DNA-Modification Detection with SMRT-Sequencing using R

Pacific Biosciences

October 18, 2011

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 analysts to conduct their own kinetic analysis.

1.1 R Packages/System Requirements

In this analysis we will make heave 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 ggplot2 from Hadley Wickham and the xtable package on CRAN. All of the analysis conducted herein can be performed just using the pbh5 package, 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.

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 which will be used in comparison to each treatment template. For the lambda data set, we have both

	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.

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.

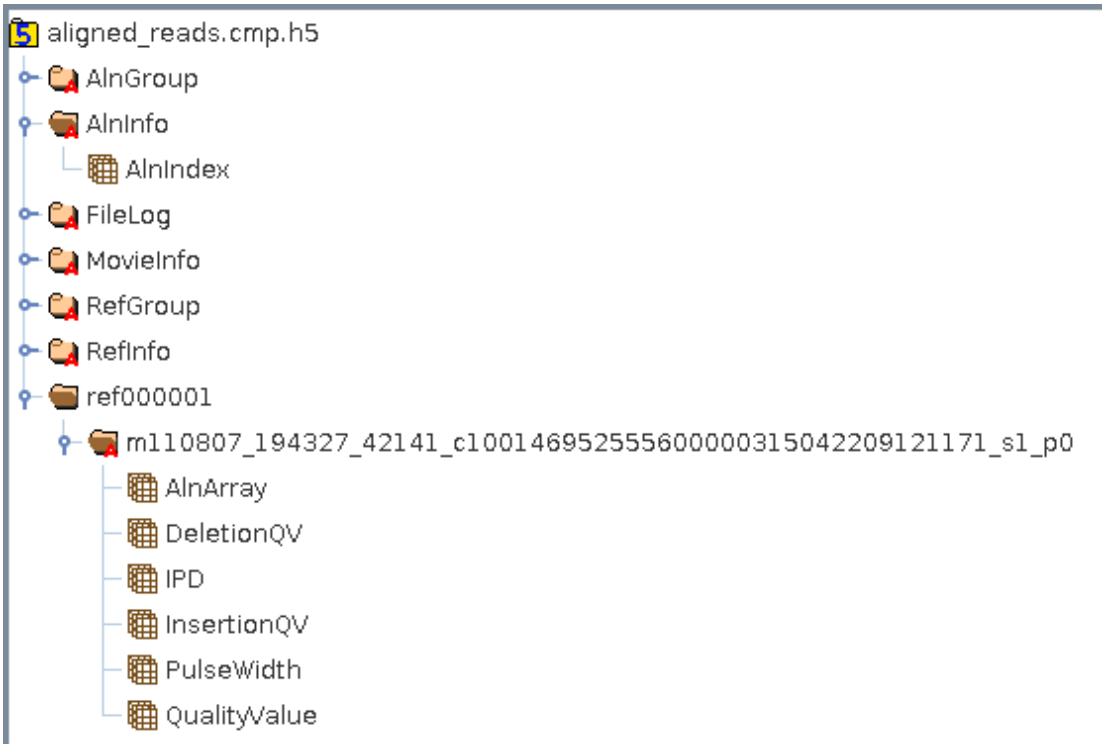

Figure 1: *cmp.h5 Structure* - A screenshot of the cmp.h5 file structure. PacBio cmp.h5 files provide a wealth of additional information about the sequencing reaction. Broadly, HDF5 files can be thought about as a file system for your data.

2.1 Working with the Compare H5 File

The cmp.h5 file (pronounced comp H5 or compare H5) provides a rich set data resulting from the alignments of Pac Bio 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.

```
  RefInfoID ID       Path offsetBegin offsetEnd      Name       FullName Length
1         1  1 /ref000001           1    149408 ref000001 lambda_NEB3011  48502
                              MD5
1 a1319ff90e994c8190a4fe6569d0822a
```

The alignments, along with all of the relevant kinetics data, are stored in a directory like structure corresponding to their reference and movie, e.g.,

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

Each of the datasets stored here represent all of the alignments for a given movie. For our work, the most pertinent datasets are: AlnArray, IPD (inter-pulse duration), and PulseWidth. The IPD and PulseWidth describe kinetic properties of the sequencing and the AlnArray will tell us which base we are incorporating. All of the alignments in the file are stored in a global alignment index, which can be accessed as follows:

```
     ID
1 93787
```

```
2 74915
                                                                alnGroupPath
1 /ref000001/m110818_075520_42141_c100129202555500000315043109121112_s1_p0
2 /ref000001/m110818_075520_42141_c100129202555500000315043109121112_s2_p0
                                                          movieName   refName
1 m110818_075520_42141_c100129202555500000315043109121112_s1_p0 ref000001
2 m110818_075520_42141_c100129202555500000315043109121112_s2_p0 ref000001
     fullRefName tStart tEnd alignedStrand holeNumber setNumber strobeNumber
1 lambda_NEB3011      1   89             0      22178         1            0
2 lambda_NEB3011      1   97             1      17892         2            0
  moleculeID rStart rEnd mapQV nMatches nMisMatches nInsertions nDeletions
1     342178    278  381   254       87           0          17          2
2     657892   3520 3625   254       93           4           9          0
  offsetBegin offsetEnd nBackRead nOverlap
1     4542502   4542607         0        0
2    25681054  25681159         1        1
```

or more succinctly,

```
     ID
1 93787
2 74915
                                                                alnGroupPath
1 /ref000001/m110818_075520_42141_c100129202555500000315043109121112_s1_p0
2 /ref000001/m110818_075520_42141_c100129202555500000315043109121112_s2_p0
                                                          movieName   refName
1 m110818_075520_42141_c100129202555500000315043109121112_s1_p0 ref000001
2 m110818_075520_42141_c100129202555500000315043109121112_s2_p0 ref000001
     fullRefName tStart tEnd alignedStrand holeNumber setNumber strobeNumber
1 lambda_NEB3011      1   89             0      22178         1            0
2 lambda_NEB3011      1   97             1      17892         2            0
  moleculeID rStart rEnd mapQV nMatches nMisMatches nInsertions nDeletions
1     342178    278  381   254       87           0          17          2
2     657892   3520 3625   254       93           4           9          0
  offsetBegin offsetEnd nBackRead nOverlap
1     4542502   4542607         0        0
2    25681054  25681159         1        1
```

To access an alignment or data associated with an alignment, we will use accessor functions which take a cmp.h5 file as well as a vector of indices refering to the rows in the alignment index which we want to retrieve.

```
[1] 106 106 116
```

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

```
[4,] "C"  "C"
[5,] "G"  "G"
[6,] "G"  "G"
```

The above command retrieves the first 3 alignments. 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`.

2.2 Visualizing Kinetic Properties of the System

In this section, we visualize pulse width and IPD (inter-pulse duration). 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.

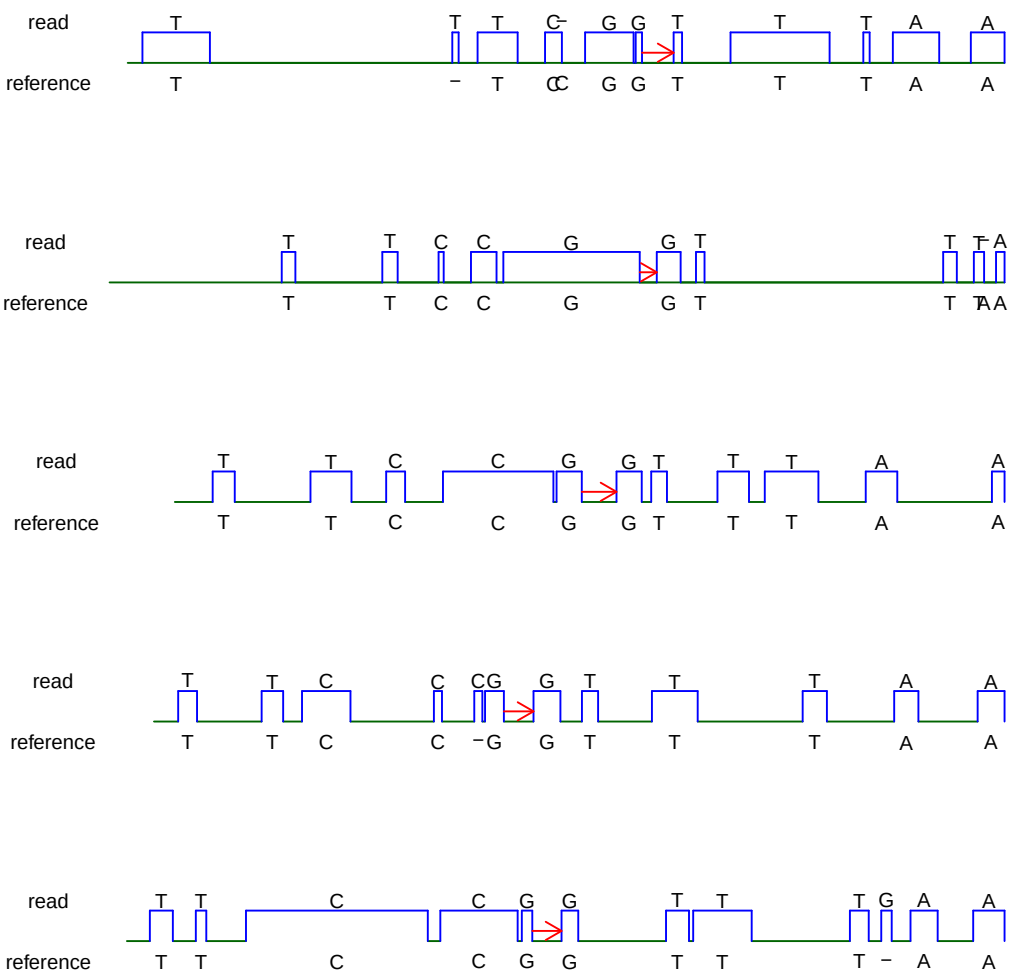

Figure 2: *Trace Cartoon* - Here we plot "trace" views directly from the cmp.h5 file. When we allow the possibility of insertions/deletions/mismatches it is more complicated to ensure that you are viewing a proper incorporation event.

