Mixture modeling for genome-wide localization of transcription factors

Sündüz Keleş[1,2] and Heejung Shim[1]
[1] Department of Statistics
[2] Department of Biostatistics & Medical Informatics
University of Wisconsin, Madison

December 22, 2005

1 Introduction to the HGMM package

ChIP-chip experiments enable researchers to the investigate interactions of DNA binding proteins and DNA on the genome scale. **HGMM** package implements the Hierarchical Gamma Mixture Model (HGMM) approach proposed by [1] for analyzing data from these experiments. This vignette provides a brief overview of the functions in the **HGMM** package with examples. We refer to [1] for methodological details. The data used for illustration of the package functionality are from ChIP-chip experiments of [2] for the transcription factor p53 on human chromosomes 21 and 22. In [2], two types of controls are provided, we utilize the antiGST control here.

The package can be loaded with the command

> *library(HGMM)*

The followings are the function in **HGMM**.

preprocess	Performs log2 transformation, quantile normalization, and median scaling of the raw data.
partition	Partitions a given genomic region into several shorter genomic regions.
HGMM	Implements the EM algorithm for the Hierarchical Gamma Mixture Model.
fdrHGMM	Thresholds the posterior probabilities controlling the false discovery rate with a direct posterior probability approach to extract bound regions.
combinePK	Combines peaks that are in close vicinity of each other.
CVplot	Checks the constant coefficient of variation assumption.
gObsCompPlot	Checks the gamma observation component assumption.
emfitHGMM	Class used to store outputs of the function HGMM.

HGMM is the main function of this package. Functions **preprocess** and **partition** are for pre-processing the raw data whereas **fdrHGMM** and **combinePK** are for post-processing. Functions **CVplot** and **gObsCompPlot** are for checking the model assumptions through diagnostic plots.

2 Loading the data

There isn't yet a standardized format for storing data from ChIP-chip experiments. For the time being, we assume that the user provides (1) a data matrix for treatment observations, where rows are different probes and columns are replicated experiments; (2) similarly a data matrix for control observations; (3) a vector containing genomic locations of the probes. If the data are from different chromosomes, we suggest to analyze them separately for computational reasons. See the Appendix on a note for dealing with lower level data files including the CEL and BPMAP files.

For illustration purposes, we choose a subset of the p53 ChIP-chip data from [2]. These were extracted after our analysis of the whole data making sure that we have some peak (bound) and some non-peak (unbound) regions in the subset. The data are stored in **hgmmdata** and can be loaded by

```
> data(hgmmdata)
```

Here, the matrices `xt` and `xc` contain treatment and control measurements, respectively, and `locs` is a vector containing the actual genomic locations of the probes (these numbers indicate the start position of the probes by design). There are a total of 2117 probes with $n = 6$ treatment and $m = 6$ control replicate observations each. Here are the records for the first five probes:

```
> xt[1:5, ]
```

```
         [,1]     [,2]     [,3]     [,4]     [,5]     [,6]
[1,] 5.239036 5.333938 5.609558 5.495411 5.323226 5.231377
[2,] 5.608889 5.117671 4.826883 5.042655 4.793575 4.881180
[3,] 5.996320 5.613032 5.997989 5.146168 5.497901 5.194643
[4,] 4.524612 4.593985 4.642696 4.431579 4.486550 4.679036
[5,] 7.374372 7.410030 6.647467 6.928924 6.614949 6.883444
```

```
> xc[1:5, ]
```

```
         [,1]     [,2]     [,3]     [,4]     [,5]     [,6]
[1,] 5.036899 4.891130 5.193042 5.069125 5.283436 5.281643
[2,] 5.122285 4.782532 5.012070 5.130117 5.074619 4.817917
[3,] 5.546031 5.672227 5.749397 5.280796 5.473729 5.463238
[4,] 4.454650 5.580125 4.659843 4.440081 4.678203 4.841663
[5,] 7.873422 7.566493 6.508214 7.687277 7.183076 5.814985
```

```
> locs[1:5]
```

```
[1] 7777062 7777092 7777124 7777146 7777179
```

```
> dim(xt)
```

```
[1] 2117    6
```

```
> dim(xc)
```

```
[1] 2117    6
```

```
> length(locs)
```

```
[1] 2117
```

The above treatment and control values correspond to preprocessed version of the Perfect Match (PM) raw intensities. We provide further information on preprocessing in the next section.

