 and 12 demonstrate, a natural statistic when comparing the IPDs of a control sample to a treatment sample is the mean ratio, either logged or unlogged. We sometimes refer to the following statistic S as the IPD ratio:

$$S = \frac{1/N_{treatment} \sum_{i=1}^{N_{treatment}} IPD_{i,treatment}}{1/N_{control} \sum_{i=1}^{N_{control}} IPD_{i,control}} \tag{1}$$

Here S is specific to a particular reference position, and $N_{treatment}$ corresponds to the number of IPD events at position p in the treatment sample. In Figure 13, we plot S as defined above for the four methylated synthetic templates.

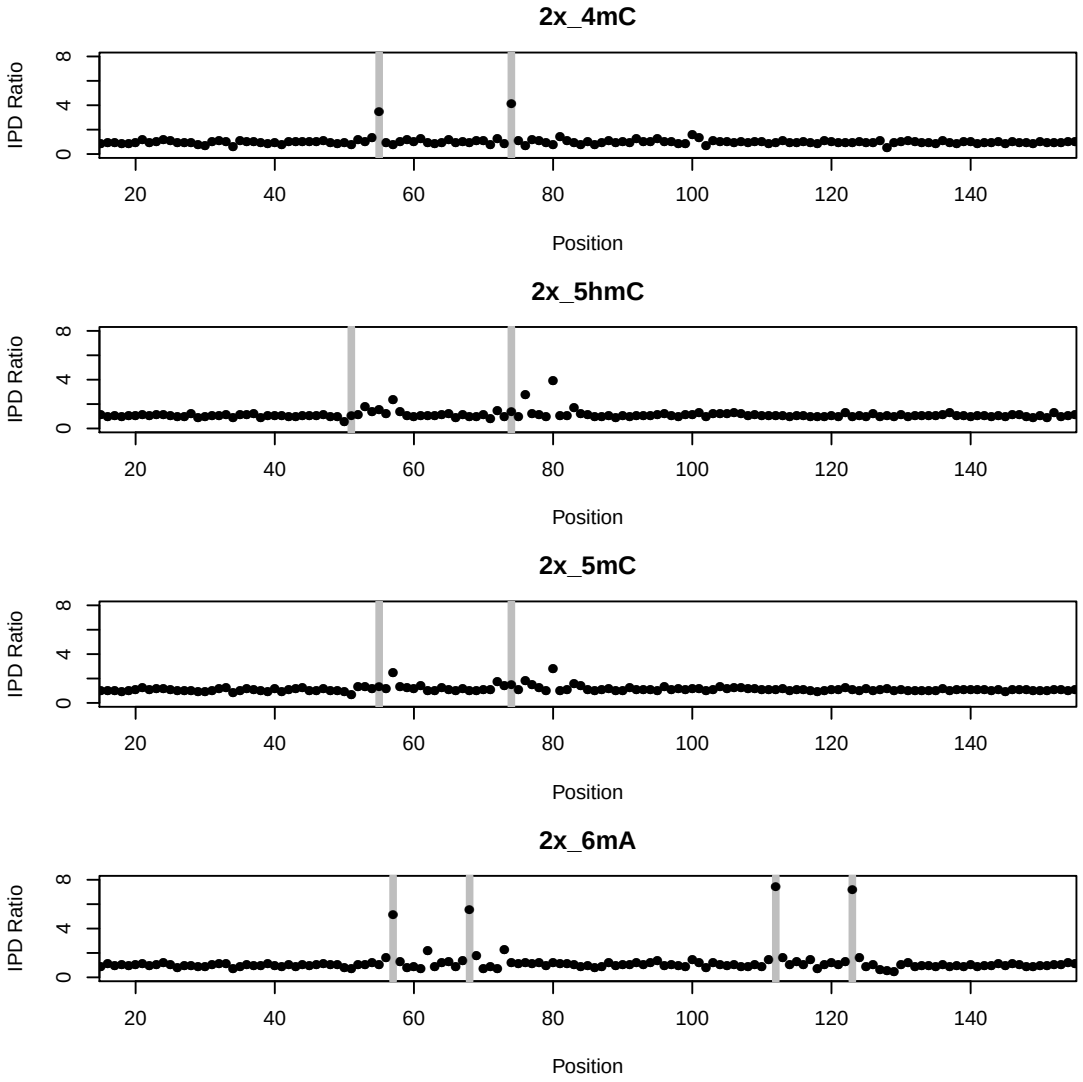


Figure 13: *IPD Ratios* - Here we plot the IPD ratio statistic. This statistic is a measure of the mean shift in IPD distributions. One pertinent aspect of the IPD data that this plot shows is that the effect on the IPD of a methylated base.

