Perm-seq pipeline for Multi-read Analysis of ChIP-seq Data

Xin Zeng[1], Ye Zheng*[1], M. Constanza Rojo-Alfaro*[1], and Sündüz Keleş[1,2]

[1]Department of Statistics, University of Wisconsin-Madison.
[2]Department of Biostatistics and Medical Informatics, University of Wisconsin-Madison

June 2015

Contents

1 Overview

The overall pipeline allocates multi-reads which are reads that map to multiple positions in the genome and are typically discarded or utilized in a sub-optimal way by standard alignment of ChIP-seq data.
This package can be used to align ChIP-seq data of transcription factors and Histone modifications with punctuated signals (e.g., H3K4me3) with or without prior information.

2 Prerequisites

The overall pipeline requires R, C++ and Perl. Users need to download and install Bowtie (http://sourceforge.net/projects/bowtie-bio/files/bowtie/), CSEM (CSEM >= 2.3: http://deweylab.biostat.wisc.edu/csem/) and R package *permseq* (https://github.com/keleslab/permseq).

```
library(permseq)
```

Besides, to run the *permseq*, users should also install Perl and Perl module, Statistics:Descriptive, which will be called by *permseq*

After downloading Bowtie, users also need create the Bowtie indices (or download them).

2.1 Bowtie index

1. Download the Bowtie indices for your genome: The Bowtie website provides pre-built indices for a number of commonly-used genomes. For example, you can download the index for *hg19* genome (UCSC hg19) from the Bowtie website.
2. Build new Bowtie indices: To create your own (*hg19*, for example), you can run:
   `bowtie-build chr19.fa,chr22.fa hg19`
   The first argument "chr19.fa,chr22.fa" is a comma-separated list of FASTA files containing the reference sequences to be aligned (only human chromosome 19 and chromosome 22 in this case) and `hg19` is the basename of the index files to be created.

Please refer to Bowtie documentation for more options and details.

3 permseqExample Package

A *permseqExample* package is available with demo data so the users can test *permseq* in a reduced period of time using the given script and compare results with the R images included. Bowtie indices are also available. We first created a sample reduced genome using 1% of the human chromosome 22 from the *hg19* assembly. The original ChIP, DNase, and Histone ChIP-seq fastq files were downsampled to contain reads that map to our sample reduced genome. These datasets are:

1. *ChIP_partial.fastq*: Atf3 ChIP-seq fastq file from GM12878.
2. *DNase_partial.fastq*: DNase-seq replicate 1 fastq file from GM12878.
3. *H2az_partial.fastq*: H2az Histone ChIP-seq fastq file from GM12878.
4. *H3k4me1_partial.fastq*: H3kme1 Histone ChIP-seq fastq file from GM12878.

We will use these data in the rest of this vignette. We refer the reader to the *permseqExample* package help manual for more usage details.

4 Workflow

The *permseq* package consists of prior module generation, prior generation, and multi-read allocation. Prior module generation consists of processing of the data for fitting the prior model and it is separated into different functions depending on the type of prior data that the user wants to use.
In summary, the three steps are:

1. Creating prior module
   1.1 `priorProcess`: Available prior data is either only DNase-seq or both DNase-seq and Histone ChIP-seq.
   1.2 `priorHistone_initial` and `priorHistone_multi`: Available prior data is based on Histone ChIP-seq.
2. Generating priors: `priorGenerate`
3. Allocate multi-reads: `readAllocate`

Table 1 summarizes relevant sections of this vignette based on user input data.

Input Data	Functions	Section
ChIP+DNase	`priorProcess`	4.1.1
	`priorGenerate`	4.2
	`readAllocate`	4.3
ChIP+DNase+Histone	`priorProcess`	4.1.2
	`priorGenerate`	4.2
	`readAllocate`	4.3
ChIP+Histone	`priorHistone_init`	4.1.3
	`priorHistone_multi`	4.1.3
	`priorGenerate`	4.2
	`readAllocate`	4.3
ChIP	`readAllocate`	6.1

Table 1: Input data and functions

For the enumerated cases in Table 1, each function will take as input the result of the previous one. Therefore, these functions should be executed in the same order as shown in Table 1.
In each step, *permseq* will return an S4 *Prior* class, which is described in section 5. This S4 class summarizes the analysis information and it will be updated in each step.