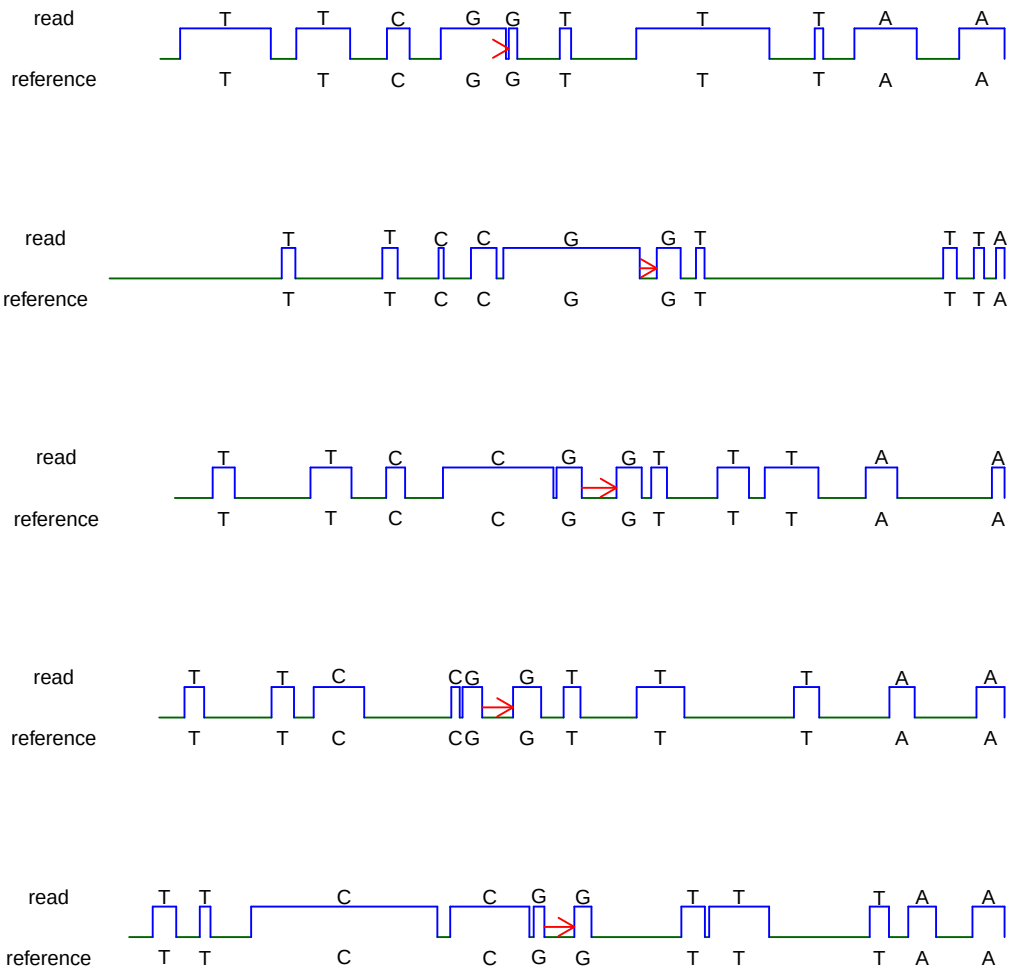

Figure 3: *Trace Cartoon* - Here we plot "trace" views directly from the cmp.h5 file. This is the same plot as above, however we are restricting things to those bases which match. In practice, such a filtering will be too stringent.

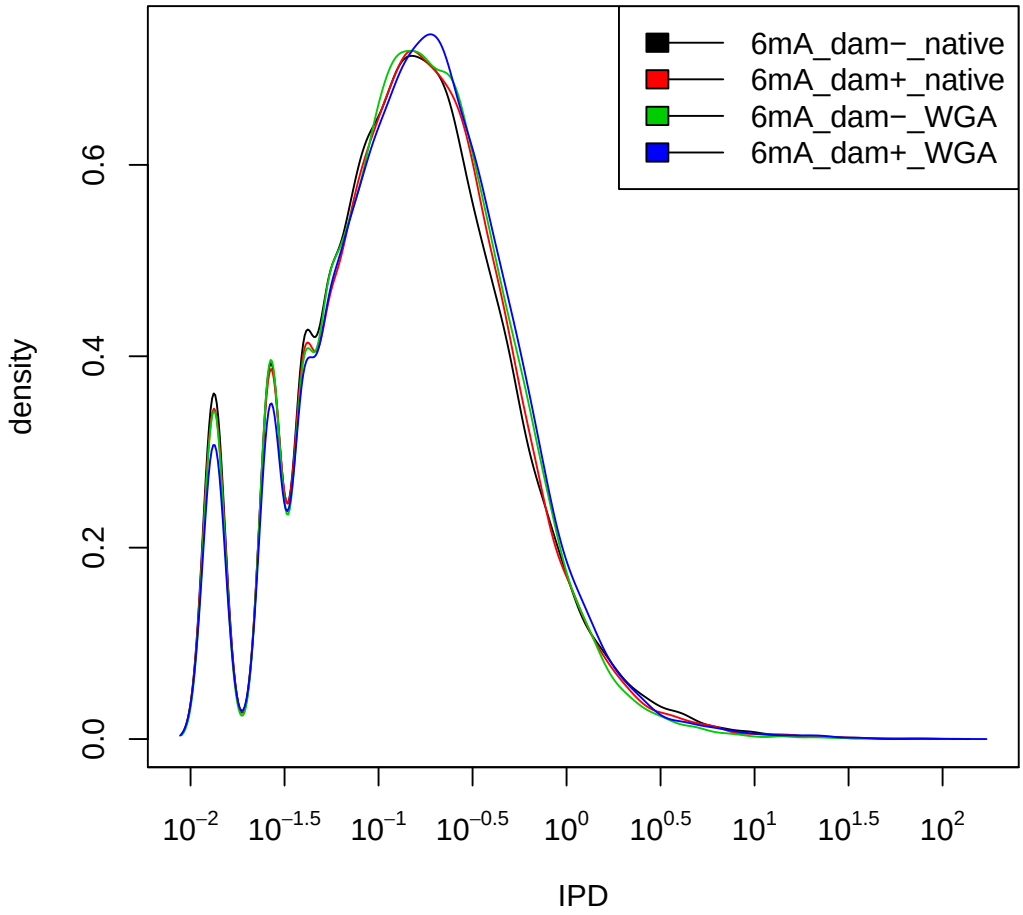

Figure 4: *Global IPD Density* - Here we plot the global IPD distribution. This distribution is a mix of incorporations of the 4 nucleotides.

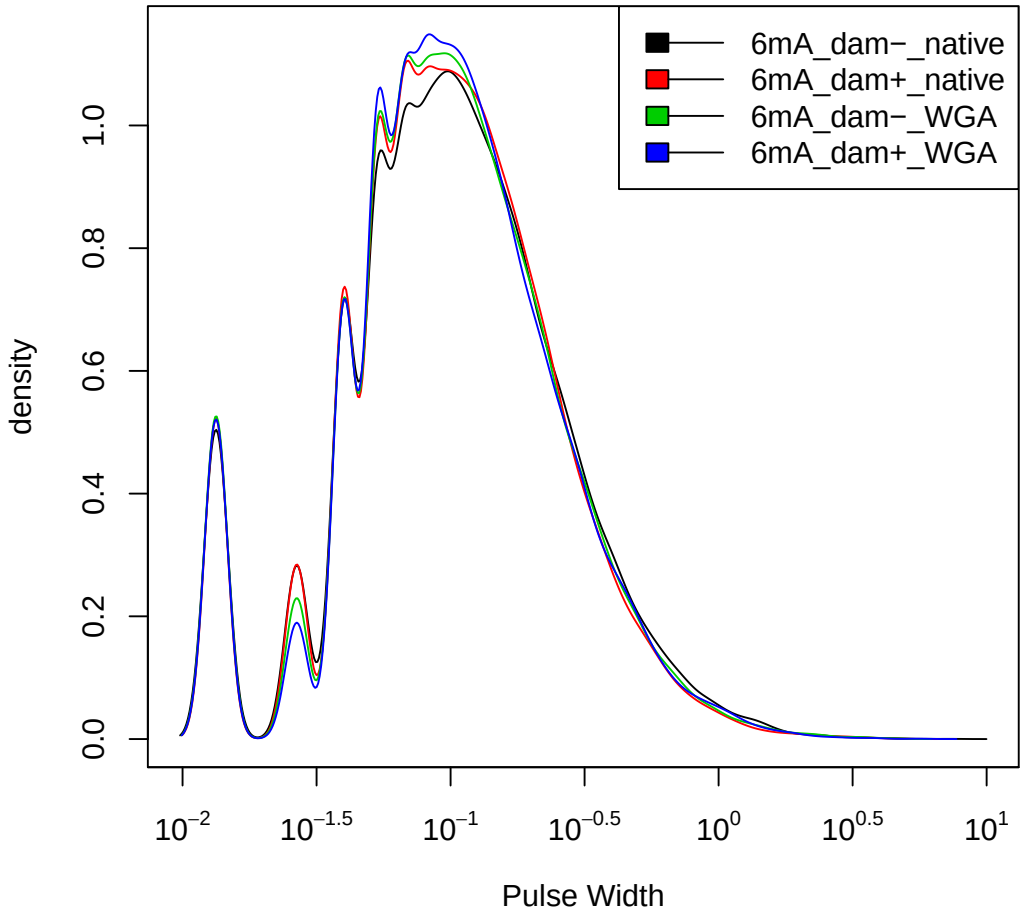

Figure 5: *Global Pulse Width Density* - Here we plot the global Pulse Width distribution. This distribution is a mix of incorporations of the 4 nucleotides.