3 Preprocessing: $\log_2$ transformation, quantile normalization and median scaling

preprocess function provides some simple ways of preprocessing ChIP-chip data. The perfect match raw intensity values are first $\log_2$ transformed, then quantile normalized [3] and median scaled. User can choose to normalize the data by other means and skip this step. As mentioned in the previous section, the p53 ChIP-chip data provided by this package are already preprocessed. For illustration purposes, here is a toy example:

```
> treatment = matrix(data = c(1 ,2 ,2 ,1 ,3 ,4 ,2 ,1 ,6 ,7 ,8 ,9), nc = 3, nr = 4)
> control = matrix(data = c(4, 5, 6, 5, 7, 8, 5, 8, 4, 9, 8, 1), nc = 3, nr = 4)
> treatment
```

```
     [,1] [,2] [,3]
[1,]    1    3    6
[2,]    2    4    7
[3,]    2    2    8
[4,]    1    1    9
```

```
> control
```

```
     [,1] [,2] [,3]
[1,]    4    7    4
[2,]    5    8    9
[3,]    6    5    8
[4,]    5    8    1
```

```
> library(affy)
> result = preprocess(treatment, control)
> result$xt
```

```
         [,1]     [,2]     [,3]
[1,] 1.634559 2.430827 1.430827
[2,] 2.634559 2.625815 1.838291
[3,] 2.634559 1.838291 2.430827
[4,] 1.634559 1.430827 2.625815
```

```
> result$xc
```

```
         [,1]     [,2]     [,3]
[1,] 1.198774 1.935785 1.935785
[2,] 2.134559 2.333333 2.477653
[3,] 2.676427 1.000000 2.333333
[4,] 2.134559 2.333333 1.000000
```

4 Model diagnosis: checking the model assumptions through diagnostic plots

HGMM package is based on a parametric model for the ChIP-chip data, therefore model assumptions need to be satisfied for correct statistical inference. Two major assumptions of the model are constant coefficient of variation and gamma observation component assumption. The function **CVplot** provides a diagnostic plot for checking the first assumption. It plots coefficient of variation of the probes versus their sample means and fits a smooth line to the resulting scatter plot. If the smooth line is approximately flat, this indicates that there is no obvious violation of this assumption. Figures 1 and 2 display such plots for the treatment and control observations, respectively. They are obtained by the following commands:

```
> CVplot(xt)
> CVplot(xc)
```

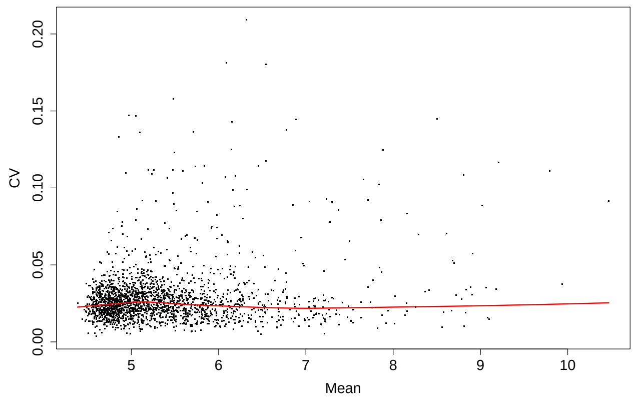

Figure 1: *Checking the constant coefficient of variation assumption for treatment (IP-enriched) observations.* Plotted is the coefficient of variation versus sample mean across 6 treatment replicates for each probe in the sample. Horizontal red line is the lowess fit to this scatter plot. This flat line indicates that the coefficient of variation assumption is overall satisfied.

The second class of diagnostic plots concerns the gamma observation component assumption. For various mean levels of the probes, the empirical quantiles are plotted against the quantiles of the fitted gamma distribution . Figure 3, generated by the following command, displays such a quantile plot for a total of 9 mean levels.