To align different sequencing data (ChIP-seq, Input ChIP-seq, DNase-seq, and Histone ChIP-seq), *permseq* utilizes the following Bowtie parameters:

```
bowtie -q -v 2 -a -m 99 -p 8 --sam hg19 data.fastq data.sam
```

The package allows the user to change the `-v`, `-m`, and `-p` options in Bowtie, which are set to 2 (alignments may have no more than 2 mismatches), 99 (suppress alignments for a particular read if more than 99 reportable alignments exist for it) and 8 (launch 8 parallel search threads) by default, respectively.
The other arguments are:

1. `hg19`: Name of the Bowtie index obtained as described in section 2.1, given as an argument in *permseq*.
2. `data.fastq`: A comma-separated list of file names containing the reads in FASTQ format, given as an argument in *permseq*.
3. `data.sam`: Aligned file in SAM format (output file).

The main structure of the algorithm with DNase-seq and Histone ChIP-seq driven priors is summarized in Figure 1.

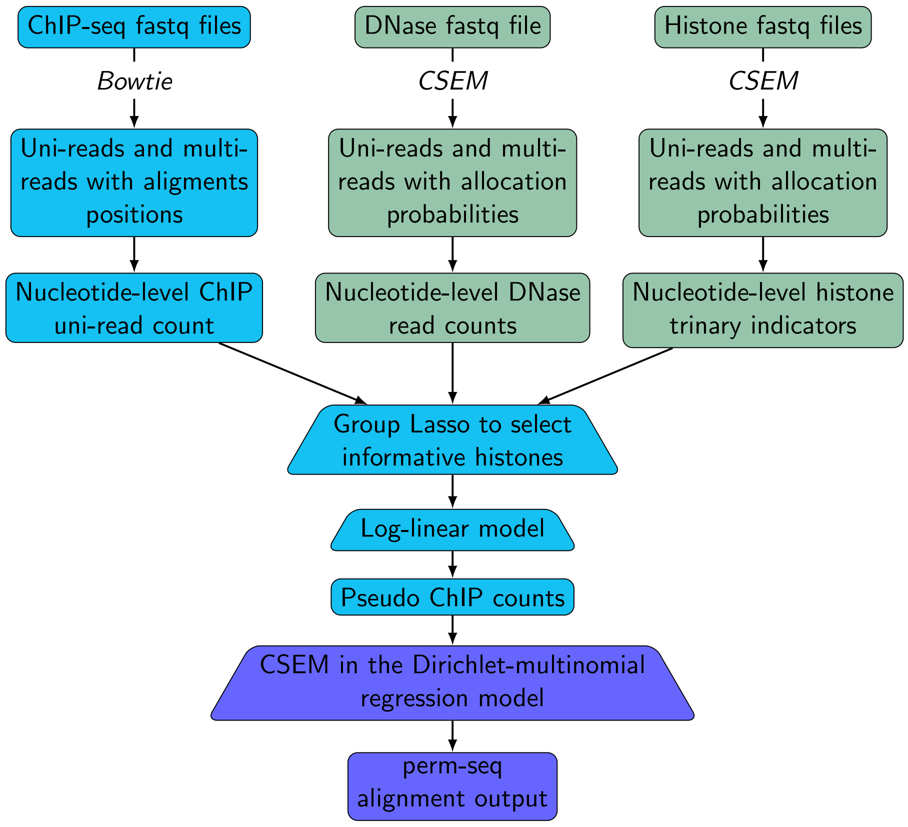

ChIP-seq fastq files
DNase fastq file
Histone fastq files
Bowtie
CSEM
CSEM
Uni-reads and multi-reads with aligments positions
Uni-reads and multi-reads with allocation probabilities
Uni-reads and multi-reads with allocation probabilities
Nucleotide-level ChIP uni-read count
Nucleotide-level DNase read counts
Nucleotide-level histone trinary indicators
Group Lasso to select informative histones
Log-linear model
Pseudo ChIP counts
CSEM in the Dirichlet-multinomial regression model
perm-seq alignment output