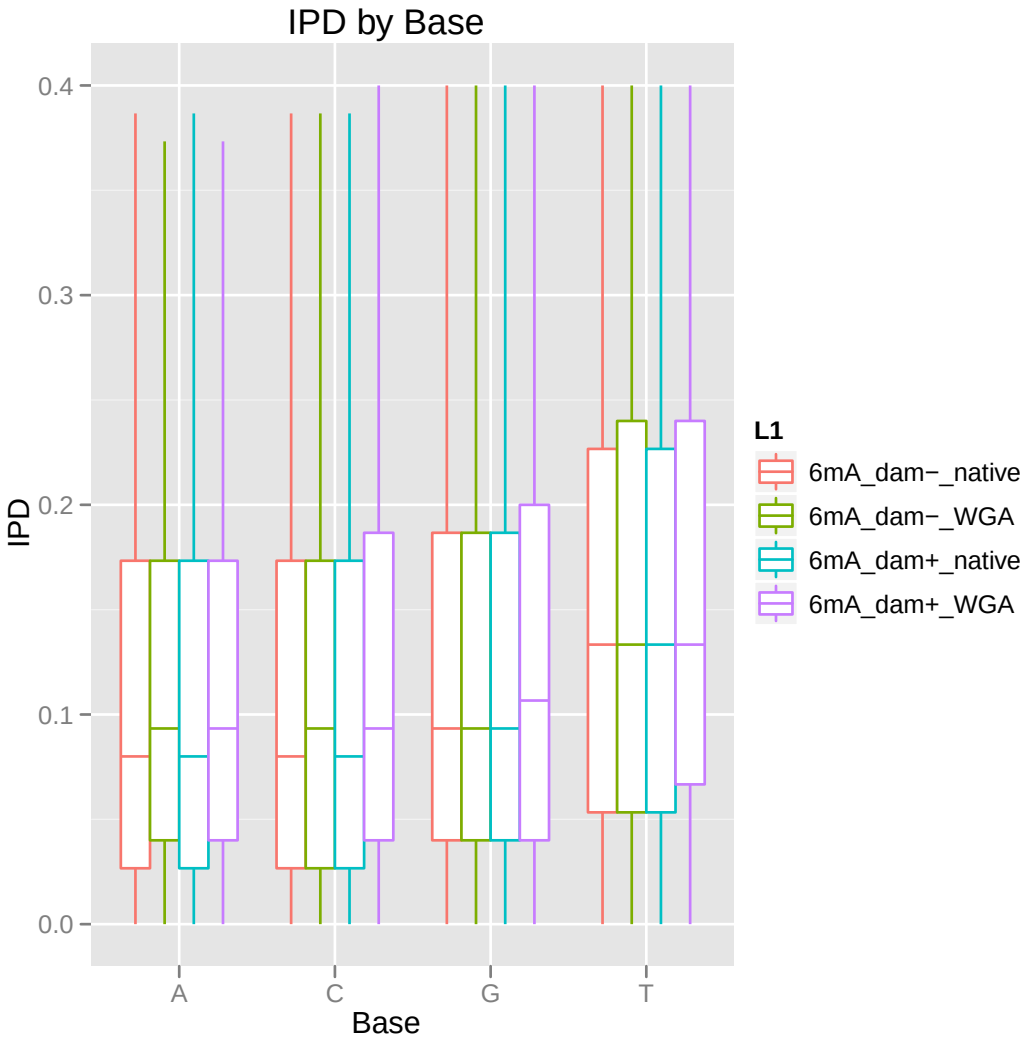

Figure 6: *IPD By Base* - Here we plot the IPD distribution stratified by the base being incorporated. We see that there is a base effect on IPD, i.e., which base we are incorporating changes the kinetic behavior of the enzyme.

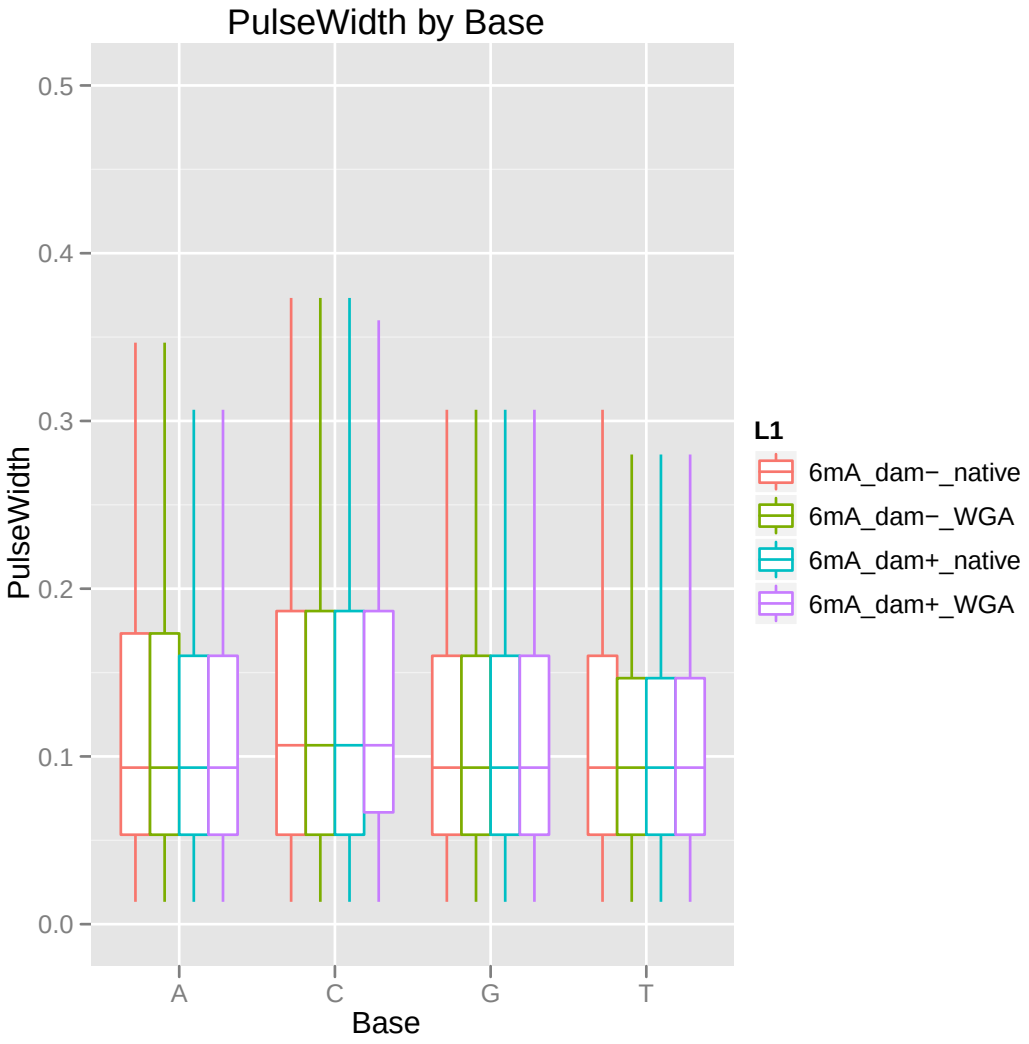

Figure 7: *Pulse Width By Base* - Here we plot the IPD distribution stratified by the base being incorporated. We see that there is a base effect on IPD, i.e., which base we are incorporating changes the kinetic behavior of the enzyme.

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    G   G   1      0 0.15999961
2        2    G   G   1      0 0.37333241
3        3    G   G   1      0 0.06666651
4        4    C   C   1      0 1.02666414
5        5    G   G   1      0 0.11999971
6        6    G   G   1      0 0.02666660
```

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
1 0.06666651 0.01333330        3           G           G           G
2 1.02666414 0.13333301        4           G           G           C
3 0.11999971 0.06666651        5           G           C           G
4 0.02666660 0.01333330        6           C           G           G
5 0.06666651 0.22666611        7           G           G           C
6 0.01333330 0.09333310        8           G           C           G
  context.P04 context.P05
1           C           G
2           G           G
3           G           C
4           C           G
5           G           A
6           A           C
```

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  1505 0.1199997
AAC   511 0.0933331
AAG   482 0.0933331
AAT   378 0.1066664
ACA   303 0.1599996
ACC   275 0.1466663
```

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)
```