3.1 Testing by Coverage

Naturally, we generally want more than just an IPD ratio, or measure of the difference between two distrubtions, we want to know whether that difference would have been likely to be observed by chance. As is clear from Figure 13, the different modifications have dramatically different signals and effect sizes. Below we investigate the performance of a Wilcox test when making the comparison for increasingly larger amounts of coverage. The Wilcox test is general and robust and does not depend on a particular form of the distribution. Where site-wise statistical testing becomes more subtle is the independence assumption between sites (i.e., position 10 and 11 contain a large number of common reads) as well as the independence assumption within molecule. Both of these assumptions need to be addressed when correcting for multiple testing.

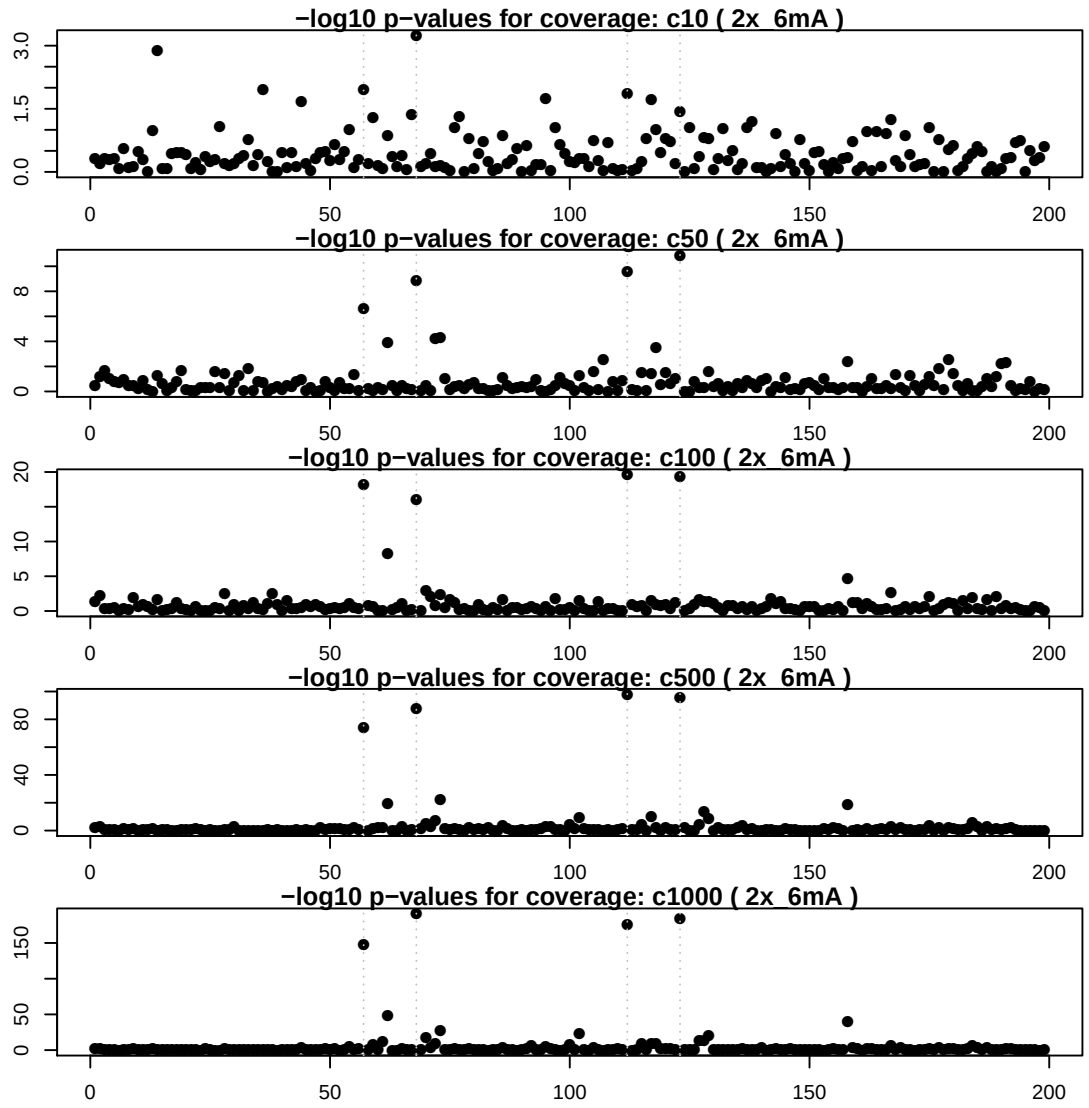


Figure 14: Here we plot the $-\log_{10}$ p-values from the Wilcox test for increasing levels of coverage for the 2x_6mA modified template.