Figure 1: Rectangle shapes indicate data (input/output) and trapezoids indicate models/methods.

In the Perm-seq model, Histone ChIP-seq read counts are converted into trinary indicators γ according to 90$-th$ and 99$-th$ percentile of Histone ChIP-seq read count distribution:

$$\gamma_i = \begin{cases} 0 & \text{if histone read counts} \leq 90 - th, \\ 1 & \text{if } 90 - th < \text{histone read counts} \leq 99 - th, \\ 2 & \text{if histone read counts} > 99 - th, \end{cases}$$

for $i = 1, \ldots, h$, where h is the number of Histone ChIP-seq data sets given by the user.

In what follows, we will use the default options for the function arguments: `fragL`, `AllocThres`, `capping`, `vBowtie`, `mBowtie`, `pBowtie`, and `save.files`. For more options, please refer to section 6.

4.1 Creating prior module based on multiple data sets

This section describes how to create prior module from your DNase-seq and/or Histone ChIP-Seq data. Creating prior module involves the following tasks.

1. Align DNase-seq and Histone ChIP-seq reads with Bowtie. SAM files are generated as a result for each input dataset.
2. Allocate multi-reads using CSEM. BAM files are generated as a result for each input dataset.
3. BAM output from CSEM is converted to BED format and a subset of reads with high allocation score is selected to avoid generation of spurious signals. In the applications presented in
4. Generate nucleotide-level read counts for the DNase-seq and Histone ChIP-seq by essentially counting the number of reads overlapping each nucleotide.
5. Partition the genome into multiple segments (based on the DNase counts and/or trinary Histone ChIP-Seq indicators).

4.1.1 Creating prior module with only DNase-seq data

```
priorProcess_DNase <- priorProcess(dnaseFile="DNase_partial.fastq", histoneFile=NULL,
 dnaseName="DNase_partial", histoneName=NULL, fragL=200, AllocThres=900,
 chrList="chr22_partial", capping=0, outfileLoc="./", outfile="only_dnase",
 bowtieIndex="./chr22_partial", csemDir="./csem-2.3", bowtieDir="./bowtie-1.1.1",
 vBowtie=2, mBowtie=99, pBowtie=8, save.files=TRUE)
```

This will create a *Prior* S4 object which contains the main information that needs to be inherited to continue the analysis. For more details about the *Prior* object, refer to section 4 or the R help manual of the package. The main slots of the Prior object type output are:

1. **dnaseKnots**: Vector of the knot points for the B-spline functions used in the fit.
   ```
   > priorProcess_DNase@dnaseKnots
   [1]  594 2029 3440
   ```
   They are the 90, 99, and 99.9th percentiles of DNase-seq read count distribution.
2. **dnaseThres**: Vector of DNase-seq groups. Groups are constructed based on nucleotide level counts, except the first group that will include counts equal to 0, 1 and 2, and the last group that will include counts larger than the 99.9th percentile of the read count distribution.
   ```
   > priorProcess_DNase@dnaseThres
   [1] 2 3 4 5 6 ... 3435 3436 3438 3440 3794
   ```
   DNase-seq counts smaller or equal to 2 are clustered into one group; counts equal to 3 are clustered into another group and so on. Positions with counts larger than 3440 are clustered into one group. Specifically, the corresponding count is set at the 99.95th percentile of the read count distribution for model fitting, e.g., 3794 in this specific case.
3. **posLoc_bychr**: List by chromosome.
   Each element lists a file named chr$i$_dnase_one_dnase_positions_cluster.txt (in the example for this vignette, this list is of length one since we are only using part of chr 22). This file encodes which segments of the genome are in which prior group, based on DNase-seq read counts. Each line corresponds to one chromosome, starting with the chromosome ID, followed by pairs of segment lengths and group IDs:
   ```
   chr22_partial    5    574    1    575    2    576 ...
   ```
   The first segment of chr22_partial is in group 574 according to `dnaseThres`. In other words, DNase-seq counts equal to 574 are clustered into the same group, and the number of positions that belongs to