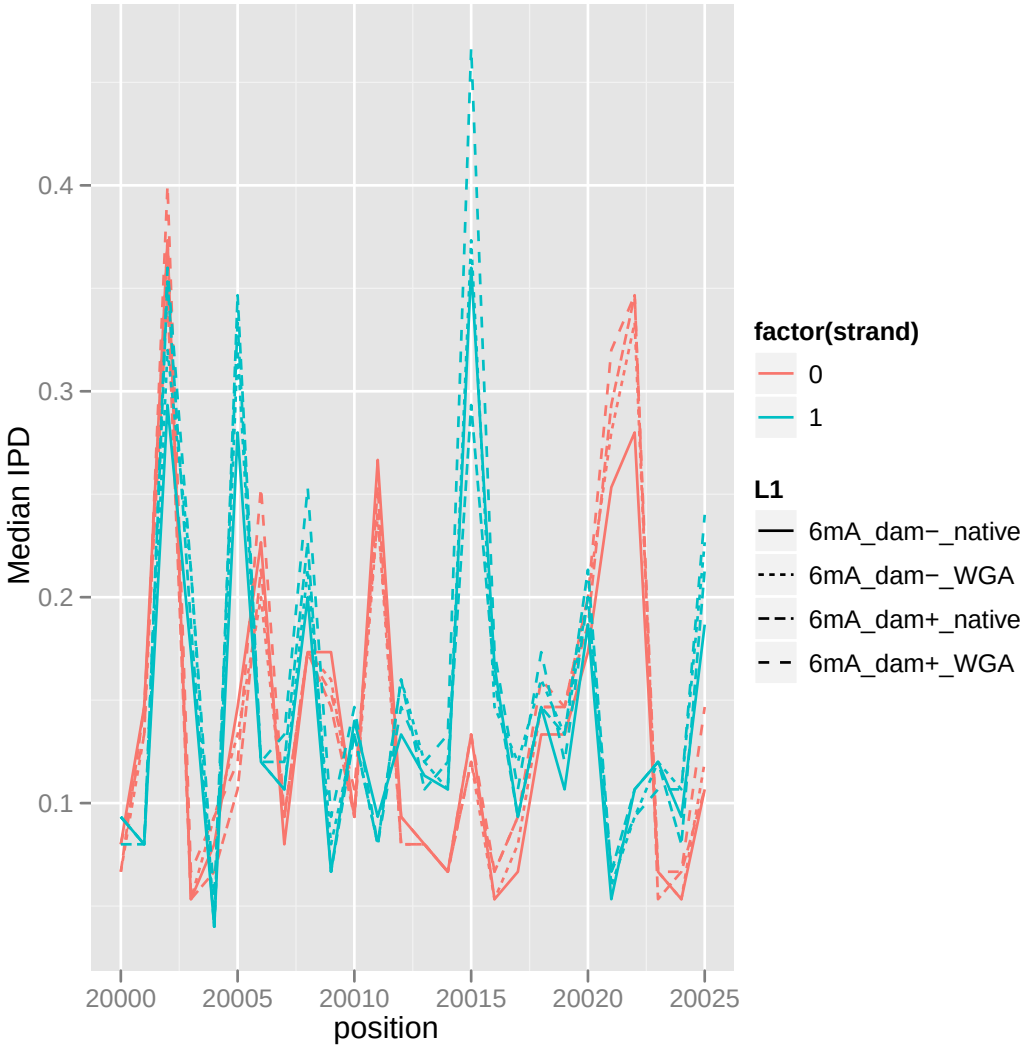

Figure 8: *IPD by Strand and Position* - Here we plot the median of the IPD distribution conditioned on both strand and position. We can see the presence of a very strong position effect

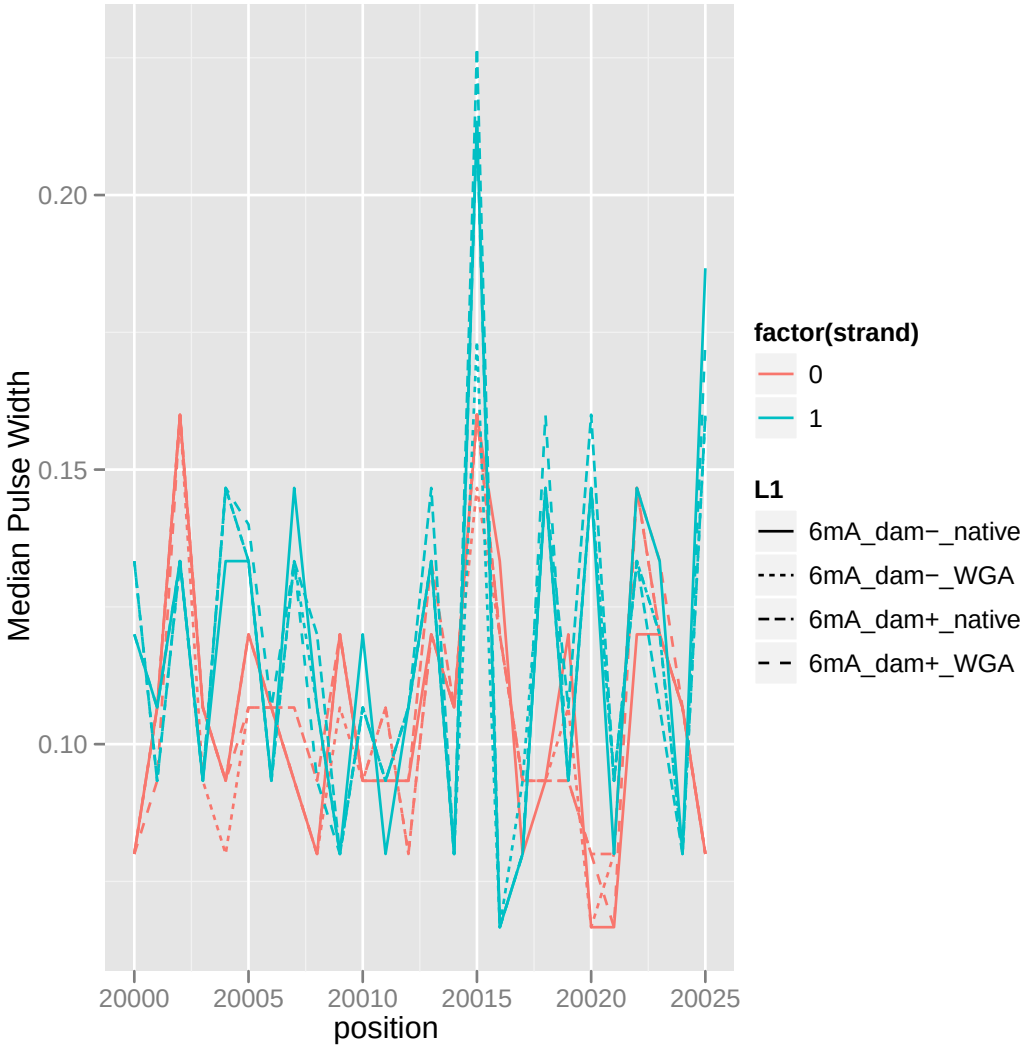

Figure 9: *Pulse Width by Strand and Position* - Here we plot the median of the Pulse Width distribution conditioned on both strand and position. We can see the presence of a position effect - althought the effect is less pronounced than that for IPD.

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: 1, then the reference sequence is reverse complemented rather than the read. The importantance 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.

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

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

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

```
    position read ref idx strand        elt
1          1    G   G   1      0 0.15999961
107        1    C   C   2      1 0.47999883
2          2    G   G   1      0 0.37333241
108        2    C   C   2      1 0.17333291
3          3    G   G   1      0 0.06666651
109        3    C   C   2      1 0.07999980
```

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
1     3   GGGCG
178   3   CGCCC
2     4   GGCGG
177   4   CCGCC
3     5   GCGGC
176   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))
```

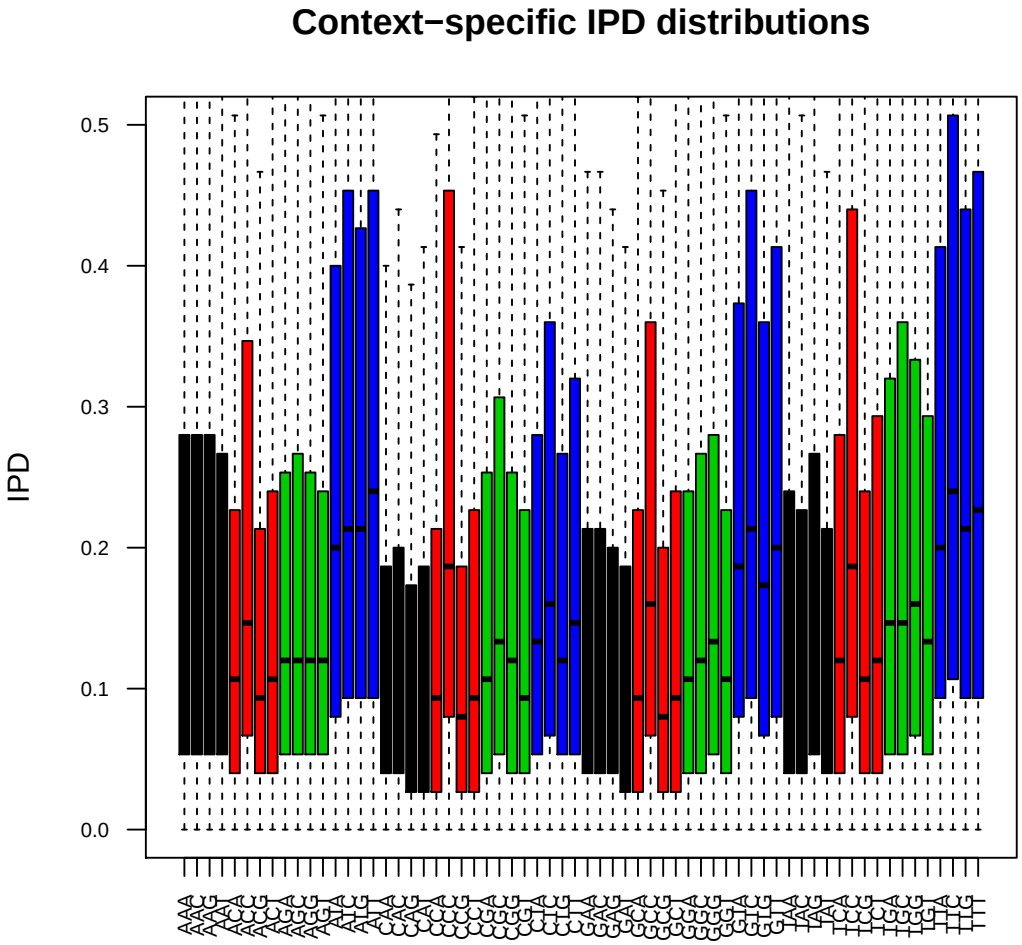

Figure 10: *IPD by Context* - Plots of IPD by context. We can see that the IPD distribution depends on context. Here the boxplots have been colored by the base being incorporated.

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.

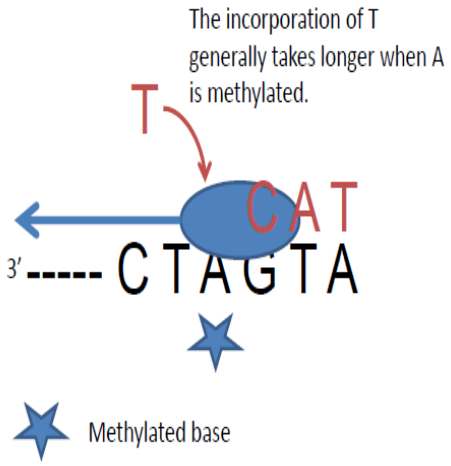

Figure 11: *GATC Cartoon* - A cartoon depicting the incorporation of T which will be "delayed" when the complement A base is methylated.