5 Fitting the Hierarchical Gamma Mixture Model

5.1 Partitioning of the genomic region for which ChIP-chip data are available into smaller genomic fragments

HGMM is fit by assuming that each genomic region has at most one peak and this requires the partitioning of the genomic region for which ChIP-chip data are available into shorter fragments. This can be carried out by the **partition** function. There are two parameters that control how fine

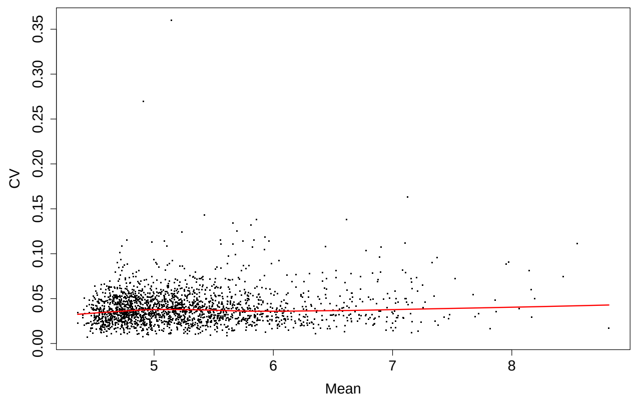

Figure 2: *Checking the constant coefficient of variation assumption for control observations.* Plotted is the coefficient of variation versus sample mean across 6 treatment replicates for each probe in the sample. Horizontal red line is the lowess fit to this scatter plot. This flat line indicates that the coefficient of variation assumption is overall satisfied.

the partitions are going to be. These are `gapmax` and `regionmax`. If the genomic gap between two adjacent probes is of at least `gapmax` base pairs (this is likely to occur due to repeat masking in the process of array design), then we have a natural point for partitioning. If the resulting genomic fragments are still long (longer than the user specified threshold `regionmax`), further partitioning is performed. For genomic fragments longer than the user specified maximum threshold `regionmax`, if the additional length is shorter than `gapmax`, this genomic fragment is left as it is. For example, assume that we set `gapmax = 500` and `regionmax = 2000`. Then, a fragment of 2100 base pairs will not be partitioned further since $2100 - 2000 < 500$.

```
> y = partition(xt, xc, locs, gapmax = 1000, regionmax = 2000)
> length(y)
```

```
[1] 47
```

The function **partition** returns a list of partitioned data. The list has as many elements as the number of generated genomic regions/fragments. Each element of the list is also a list of xt = "treatment (IP-enriched) data matrix", xc = "control data matrix", and locs = "genomic location" vector. Data matrices have as many rows as the number of probes in the genomic regions and a column for each replicate data. Genomic location vector has the actual genomic locations of the probes for each genomic region. Here, 2117 probes are partitioned into 47 regions. Each partition can be accessed by `y[[i]]` where i represents the partition number. Below is the summary of these newly generated genomic regions. We have genomic regions with as small as 2 probes and as many as 75 probes.

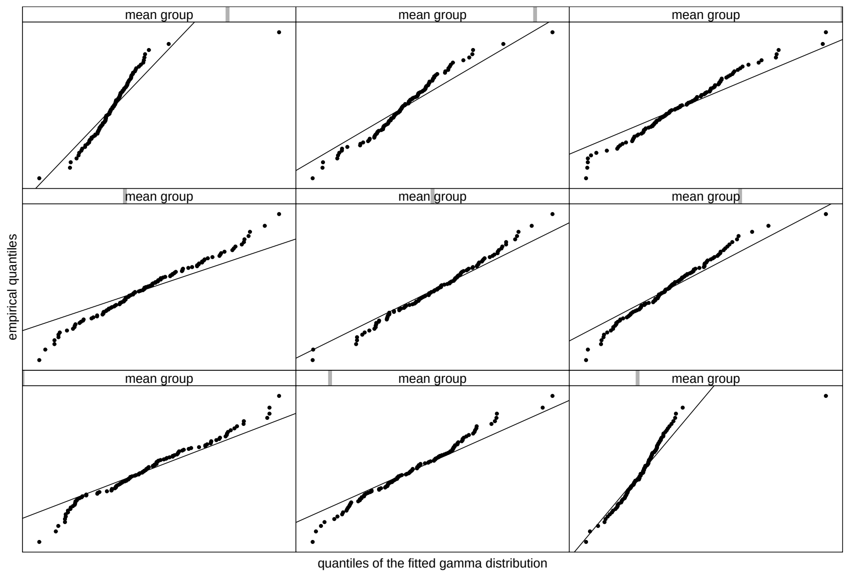

Figure 3: *Checking the gamma observation component assumption for the treatment (IP-enriched) sample.* For various mean levels, empirical quantiles of the IP-enriched hybridization data across probes with that mean level versus quantiles of the fitted gamma distribution are plotted.