group 574 is 5.

4.1.2 Creating prior module with DNase-seq and Histone ChIP-seq data

```
priorProcess_DNaseHistone <- priorProcess(dnaseFile="DNase_partial.fastq",
 histoneFile=c("H2az_partial.fastq", "H3k04me1_partial.fastq"),
 dnaseName="DNase_partial", histoneName=c("H2az_partial", "H3k04me1_partial"),
 fragL=200, AllocThres=900, chrList="chr22_partial", capping=0, outfileLoc="./",
 outfile="dnase_histone", bowtieIndex="./chr22_partial", csemDir="./csem-2.3",
 bowtieDir="./bowtie-1.1.1", vBowtie=2, mBowtie=99, pBowtie=8, save.files=TRUE)
```

This function will create a *Prior* S4 object which contains the main information that needs to be inherited to continue the analysis.
The main slot that is different from the ones described in section 4.1.1 is **posLoc_bychr**, which is a list by chromosome.
Similarly, each element points to a file chr$i$_dnase_dnase_histone_positions_cluster.txt. This file encodes which segments of the genome are in which group, based on DNase-seq counts and Histone trinary indicators. Each line corresponds to one chromosome, starting with the chromosome ID, followed by pairs of segment lengths and group IDs:

```
chr22_partial    5    574_0_1    1    575_0_1    2    576_0_1    1 ...
```

The first segment of chr22_partial is in group 574_0_1. Each group is based on the DNase-seq reads count and the histones trinary indicator. Positions within group 574_0_1 have 574 DNase-seq reads aligned to them, and H2az histone ChIP-seq reads counts at these positions are smaller than the 90th percetile resulting in the number 0̈here . Besides, H3k4me1 histone ChIP-seq reads count is larger than the 90th percentile and smaller or equal than the 99th percentile. Thus the indicator for it is 1̈. Moreover, the number of positions that belong to group 574_0_1 is 5.

4.1.3 Creating prior module with only Histone ChIP-seq data

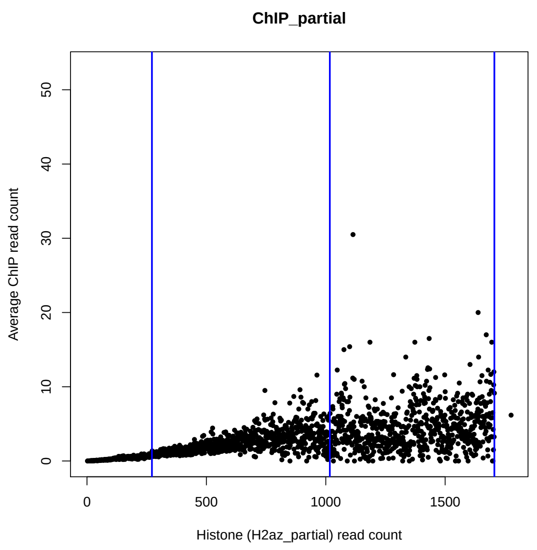
Utilizing only histone ChIP-seq data requires the following two steps.