```
> par(mfrow = c(2, 1), mar = c(3, 5, 1, 1))
> lapply(cmpH5s[c("6mA_dam+_native", "6mA_dam-_native")], function(cmp) {
+     tmp <- associateWithContext(cmp, idx = sample(1:nrow(cmp),
+         size = 5000), f = getIPD, collapse = T, useReference = T,
+         up = 2, down = 2)
+     contextMedians <- tapply(tmp$elt, tmp$context, median, na.rm = T)
+     plot(x <- 1:length(contextMedians), y <- contextMedians,
+         pch = 16, ylim = c(0, 1), xlab = "Context", xaxt = "n",
+         ylab = "Median")
+     w <- grep("^GATC", names(y))
+     text(x[w], y[w], names(y)[w], col = "darkblue", cex = 1.3)
+ })
```

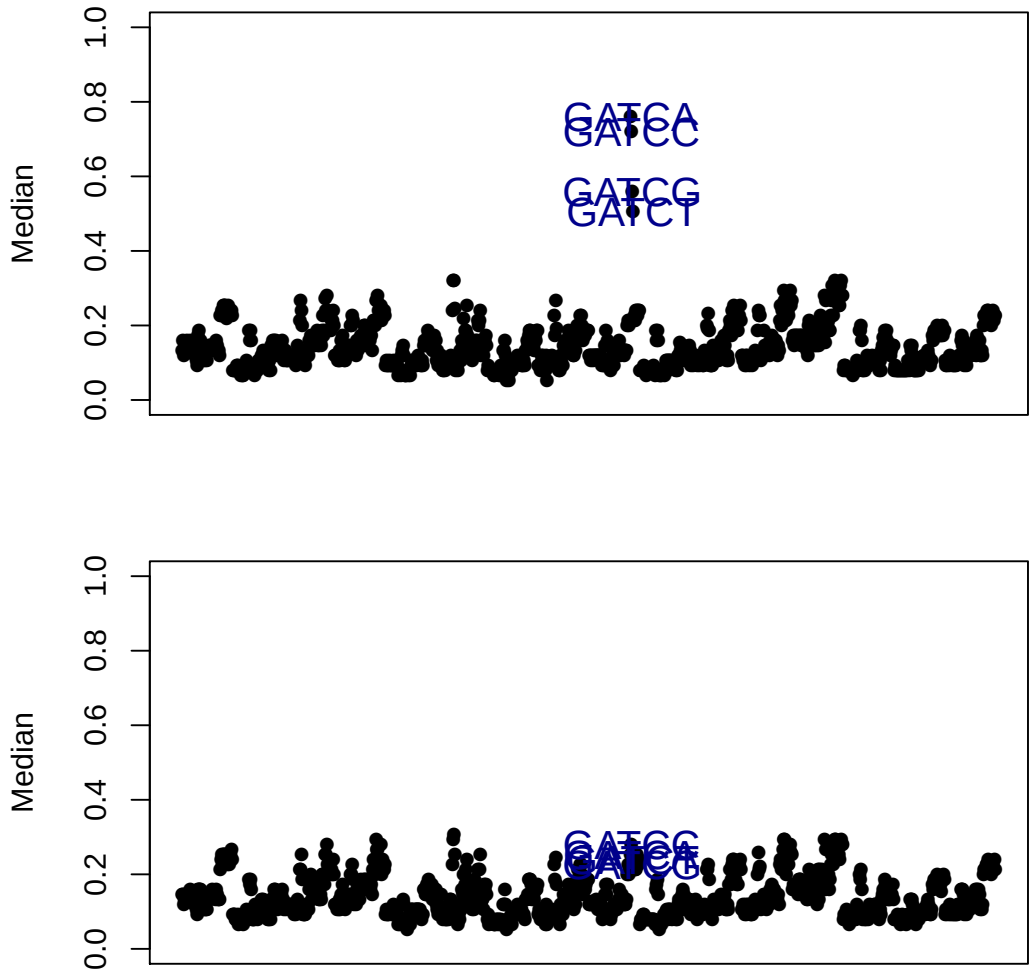

Figure 12: *Context-specific Modificationse* – Here we plot the median IPD for 5 based contexts for both the DAM+ and DAM- Lambda strains. First, the range of IPDs is quite similar for all non-modifi

3 Normalization

3.1 Strand Effect

4 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 existing procedure.

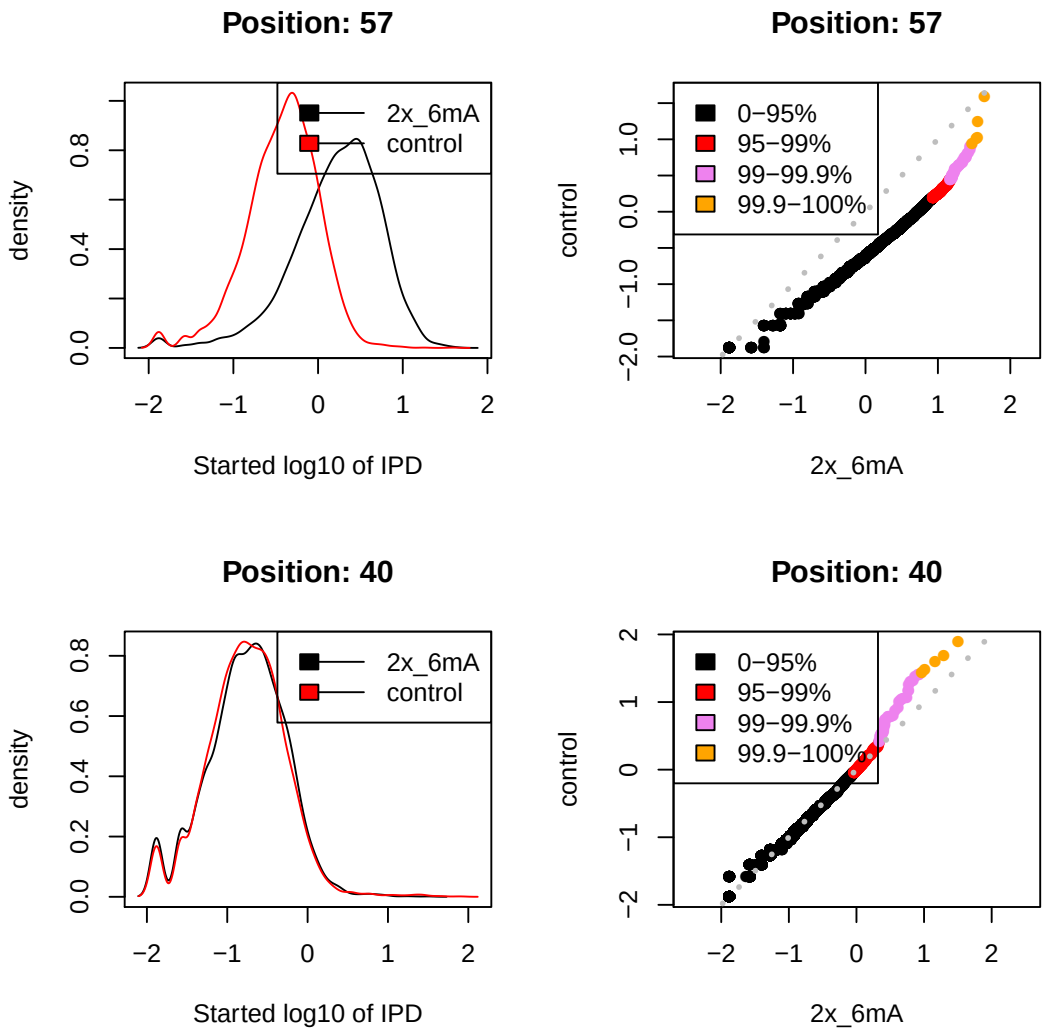

Figure 13: *Density at position and Strand* - Here we plot the IPD distribution for both a trully modified site as well as a non-modified site. We can see that there is a very large effect on the distribution when the modification is a methyl-A.

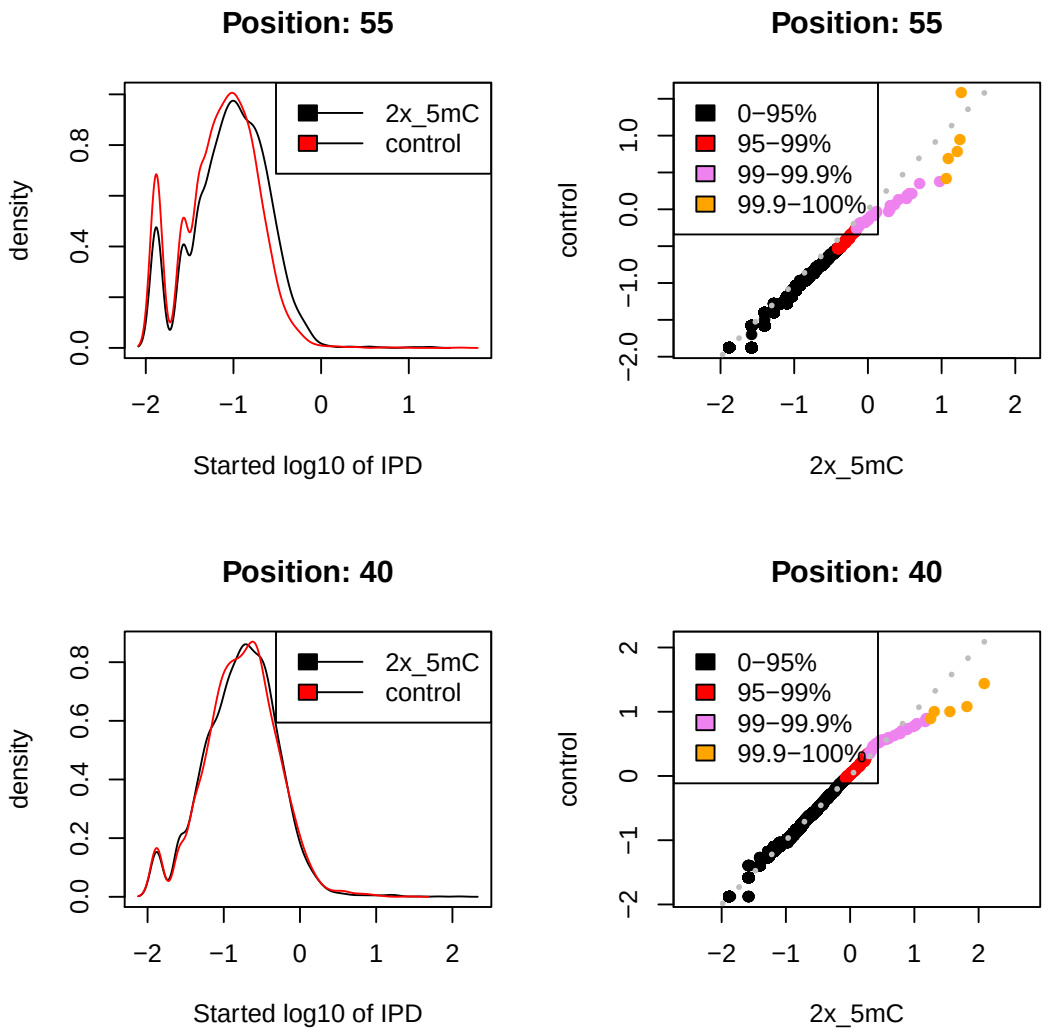

Figure 14: *Density at position and Strand* - Here we plot the IPD distribution for both a trully modified site as well as a non-modified site. We can see that the effect on IPD is much smaller when the modification is a methyl-C indicating that we will need larger sample sizes to determine a true effect.

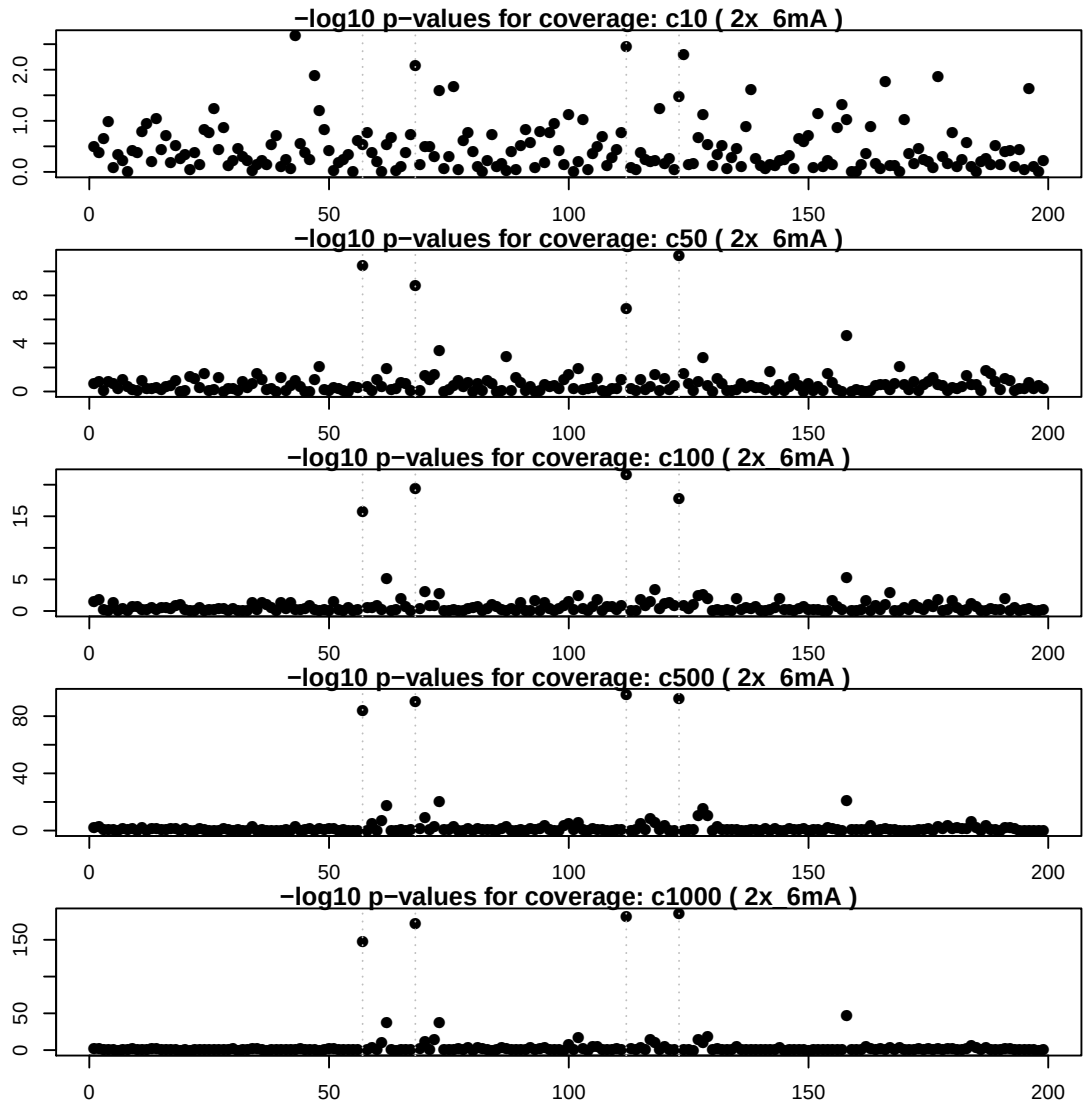

Figure 15: Here we plot the $-\log_{10}$ p-values from the Wilcox test for increasing levels of coverage for the 2x_6mA modified template.