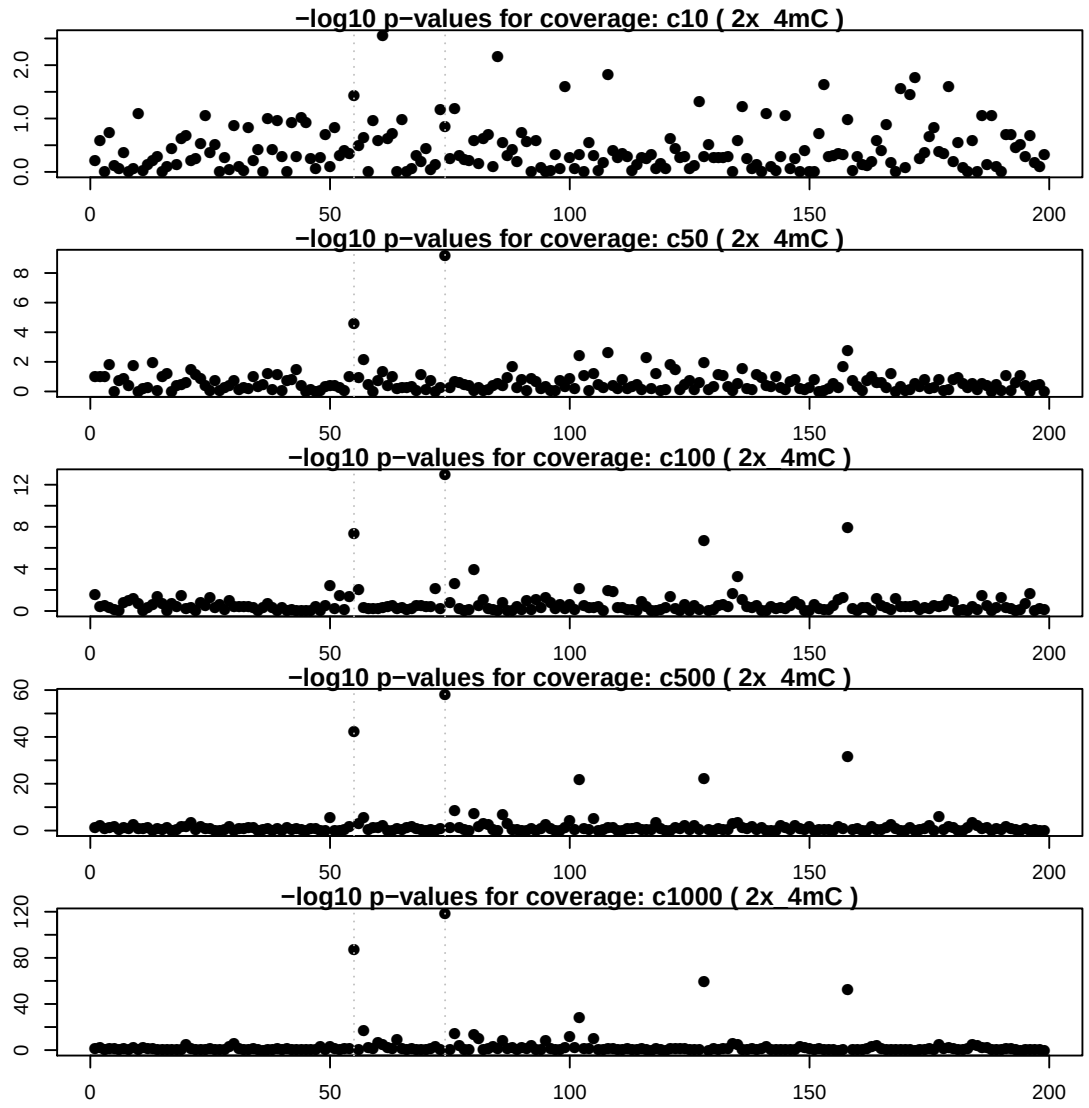


Figure 15: Here we plot the $-\log_{10}$ p-values from the Wilcox test for increasing levels of coverage for the 2x_4mC modified template.