1. `priorHistone_init`: This function processes histone ChIP-seq datasets and generates plots depicting marginal relationship of each histone ChIP-seq with the ChIP-seq data to be analyzed. In these marginal plots, histone ChIP-seq counts are not converted into trinary counts. Instead, they are treated as numerical variables as for the DNase-seq case. Based on these plots, the user specifies the histone ChIP-seq data to be used for prior construction. We recommend to choose the histone that seems most correlated with the ChIP read.
2. `priorHistone_multi`: This function generates prior from the selected histone ChIP-seq data using it as a numerical variable and applying the prior construction framework of Section 4.1.1.

For example, we can utilize two histone ChIP-seq datasets, namely, H2az and K3k04me1 as follows:

```
priorHistone_init <- priorHistone_init(histoneFile=c("H2az_partial.fastq",
 "H3k04me1_partial.fastq"), histoneName=c("H2az_partial", "H3k04me1_partial"),
 fragL=200, AllocThres=900, chrList="chr22_partial", capping=0, outfileLoc="./",
 chipFile="ChIP_partial.fastq", bowtieIndex="./chr22_partial", csemDir="./csem-2.3",
 bowtieDir="./bowtie-1.1.1", vBowtie=2, mBowtie=99, pBowtie=8, save.files=TRUE)
```

This results in the marginal plots in Figure 2.



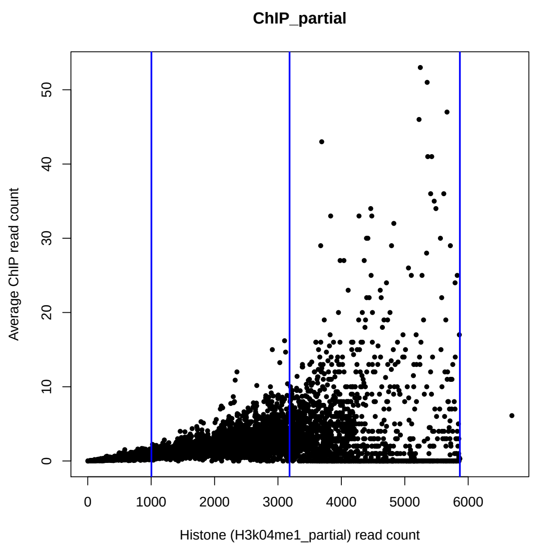


Figure 2: Output plots from priorHistone_init function. These plots depict the relationships between ChIP counts (the main dataset for which we are building a prior for) and the Histone ChIP-seq counts.

To choose and utilize H2az without DNase-seq, we set `dnaseIndex=1` in `priorHistone_multi` function. We note that since these marginal plots were constructed only using a very small subset of the actual datasets, the do not exhibit as strong relationship as the full data

```
priorHistone_multi <- priorHistone_multi(priorHistone_init, dnaseIndex=1,
 outfileLoc="./", outfile="twodata_histone")
```

This function creates a *Prior* S4 object which contains the main information that needs to be inherited to continue the analysis.
The main slot that is different from the ones described in sections 4.1.1 and 4.1.2 is **dnaseName**. Previously, **dnaseName** was the name of the DNase data but when DNase data is not available, it will be the name of the Histone ChIP-seq data used as DNase:

```
> priorHistone_multi@dnaseName
[1] "H2az_partial"
```

In this case, knots (**dnaseKnots**) and groups (**dnaseThres**) are calculated based on the given Histone ChIP-seq data.

4.2 Generating priors

Using the *Prior* object created in section 4.1, the user can run the `priorGenerate` function of *permseq* to generate priors.
The internal structure of the function is:

1. Align ChIP-seq reads using Bowtie. SAM files are obtained as a result for each ChIP data.
2. Generate the corresponding BED file containing uni-reads from the SAM formatted alignment files.

3. Average over the uni-reads ChIP-seq read counts across positions within the same group, based on their read count.
4. Select important Histone ChIP-seq datasets by Group Lasso and fit the model (if Histone ChIP-seq datasets are used). `maxHistone` is the maximum number of allowed Histone ChIP-seq datasets in the model and is suggested to be smaller than 6 in order to avoid too many groups with very few read counts in each position.
5. Generate as many prior files named "prior_chipFile.txt" as ChIP files given.