> *unlist(lapply(y, a < − function(x)length(x$locs)))*

```
 [1] 43 55  2 30 44 37 62 58 50 37 49 52 52 56  4 33 48 31 44 63 49 48  4 40 33
 [26] 66 45 51 69 61 45 51 66  7 46 37 55 60 71 37 75 63 45  4 39 58 42
```

> *summary(unlist(lapply(y, a < − function(x)length(x$locs))))*

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   2.00   37.00   48.00   45.04   57.00   75.00 
```

Furthermore, below is a summary of the genomic length of the partitions. We regions that are as small as 84 base pairs and as large as 2800 base pairs in length.

> *summary(unlist(lapply(y, a < − function(x)x$locs[length(x$locs)]− x$locs[1])))*

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
     84    1822    1965    1778    1996    2800 
```

5.2 The EM algorithm for fitting the Hierarchical Gamma Mixture Model

The function **HGMM** implements the maximum likelihood estimation for the Hierarchical Gamma Mixture Model. Arguments of this function include: initial values for the model parameters a_0, x_0, a_1, a_2, and π_0 (`a0, x0, a1, a2, and mixprop`); a vector of allowed peak sizes (peak size is represented in terms of the number of probes allowed to form a peak) with their corresponding probabilities, i.e., weights that sum to 1, (`peaksize` and `rho`); maximum number of EM iterations allowed (`NITER`); minimum increase allowed on the observed data log likelihood to continue EM iterations (`eps`). The output from **HGMM** contains estimates of the model parameters including the mixing proportion (proportion of unbound regions) and various posterior probabilities in `eta` and `zeta` slots and further information in `data`, `L`, and `ollik` slots.

```
> mixprop = 0.5
> peaksize = 15
> rho = c(1)
> res = HGMM(y, peaksize, rho, mixprop, NITER = 100, a0 = 60, a1 = 400, a2 = 800, x0 = 5,
eps = 1e-4)
> res@mixprop
```

```
[1] 0.6157925
```

```
> res
```

```
 HGMM Fit


 Estimates of the model parameters:

  a0 : 59.36836
  a1 : 447.5725
  a2 : 805.1743
  x0 : 5.302375


 Estimate of the mixing proportion p0 (proportion of unbound regions):

  p0 : 0.6157925


 Additional slots: @eta, @zeta, @data, @L, @ollik
```

In particular, `eta` slot contains one minus the posterior probability of binding (interaction with the transcription factor) for each genomic region. One can either use all the genomic regions with eta values smaller than 0.5 or use the **fdrHGMM** function explained in the next section for deciding on a posterior probability threshold by controlling the False Discovery Rate (fdr) by the direct posterior probability approach of [4].

6 Post-processing

The function **fdrHGMM** implements the direct posterior probability approach of [4] for controlling the fdr at a user specified threshold. Using an fdr threshold value of 0.01 and a peak size of 15 (same as the peak size value used in fitting the model) probes, we identified 18 regions as bound. The output from **fdrHGMM** include the following for each identified peak: column 1:

start site; column 2: end site; column 3: log posterior odds; column 4: posterior probability of binding; column 5: genomic partition number. The last column is reported for easing the access to the data from specific peaks.

> peaks = fdrHGMM(res, 15, 0.01)
> peaks

```
          [,1]     [,2]       [,3]      [,4] [,5]
 [1,]  7777577  7778100 1383.25542 1.0000000    1
 [2,] 15644098 15644925  849.55104 1.0000000   17
 [3,] 24341295 24341793  580.56540 1.0000000   30
 [4,] 15703520 15704430  566.86126 1.0000000   18
 [5,] 24057822 24058318  529.37638 1.0000000   28
 [6,] 14634242 14634706  450.08264 1.0000000   12
 [7,] 11700158 11700992  403.05338 1.0000000    4
 [8,] 13201179 13201630  232.60714 1.0000000    7
 [9,] 30964550 30965031  191.75870 1.0000000   45
[10,] 29175284 29175733  176.16500 1.0000000   39
[11,] 17854668 17855150  143.82523 1.0000000   20
[12,] 31325905 31326936  121.79530 1.0000000   47
[13,] 29266195 29266600  113.13374 1.0000000   40
[14,] 14687687 14688330  103.54521 1.0000000   13
[15,] 11728556 11729076   99.45523 1.0000000    5
[16,] 29945156 29945641   90.66202 1.0000000   43
[17,] 14822815 14823339   74.53346 1.0000000   14
[18,] 29316036 29316817   16.50886 0.9999982   41
```