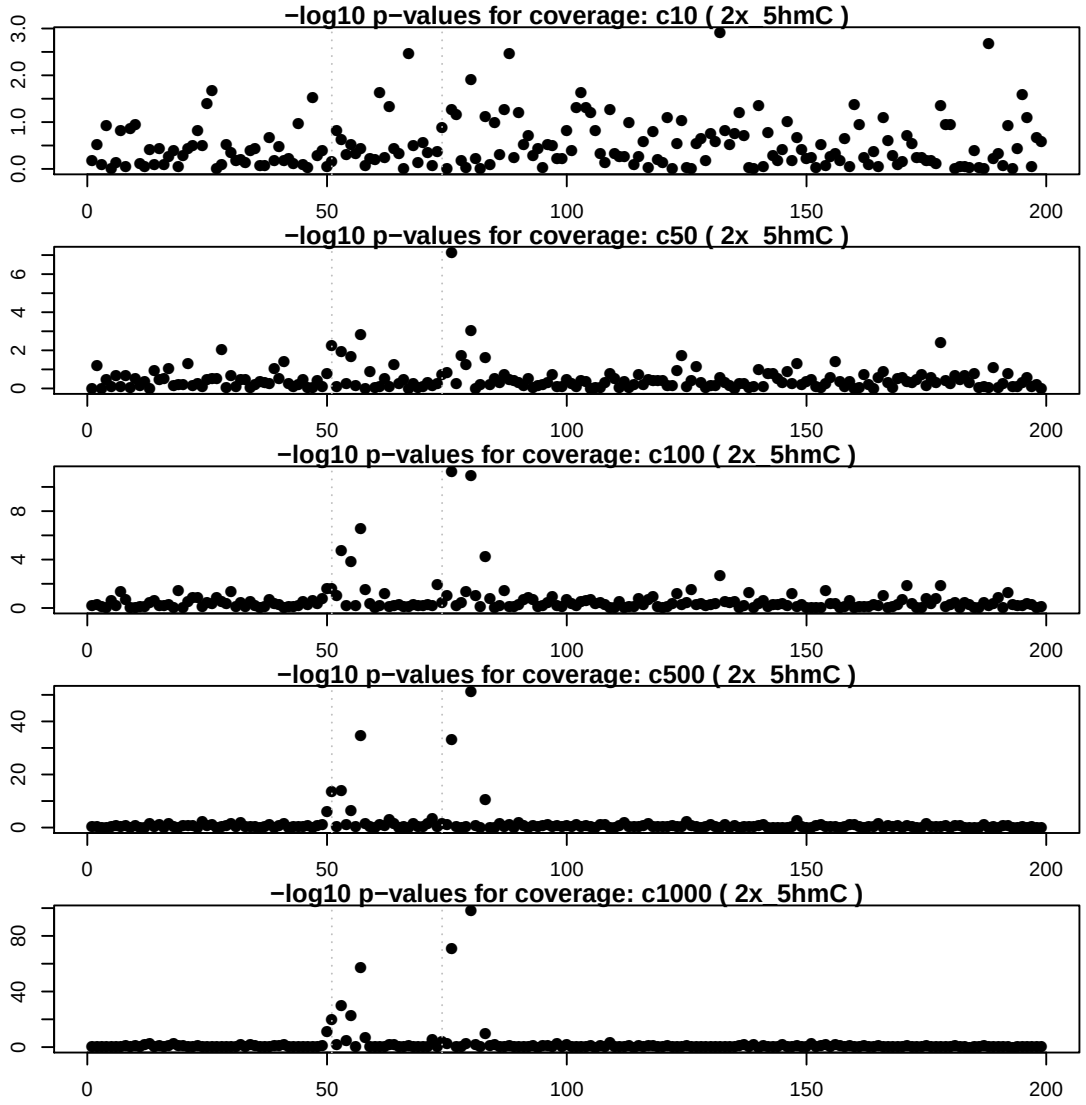


Figure 16: Here we plot the $-\log_{10}$ p-values from the Wilcox test for increasing levels of coverage for the 2x_5hmC modified template.