For example, we can use the prior module from section 4.1.1 and one ChIP_partial dataset allowing maximum 6 Histone ChIP-seq datasets.

```
priorGenerate_DNase <- priorGenerate(priorProcess_DNase,
 chipFile="ChIP_partial.fastq", maxHistone=6, outfileLoc="./")
```

This function updates the *Prior* object created in section 4.1.
It is important to emphasize that `chipFile` can be a vector of fastq files with different replicates but they should all be from the same transcription factor of the same experiment.
The main slot that is updated in this step is **prior**, which is a list of prior files, prior_chip$i$.txt, for each ChIP-seq datasets. In our case, we only use one ChIP_partial.fastq file thus only one prior file is created. The first line of the prior file starts with the number of total groups, followed by the prior values for each group. Other lines record the number of positions in each group and the corresponding groups (as in the posLoc_bychr slot):

```
3285 1.00472158441105 1.0048264374093 1.00487973388612 1.00493361889526 ...
```

In our example, there are 3285 groups, followed by their prior values for each group.

4.3 Allocating multi-reads

Using the *Prior* object created in section 4.2, the user can run the `readAllocate` function of *permseq* to allocate the multi-reads by the prior built in the previous steps and get the final results.
Allocating the multi-reads of ChIP-seq involves the following steps.

1. Allocate ChIP-seq multi-reads based the prior probability using CSEM. BAM files are generated for each ChIP data.
2. Convert BAM file into other format specified under the `outputFormat` argument. We can select the format to be tagAlign or BED.
3. Select reads from previous file with allocation score higher than `ChipThres` .

For example, we choose the output format of tagAlign and use the object created in section 4.2:

```
readAllocate_DNase <- readAllocate(priorGenerate_DNase, outfileLoc="./",
 outputFormat="tagAlign", ChipThres=500, chipFile=NULL, bowtieIndex=NULL,
 csemDir=NULL, bowtieDir=NULL, vBowtie=NULL, mBowtie=NULL, pBowtie=NULL, fragL=NULL)
```

This function creates a final *Prior* S4 object.
If no prior is provided, *readAllocate* will be reduced to CSEM alignment. Users can refer to section 6.1 for allocating multi-reads without prior information (DNase and/or Histone ChIP-seq).

5 Prior Class

Prior is an S4 class containing information for ChIP-seq alignment.
It will be created in the first step of *permseq*, section 4.1, and then updated with new information when prior is built, section 4.2, and reads are allocated, section 4.3.

5.1 names()

`names()` is equivalent to `slotNames()`. It will return the names of all the slots in a *Prior* class object.
For example:

```
> names(readAllocate_Dnase)
 [1] "dnaseName"    "dnaseAlign"   "dnaseKnots"   "dnaseThres"   "posLoc_bychr"
 [6] "dnaseHistone" "histoneName"  "histoneNum"   "histoneAlign" "histoneGrpL"
[11] "chipName"     "chipNum"      "chipAlign"    "chipSAM"      "chipAllocate"
[16] "chipUni"      "chipFormat"   "dataNum"      "fragL"        "bowtieInfo"
[21] "csemDir"      "outfileLoc"   "prior"        "chrom.ref"
```

For a detailed description of the slots, please refer to the R help manual of the package.

5.2 print()

`print()` will print all the slots of *Piror* class. The screen output will be divided into four categories:

- DNase-seq Related Information.
- Histone ChIP-seq Related Information.
- ChIP-seq Related Information.
- Other Parameter and Directory Information.

5.3 show()

`show()` will print only selected core information of *Prior* class. The screen output will be divided into four categories:

- DNase-seq Minimal Information: `dnaseName`, `dnaseThres`, `dnaseKnots` and `dnaseHistone`.
- Histone ChIP-seq Minimal Information: `histoneName` and `histoneGrpL`.
- ChIP-seq Minimal Information: `chipName` and `chipFormat`.
- Other Information: `dataNum`, `prior` and `chrom.ref`.