It is often desirable to combine peaks that are within very close proximity of each other, especially for the downstream sequence analysis. This package provides the **combinePK** function for combining adjacent peaks that are within `combineQ` base pairs of each other. Typically, `combineQ` is set to 500-1000 base pairs. Since we are using a subset of chr 21, we set this to 50000 for illustration purposes:

> combinePK(peaks,50000)

```
          [,1]     [,2]       [,3] [,4] [,5] [,6]
 [1,]  7777577  7778100 1383.25542    1    1    0
 [2,] 15644098 15644925  849.55104    1   17    0
 [3,] 24341295 24341793  580.56540    1   30    0
 [4,] 15703520 15704430  566.86126    1   18    0
 [5,] 24057822 24058318  529.37638    1   28    0
 [6,] 14634242 14634706  450.08264    1   12    0
 [7,] 11700158 11729076  403.05338    1    4    1
 [8,] 13201179 13201630  232.60714    1    7    0
 [9,] 30964550 30965031  191.75870    1   45    0
[10,] 29175284 29175733  176.16500    1   39    0
[11,] 17854668 17855150  143.82523    1   20    0
[12,] 31325905 31326936  121.79530    1   47    0
[13,] 29266195 29316817  113.13374    1   40    1
```

```
[14,] 14687687 14688330  103.54521    1   13    0
[15,] 29945156 29945641   90.66202    1   43    0
[16,] 14822815 14823339   74.53346    1   14    0
```

'1's in the 6th column of the output by **combinePK** indicate that two peaks in 4th and 5th region are coalesced into one peak and two peaks in 40th and 41th into another single peak. We further provide an example of a peak plot in Figure 4. This figure was generated by the following script:

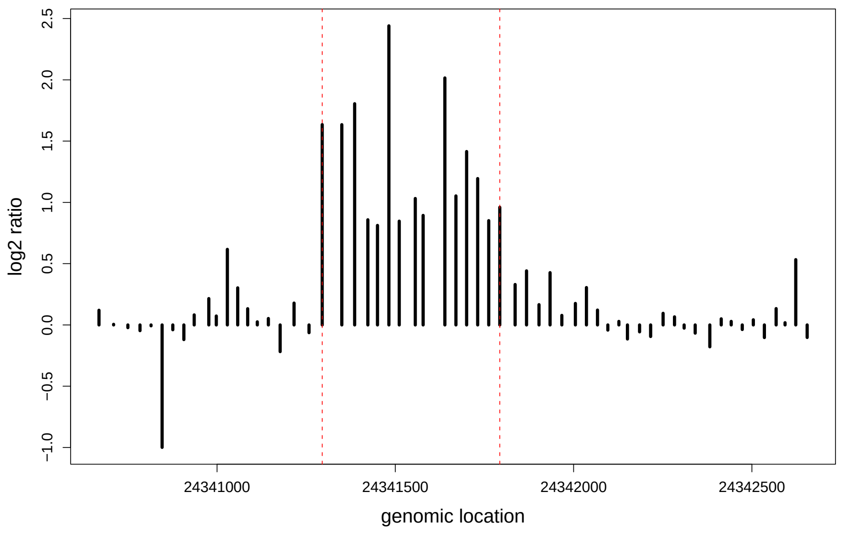

Figure 4: *A genomic region from chromosome 21 with a peak.* Data shown are from [2] and the genomic region is one of the 18 regions identified as bound by the HGMM. Dashed red vertical lines mark the estimated start and end positions of the peak.