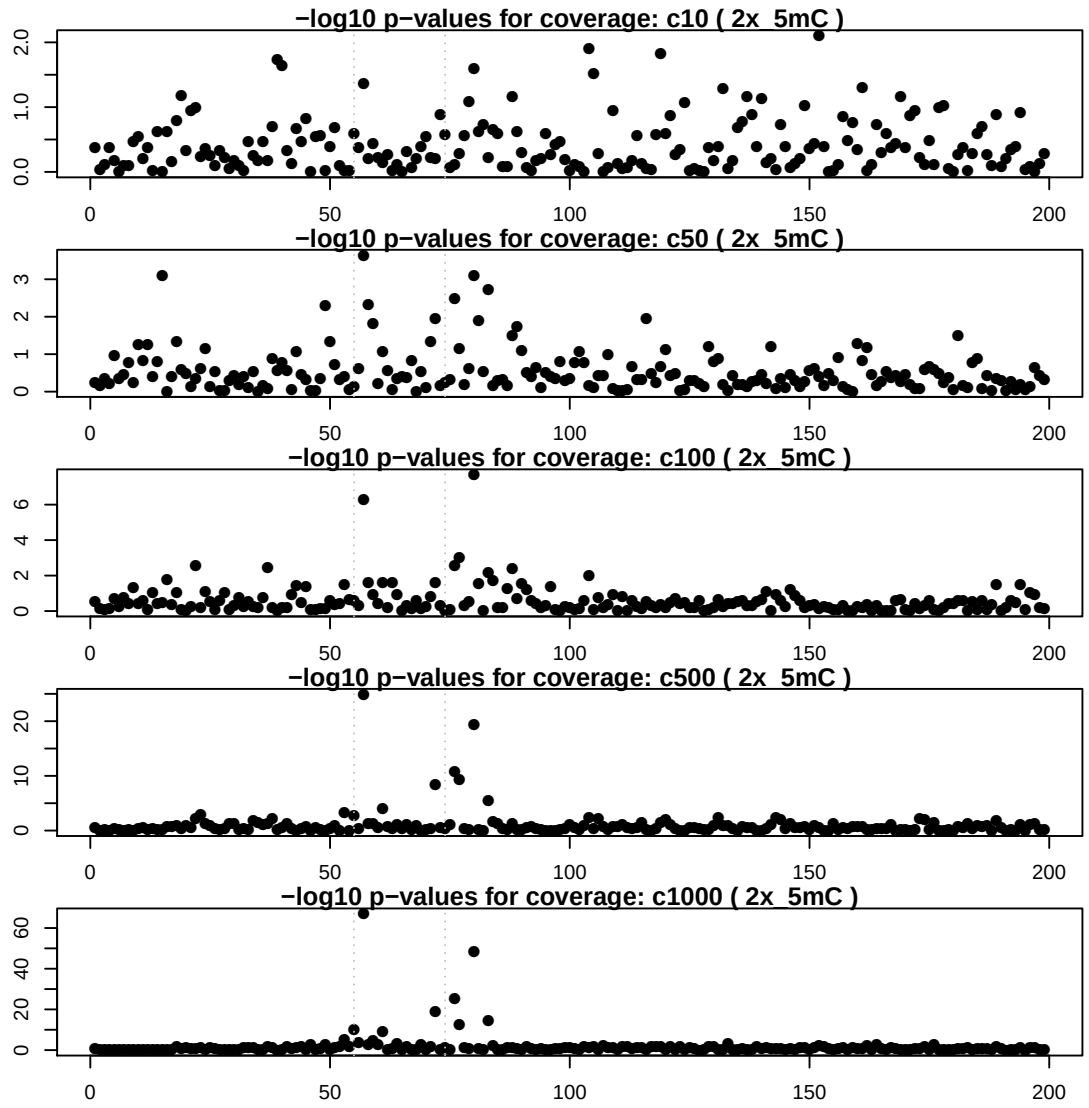


Figure 17: Here we plot the $-\log_{10}$ p-values from the Wilcox test for increasing levels of coverage for the 2x_5mC modified template.

3.2 ROC Analysis

In this section we evaluate the performance of 3 different statistical tests via ROC analysis. We can determine if a particular testing procedure outperforms another in general. Additionally, we can get a sense of our true-positive and false-positive rates for a particular modification. The different tests that we employ are three related tests where each position is tested independently of the other positions. In general, as we can see from the $-\log_{10}$ p-values by position plots, the effect of a modification alters the IPD distribution in nearby bases. More sophisticated tests should take this into account.