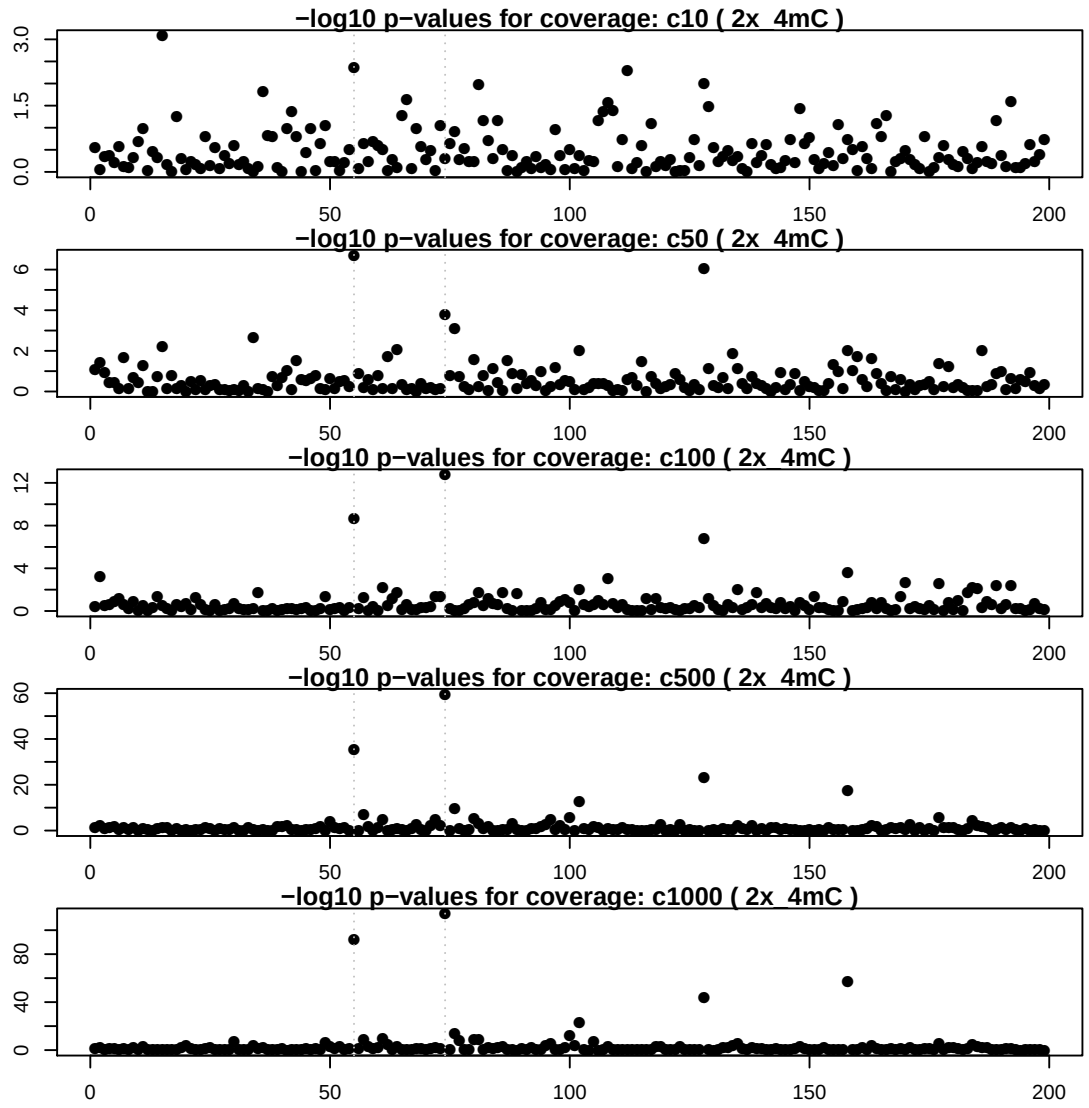

Figure 16: Here we plot the $-\log_{10}$ p-values from the Wilcox test for increasing levels of coverage for the 2x_4mC modified template.

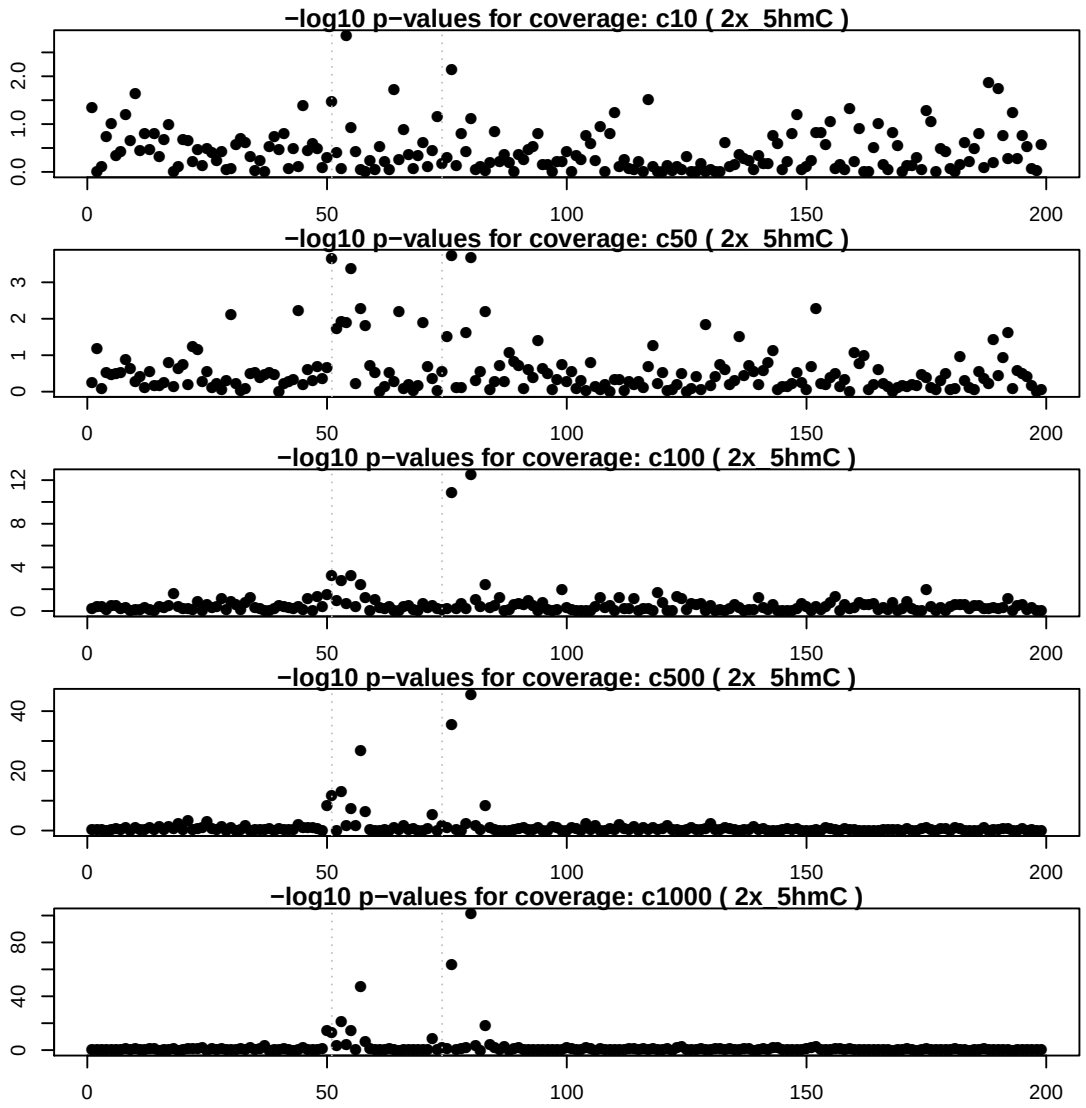

Figure 17: Here we plot the $-\log_{10}$ p-values from the Wilcox test for increasing levels of coverage for the 2x_5hmC modified template.

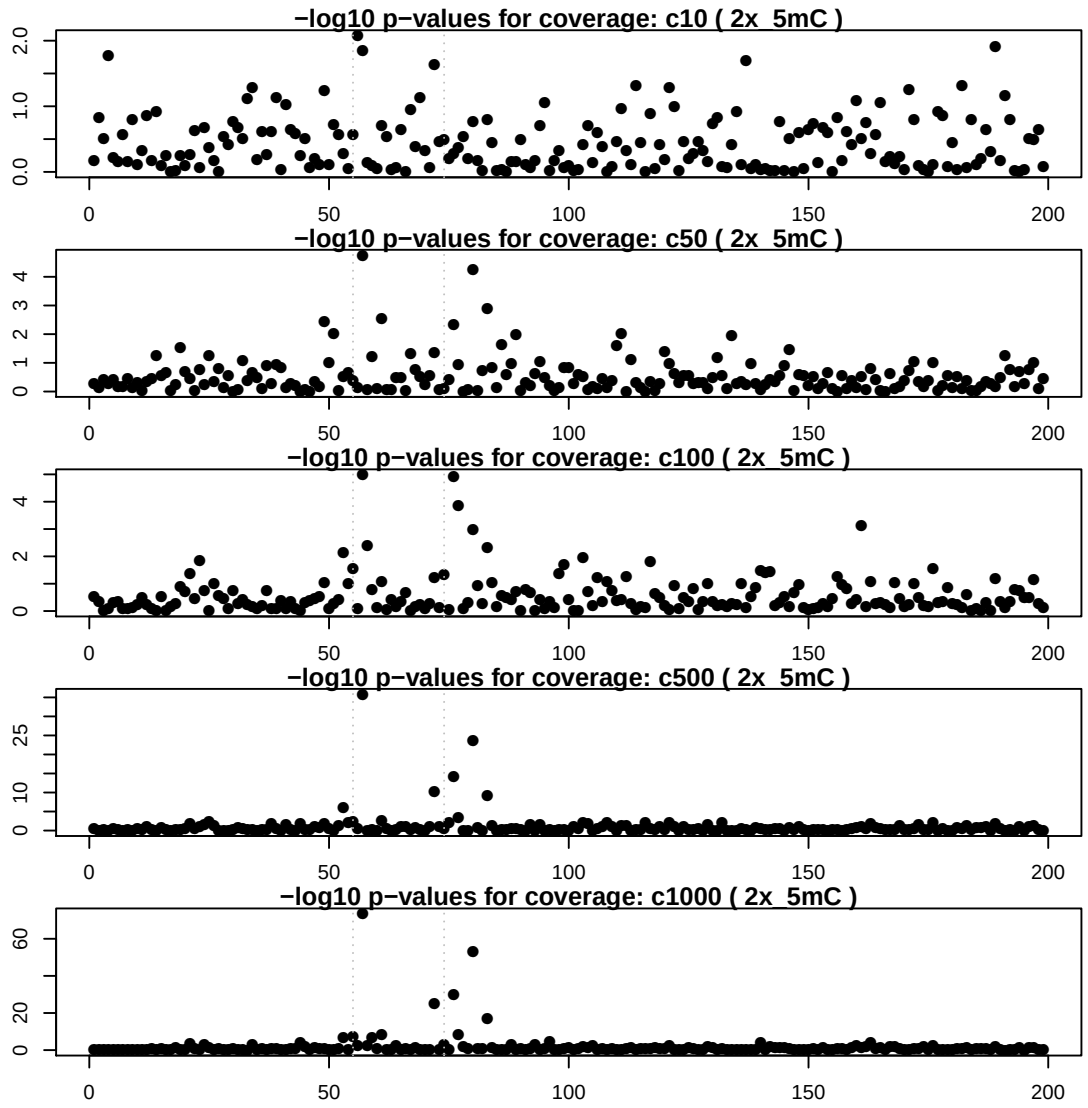

Figure 18: Here we plot the $-\log_{10}$ p-values from the Wilcox test for increasing levels of coverage for the 2x_5mC modified template.

4.1 Evaluation of Different Testing Procedures

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 get a sense of our true-positive rate as well as our false-positive rate. 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 may take this into account.

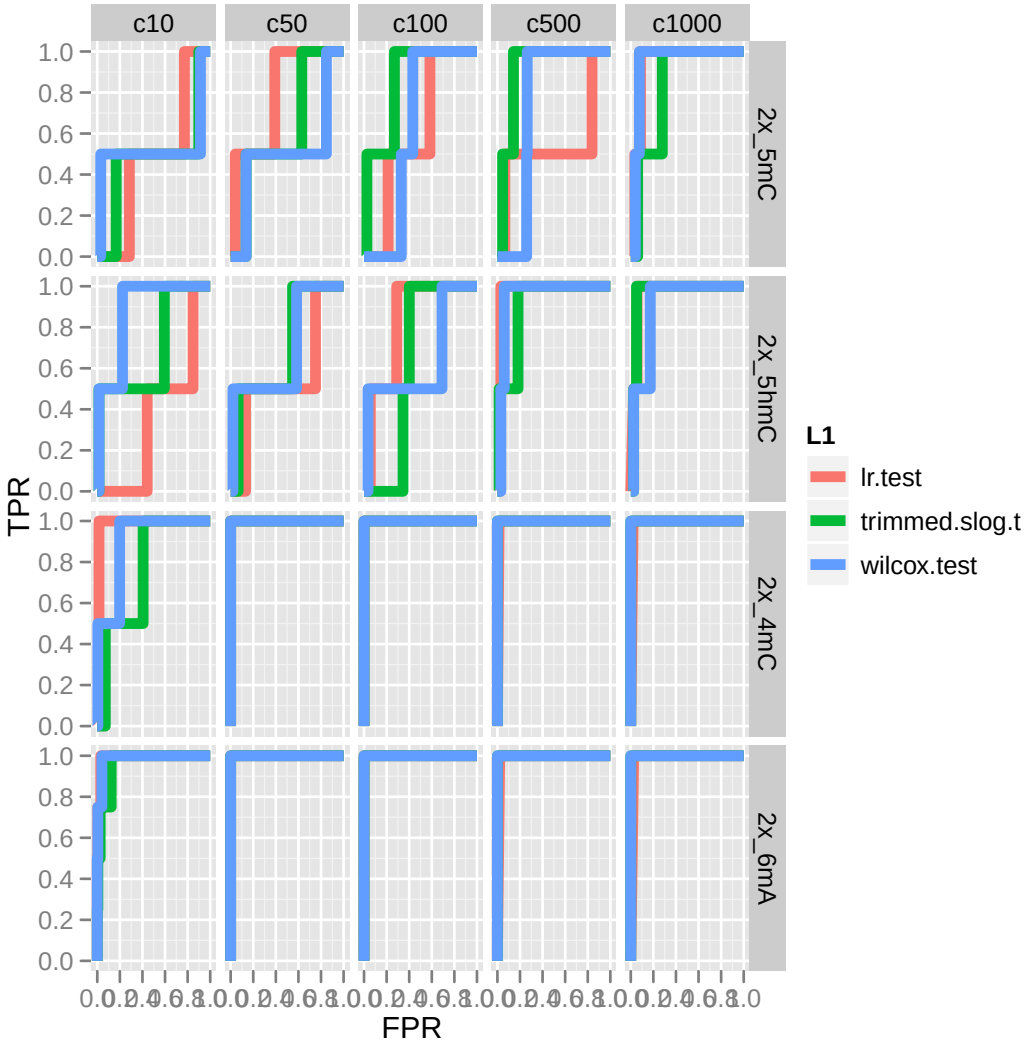

Figure 19: *ROC Curves* - Here we plot ROC curves for the three different testing procedures faceted by coverage and modification type. These curves demonstrate the differences in the magnitude of the modification effects. It is clear the 6mA is quite easy to detect, even at a relatively low level of coverage. However, it is equivalently clare that 5mC has a much smaller effect on the IPD distribution.

4.2 Statistical Testing in Lambda

As mentioned previously, a much more realistic dataset is the lambda dataset where we have 4 distinct conditions (Table 2). A reasonable test statistic for detecting base modification could be the ratio of means: $\mu_{modified}/\mu_{amplified}$, where μ could be the mean or median of the IPD values at a given position. Different applications and different levels of coverage might warrant different procedures.