5.4 summary()

Given a *Prior* object, `summary()` will return the alignment details of DNase-seq, Histone ChIP-seq and/or ChIP-seq files from bowtie.
For example, in section 4.3 we only use DNase-seq to build the prior on which we allocate the ChIP-seq reads. The summary for the resulting *Prior* object from readAllocate will be:

```
> summary(readAllocate_DNase)
**********************************************************************************
Alignment Summary for DNase-seq Dataset:
**********************************************************************************
                                                              Alignment Information
Number of reads processed:                                                  1436820
Number of reads with at least one reported alignment:                       1436820
Percentage of reads with at least one reported alignment(%):                    100
Number of reads failed to align:                                                  0
Percentage of reads failed to align(%):                                           0
Total number of alignment reported:                                         8970293
**********************************************************************************
Alignment Summary for ChIP-seq Dataset chip_partial:
**********************************************************************************
                                                              Alignment Information
Number of reads processed:                                                   504409
Number of reads with at least one reported alignment:                        504409
Percentage of reads with at least one reported alignment(%):                    100
Number of reads failed to align:                                                  0
Percentage of reads failed to align(%):                                           0
Total number of alignment reported:                                         3287483
```

We use the demo data that includes only the reads that will align to the partial chr22 which explain why bowtie can achieve 100% alignment here.

5.5 plot()

Given a *Prior* object from priorHistone_multi, priorGenerate or readAllocate, it will plot DNase read counts versus averaged ChIP read counts using aggregated strategy with three vertical lines indicating the DNase-seq knot points at the 90, 99 and 99.9th percentiles of the DNase-seq read count distribution. The aggragated strategy here means that genomic position with the same DNase-seq read counts are grouped together and ChIP-seq read counts are averaged within each group. The resulting plot is the one in Figure 3.

6 More Options

6.1 Allocate multi-reads without using - -prior option in CSEM

permseq also allows the users to allocate multi-reads without using any prior information based on DNase-seq and/or Histone ChIP-seq datasets.
In order to do this, run readAllocate directly and set object=NULL:

```
result <- readAllocate(object=NULL, outfileLoc="./", outputFormat=NULL,
 chipFile="ChIP_partial.fastq", bowtieIndex="./chr22_partial",
 csemDir="./csem-2.3", bowtieDir="./bowtie-1.1.1", vBowtie=2,
 mBowtie=99, pBowtie=8, fragL=200)
```