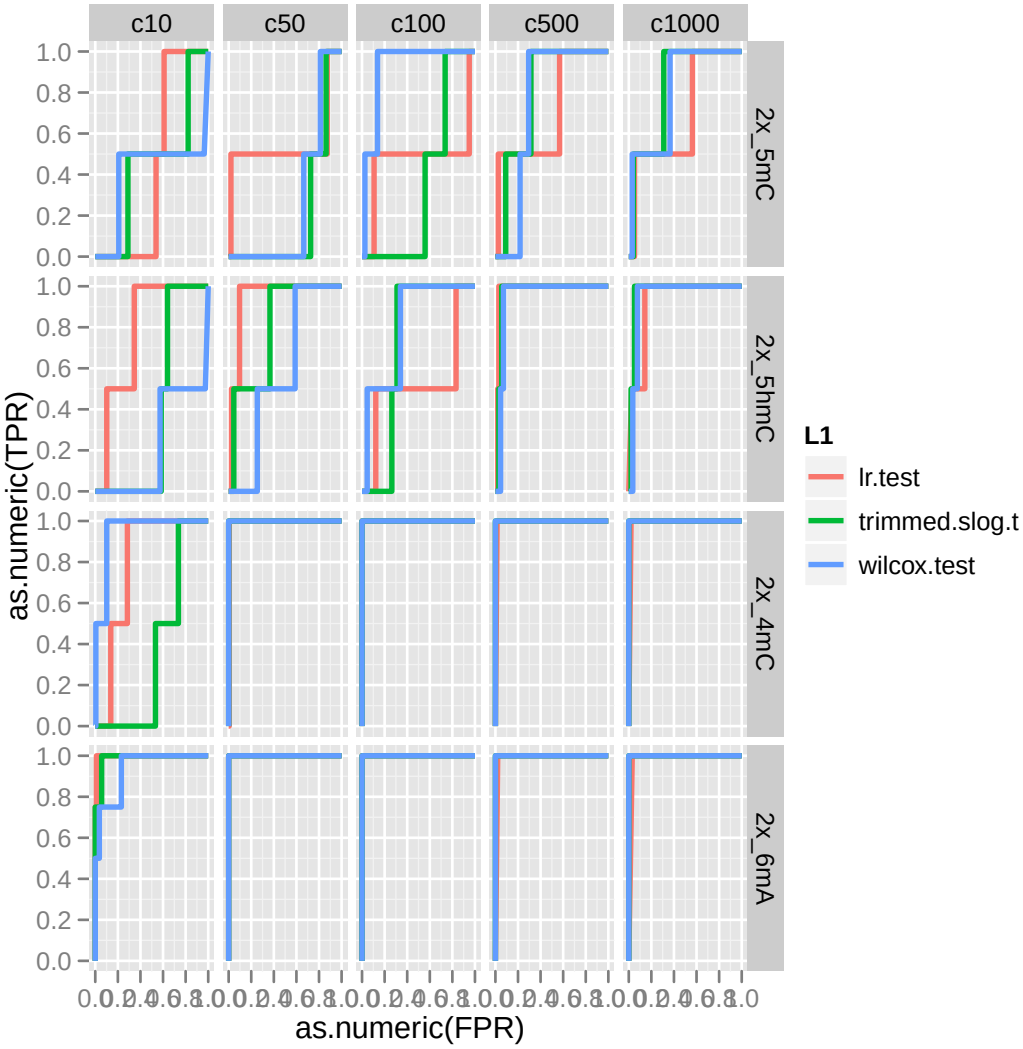


Figure 18: *ROC Curves* - Here we plot ROC curves for the three different testing procedures faceted by coverage and modification type. These curves demonstrate the differences in the magnitude of the modification effects. It is clear the 6mA is quite easy to detect, even at a relatively low level of coverage. However, it is equivalently clare that 5mC has a much smaller effect on the IPD distribution.

As we can see, the differing test procedures do not produce dramatically different results. Indeed, we already knew we would produce false positives in the case of the 5mC modification as its effect on the IPD occurs three bases downstream. In any case, we can see that at sufficient coverage our false positive/true positive tradeoff is quite good for 5hmC, 4mC, and 6mA modifications is quite good. Additionally, as we saw earlier, taking into account position, strand, and context variability will be quite relevant to obtaining more precision and therefore lower coverage requirements.

3.3 Statistical Testing in Lambda

As mentioned previously, a much more realistic dataset is the lambda dataset where we have 4 distinct conditions (Table 2). In this context, we have both a DAM+ and a DAM- condition which we can compare as well as their corresponding WGA vs. Native preparations. For simplicity, we can compare

DAM+ and DAM- and focus on GATC methylation sites as our true positives.

For this analysis, we will need to use the Biostrings package to determine where the GATC sites are in the genome. As an example we can look at one position.

```
> if (!require(Biostrings)) {
+     stop("Unable to execute Lambda testing examples without Biostrings package.")
+ }
> lambda <- read.DNAStringSet("../ReferenceRepository/lambdaNEB/lambdaNEB.fa")[[1]]
> matches <- matchPattern("GATC", lambda)
> gatcExample <- pbutils::collapse(lapply(cmpH5s[c("6mA_dam+_native",
+     "6mA_dam-_native")], function(cmp) {
+     s <- start(matches)[1]
+     e <- end(matches)[1]
+     subset(getByTemplatePosition(cmp, idx = getReadsInRange(cmp,
+         1, s, e), f = getIPD), position >= s & position <= e &
+         read == ref)
+ }))
```