```
> computeIPDRatio <- function(controlCmpH5, treatmentCmpH5, start,
+     end) {
+     control <- subset(getByTemplatePosition(controlCmpH5, idx = getReadsInRange(controlC
+         1, start, end)), read == ref & position >= start & position <=
+         end)
```

```
+     treatment <- subset(getByTemplatePosition(treatmentCmpH5,
+         idx = getReadsInRange(treatmentCmpH5, 1, start, end)),
+         read == ref & position >= start & position <= end)
+     tapply(treatment$elt, factor(treatment$position, start:end),
+         mean)/tapply(control$elt, factor(control$position, start:end),
+         mean)
+ }
> ipdRatios <- computeIPDRatio(cmpH5s[["6mA_dam-_native"]], cmpH5s[["6mA_dam+_native"]],
+     1000, 2000)
```

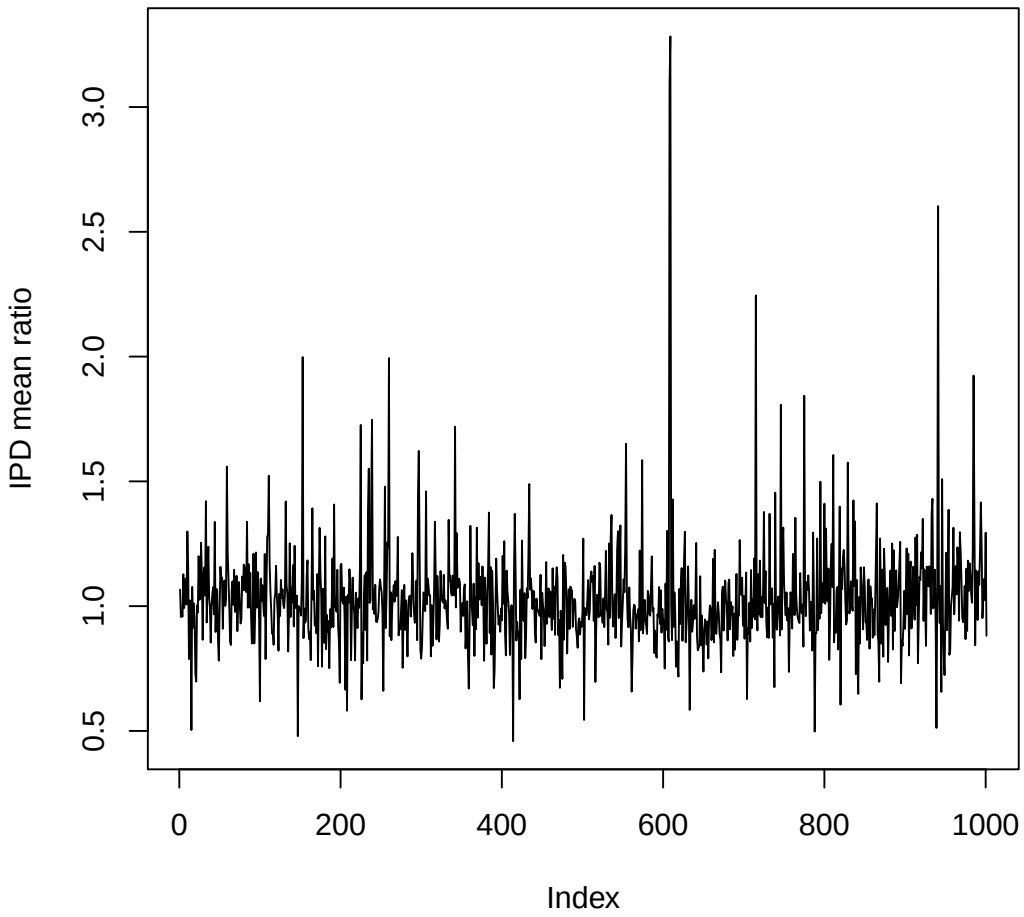

Figure 20: *IPD Ratio* - A statistic which is bounded between 0-Inf where large values indicate differences in means. Naturally, we need to associate confidence to these values.

4.3 Top Table

Using the functions described previously in the document we can easily write a procedure to construct a top table - that is - a table is a ranked data set with p-values.