> i = 3
> ii = peaks[i, 5]
#compute the log2 ratio of treatment to control observations for each probe.
> ratio = log2(apply(2ˆres@data[[ii]]$xt, 1, mean))
- log2(apply(2ˆres@data[[ii]]$xc, 1, mean))
> plot(res@data[[ii]]$locs, ratio, type = "h", lwd = 4, xlab = "genomic location", ylab = "log2 ratio", cex.lab = 1.5, cex.axis = 1.2)
#mark the peak start and end sites.
> abline(v = c(peaks[i, 1], peaks[i, 2]), col = "red", lty = 2)

References

[1] S. Keleş. Mixture modeling of genome-wide localization of transcription factors. Technical report, University of Wisconsin, Madison, 2005. Department of Biostatistics and Medical Informatics, #189. `http://www.cs.wisc.edu/~keles/ggcc.v1.pdf`.

[2] S. Cawley, S. Bekiranov, H.H. Ng, P. Kapranov, E.A. Sekinger, D. Kampa, A. Piccolboni, V.I. Sementchenko, J. Cheng, A.J. Williams, R. Wheeler, B . Wong, J. Drenkow, M. Yamanaka, S. Patel, S. Brubaker, H. Tammana, G. Helt, K. Struhl, and T.R. Gingeras. Unbiased mapping of transcription factor binding sites along h uman chromosomes 21 and 22 points to widespread regulation of non-coding RNAs. *Cell*, 116(4):499–511, 2004.

[3] B. M. Bolstad, Irizarry R. A., M. Astrand, and T. P. Speed. A comparison of normalization methods for high density oligonucleotide array data based on bias and variance. *Bioinformatics*, 19(2):185–193, 2003.

[4] M.A. Newton, A. Noueiry, D. Sarkar, and P. Ahlquist. Detecting differential gene expression with a semiparametric hierarchical mixture method. *Biostatistics*, 5:155–176, 2004.

Apprendix: A note on the low level data files from Affymetrix tiling arrays

For the p53 ChIP-chip data, [2] provided the raw CEL data (including PM measurements, probe sequences etc...) joint with the genomic positions. These data are available at `http://transcriptome.affymetrix.com/publication/tfbs/`.

In general, to prepare data for the statistical analysis one needs to combine CEL files that contain experimental array data with the BPMAP files that contain the information regarding where on the genome each probe maps to. Currently, we are using the Java-based `TiMAT` (`http://bdtnp.lbl.gov/TiMAT/`) software of David Nix (from LBL) to accomplish this task and we are interested in incorporating this process into `R`. As a result of this process, we obtain files of the following format for each replicate treatment and control experiment:

```
TGCGAGAGTAGTGCCAACATATTGT       f       chr2L   88      323     89
TGCGAGAGTAGTGCCAACATATTGT       f       chr2L   88      410     122
TGCGAGAGTAGTGCCAACATATTGT       f       chr2L   88      482     97
TGCGAGAGTAGTGCCAACATATTGT       f       chr2L   88      399     120
TGCGAGAGTAGTGCCAACATATTGT       f       chr2L   88      358     105
TGCGAGAGTAGTGCCAACATATTGT       f       chr2L   88      395     86
TGCGAGAGTAGTGCCAACATATTGT       f       chr2L   88      391     102
TGCGAGAGTAGTGCCAACATATTGT       f       chr2L   88      352     104
TGCGAGAGTAGTGCCAACATATTGT       f       chr2L   88      395     138
TGCGAGAGTAGTGCCAACATATTGT       f       chr2L   88      390     128
GATGATAATATATTCAAGTTGCCGC       f       chr2L   157     146     37
ATAATATATTCAAGTTGCCGCTAAT       f       chr2L   161     177     63
ATAATATATTCAAGTTGCCGCTAAT       f       chr2L   161     229     67
ATAATATATTCAAGTTGCCGCTAAT       f       chr2L   161     197     54
ATAATATATTCAAGTTGCCGCTAAT       f       chr2L   161     171     65
ATAATATATTCAAGTTGCCGCTAAT       f       chr2L   161     165     65
ATAATATATTCAAGTTGCCGCTAAT       f       chr2L   161     186     55
ATAATATATTCAAGTTGCCGCTAAT       f       chr2L   161     191     59
ATAATATATTCAAGTTGCCGCTAAT       f       chr2L   161     235     64
```

Here, the columns represent (1) probe sequence; (2) strand information ("f" for "-" strand); (3) chromosome number; (4) genomic location on the chromosome; (4) PM value; (5) MM value. From these files, one can extract `xt`, `xc` (which have PM values from the treatment and control experiments) and `locs` (which has the genomic locations of the probes) to feed into the **HGMM** package.