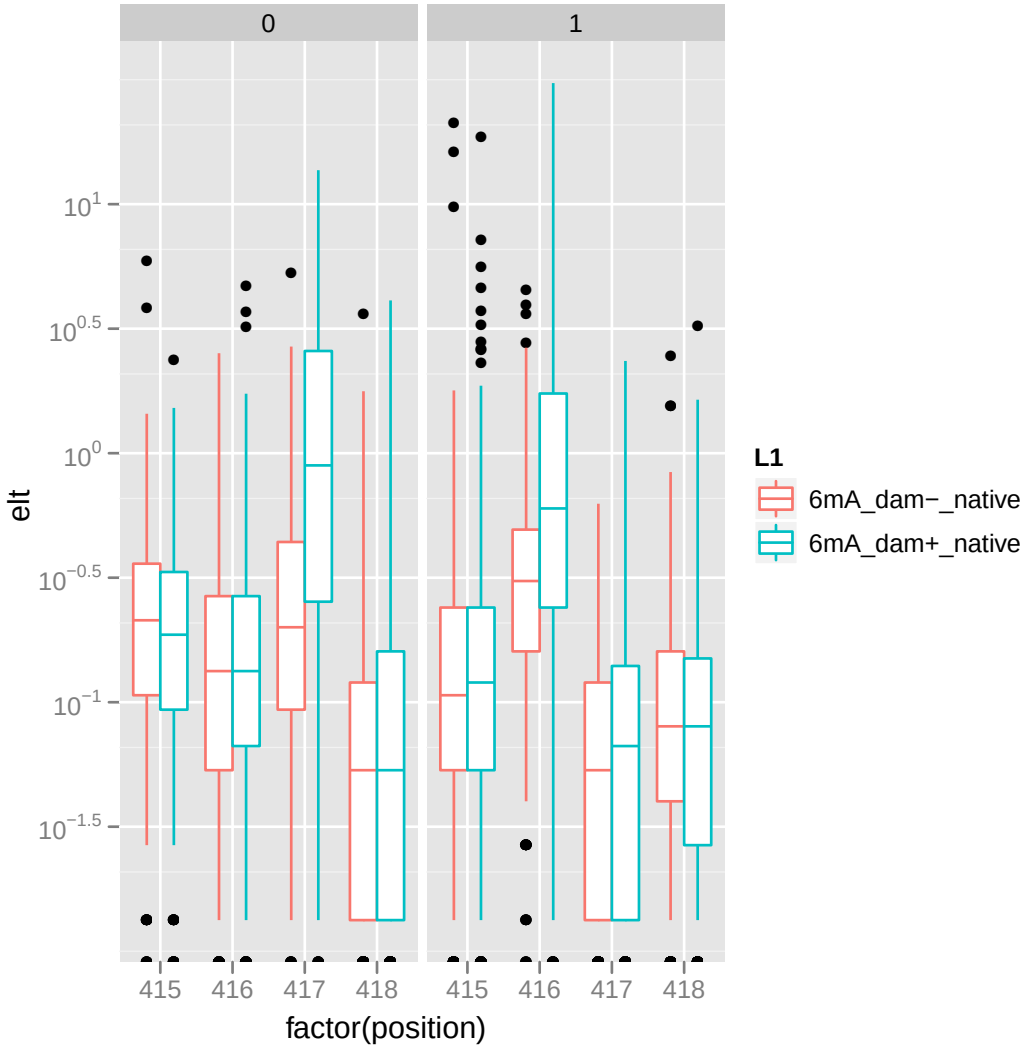


Figure 19: *GATC modification* - At this particular GATC modification, we can see that there is a strong signal of the modification and the reference position depends on the strand. This indeed makes sense as the strand we report means that we had to reverse complement the reference sequence to match the incorporated bases. Said another way, the template was the forward strand of the reference in this instance.

4 Conclusion

This document attempts to provide a user with an overview of PacBio data and APIs which support performing Kinetics analyses. This document is an attempt to present the analyst with a starting point for their kinetics analyses using Pacific Biosciences data. This document is a "live" document in the sense that we will be updating the tools presented here for more complicated analyses. For the user who wishes to process their data in the most simplistic fashion, we provide the following utility:

```
> topTable <- makeTopTable(cmpH5s[["6mA_dam+_native"]], cmpH5s[["6mA_dam-_native"]],
+     start = 1, end = 5000)
> head(topTable)
```

```
     position reference read      p.value statistic n.control n.treatment
2533     2533         T    T 4.061199e-58  150313.0       436         422
551       551         T    T 8.576932e-52  135679.5       412         409
3071     3071         T    T 6.792062e-45   89537.0       368         298
417       417         T    T 3.263057e-40  143401.0       422         447
2169     2169         T    T 1.154287e-29  114931.0       364         431
2368     2368         T    T 7.818656e-27  154689.5       481         458
     ipd.ratio          fdr
2533  5.243457 2.030600e-54
551   9.717223 2.144233e-48
3071  7.271965 1.132010e-41
417   5.396483 4.078822e-37
2169  2.566337 1.154287e-26
2368  4.574853 6.515546e-24
```

This utility produces a table ordered by p-value, where each row represents a position in the reference. Users might very well like to modify this utility function, please refer to "utils.R" in the "Src" directory of this project.