```
> makeTopTable <- function(treatmentCmpH5, controlCmpH5, whichRef = 1,
+     ipdFetch = getIPD, myTest = function(a, b) {
+         t = t.test(a$elt, b$elt)
+         c(statistic = t$statistic, p.value = t$p.value)
+     }, ..., binSize = 10000, VERBOSE = TRUE, pctReads = 1) {
+     stopifnot(getRefName(treatmentCmpH5, whichRef) == getRefName(controlCmpH5,
+         whichRef))
+     refLength <- getRefLength(treatmentCmpH5, whichRef)
+     intervals <- seq(1, refLength, by = binSize)
+     iresults <- vector("list", length(intervals))
+     getData <- function(cmpH5, start, end) {
+         reads <- getReadsInRange(cmpH5, whichRef, start, end)
+         reads <- sample(reads, size = pctReads * length(reads))
+         getByTemplatePosition(cmpH5, idx = reads, f = ipdFetch)
+     }
+     for (i in 2:length(intervals)) {
+         if (VERBOSE)
+             cat("Processing interval:", i - 1, "\n")
+         start <- intervals[i - 1]
+         end <- intervals[i]
+         control <- getData(controlCmpH5, start, end)
+         treatment <- getData(treatmentCmpH5, start, end)
+         d <- mapply(FUN = function(ci, ti) {
+             myTest(treatment[ti, ], control[ci, ])
+         }, split(1:nrow(control), factor(control$position, start:end)),
+             split(1:nrow(treatment), factor(treatment$position,
+                 start:end)), SIMPLIFY = FALSE)
+         d <- do.call(rbind, d)
+         d <- as.data.frame(d)
+         d$position <- start:end
+         iresults[[i]] <- d
+     }
+     do.call(rbind, iresults)
+ }
```

4.4 Advanced Statistical Methods

- empirical Bayes procedures
- control-free procedures
- signature specific methods

5 Conclusion