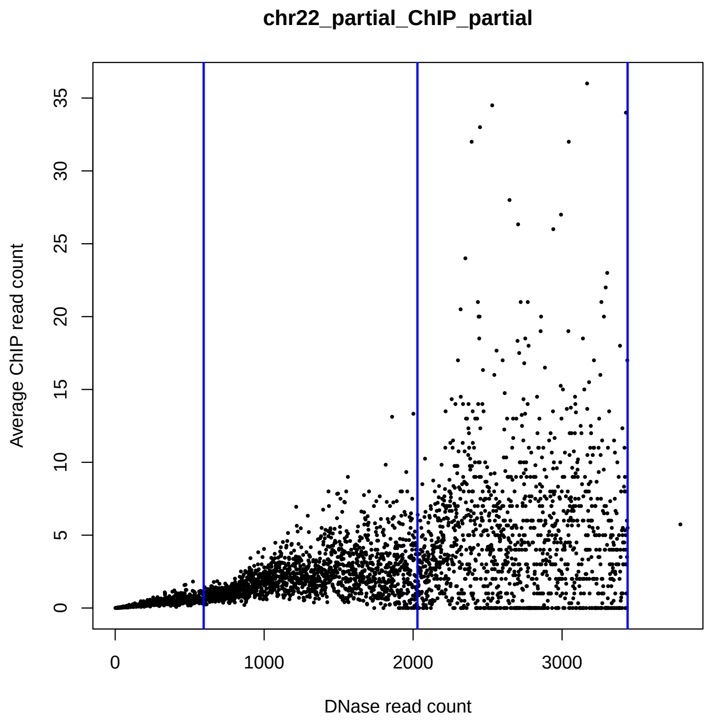


Figure 3: `plot(readAllocate_DNase)`

If no *Prior* object is provided, it is necessary to give the parameter value to *bowtieIndex*, *csemDir*, *bowtieDir*, *vBowtie*, *mBowtie*, *pBowtie*, and *fragL* which were set as `NULL` in section 4.3.
We will get BAM file of the allocated read named "ChIP_partial_permseq.bam" as the final *permseq* alignment results. Besides, we will also get tagAlign or BED files depending on the option set in `outputFormat`.

6.2 Save intermediate files

The option `save.files` in `priorProcess` and `priorHistone_init` will allow the users to store intermediate files or not. The default is set as `TRUE` which means saving the intermediate files.
The *permseq* package generates different folders for each given data (DNase-seq and/or Histone ChIP-seq) which can take a lot of space of your disk. The folders store the processed files from `priorProcess` or `priorHistone_init`. They are created in order to generate the file that encodes which segments of the genome are in which group that will be used to produce the prior file for CSEM.
Some of these files are:

- SAM file with aligned reads.
- BAM, SORTED.BAM and SORTED.BAM.BAI from the SAM file.
- BED file with allocated multi-reads.
- BED file with selected reads (scores higher than `AllocThres`).
- BED files with selected scores split by chromosomes used (given in `chromList`).

By setting `save.files=FALSE`, *permseq* will remove these intermediate files which will not be used in the latter proceduresr leaving only the BED file with allocated multi-reads and the file under the `posLoc_bychr`

slot which is necessary in the next step.
The *Prior* object generated will be the same as in section 4.1.

6.3 BAM or BED files directly

If the users have already processed the DNase and/or Histone ChIP-seq datasets and the BAM files with aligned reads or the BED files with allocated multi-reads are already available, they can be directly given as an argument to the `dnaseFile` and `histoneFile` of `priorProcess` and `priorHistone_init` instead of the fastq files. Using the aligned BAM or BED files, considerably computational time will be saved.
The *Prior* object generated will be the same as in section 4.1.

6.4 Chromosome list

In *permseq*, it is important that the users provide the *chrList* argument value consistent with the chromosome(s) name(s) in the corresponding .fa file(s). Default is set as `NULL` and *permseq* will extract the chromosome names from processed files. For instance, it will use SAM file if DNase input file is in fastq format or BED file if DNase input file is in BAM or BED format.
It is recommendable to give the chromosome list since it will accelerate the process but be sure to make it consistent with the .fa files.
The *Prior* object generated will be the same as in section 4.1.

7 SessionInfo

```
toLatex(sessionInfo())
```

- R version 3.1.1 (2014-07-10), `x86_64-redhat-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=en_US.UTF-8`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: knitr 1.10.5
- Loaded via a namespace (and not attached): BiocStyle 1.4.1, evaluate 0.7, formatR 1.2, highr 0.5, magrittr 1.5, stringi 0.5-5, stringr 1.0.0, tools 3.1.1

References

[1] Chung D, Kuan PF, Li B, Sanalkumar R, Liang K, Bresnick E, Dewey C and Keleş S (2011) *Discovering Transcription Factor Binding Sites in Highly Repetitive Regions of Genomes with Multi-Read Analysis of ChIP-Seq Data*. PLoS Computational Biology.

[2] Zeng X, Li B, Welch R, Dewey C and Keleş S *Perm-seq: Mapping protein-DNA interactions in segmental duplication and highly repetitive regions of genomes with prior-enhanced read mapping*. Under Review.