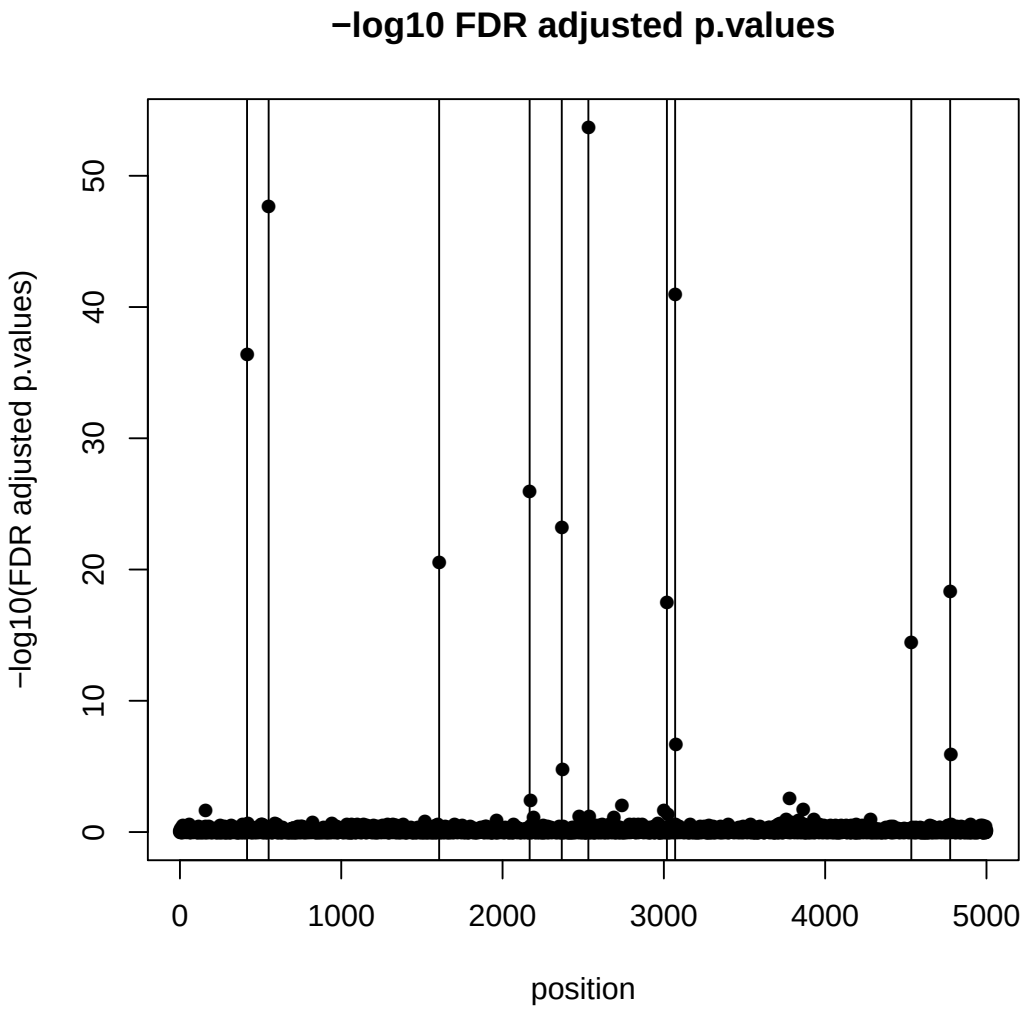


Figure 20: *Lambda FDR Adjusted P-values by Position* - For approximately 5000 bases across the Lambda genome we plot $-\log_{10} p$-values by position. Vertical lines indicate GATC positions in the genome.

4.1 Next Steps

As one can see from this document there are a number of enhancements that can be conceived of in the two-sample comparison case. First, one might prefer a permutation test when comparing the native vs. WGA samples. In the "utils.R" file there exists an example of such a procedure. One might prefer an empirical Bayes approach when the number of reads is low. Finally, one might wish to investigate whether taking into account the modification signature rather than just the individual IPD provides a substantially more powerful test than just the individual position.

4.2 Session Info

```
> sessionInfo()
```

```
R version 2.13.1 Patched (2011-09-13 r57007)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=C              LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] grid      stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] Biostrings_2.20.3 IRanges_1.10.6    ggplot2_0.8.9     proto_0.3-9.2
 [5] reshape_0.8.4     plyr_1.6          xtable_1.5-6      pbls_1.0
 [9] pbutils_1.0       pbh5_1.1          h5r_1.2

loaded via a namespace (and not attached):
[1] digest_0.5.0 tools_2.13.1
```