

# Analysis of ChIP-seq Data with ‘mosaics’ Package

Dongjun Chung<sup>1</sup>, Pei Fen Kuan<sup>2</sup> and Sündüz Keleş<sup>1,3</sup>

<sup>1</sup>Department of Statistics, University of Wisconsin  
Madison, WI 53706.

<sup>2</sup>Department of Biostatistics, University of North Carolina at Chapel Hill  
Chapel Hill, NC 27599.

<sup>3</sup>Department of Biostatistics and Medical Informatics, University of Wisconsin  
Madison, WI 53706.

September 24, 2010

## 1 Overview

This vignette provides an introduction to the analysis of ChIP-seq data with ‘mosaics’ package. R package `mosaics` implements MOSAiCS, a statistical framework for the analysis of ChIP-seq data, proposed in [1]. MOSAiCS stands for “**MO**del-based one and two **S**ample **A**nalysis and **I**nference for **ChIP-Seq** Data”. Based on careful investigation of biases in ChIP-seq data such as mappability and GC content, MOSAiCS has been developed as a flexible mixture modeling approach for detecting peaks of one-sample (ChIP sample) or two-sample (ChIP sample and matched control sample) ChIP-seq data.

‘mosaics’ package assumes chromosome-wise analysis of ChIP-seq data. In this vignette, we use chromosome 22 of the data from a ChIP-seq experiment of STAT1 binding in interferon- $\gamma$ -stimulated HeLa S3 cells from [2]. The package can be loaded with the command:

```
R> library("mosaics")
```

## 2 Workflow: One-Sample Analysis

### 2.1 Read Bin-Level Data into the R Environment

For the one-sample analysis, MOSAiCS requires four information: bin-level ChIP data, mappability score, GC content score, and sequence ambiguity score. For human genome (HG18) and mouse genome (MM9), we provide mappability score, GC content score, and sequence ambiguity score for each chromosome. Please check [http://groups.google.com/group/mosaics\\_user\\_group](http://groups.google.com/group/mosaics_user_group) for further information about these files and how to download them. Bin-level ChIP data can be obtained from the aligned data (e.g., a file obtained from ELAND) using the *perl* codes we provide. Users can download these *perl* codes from [http://groups.google.com/group/mosaics\\_user\\_group](http://groups.google.com/group/mosaics_user_group).

Bin-level data obtained from our *perl* codes can be imported to the R environment with the command:

```
R> bin1 <- readBins(type = c("chip", "M", "GC", "N"), fileName = c("./chip_chr22.txt",  
+      "./M_chr22.txt", "./GC_chr22.txt", "./N_chr22.txt"))
```

In ‘type’, “chip”, “M”, “GC”, “N” indicate bin-level ChIP data, mappability score, GC content score, and sequence ambiguity score, respectively. User needs to provide the corresponding file names in ‘fileName’. ‘mosaics’ package assumes that each file name in ‘fileName’ is provided in the same order as in ‘type’. This method also provides information about percentage of bins with ambiguous sequences (i.e., sequence ambiguity score = 1 ) and first/last coordinates of bins before and after preprocessing.

R package *mosaics* provides simple investigation tools for bin-level data. The following command prints out basic information about the bin-level data, such as number of coordinates and total “effective” tag counts. Total effective tag counts are defined as the sum of tag counts of all bins. This value is usually larger than the number of all the tags used in the analysis. Total effective tag counts provide more useful information about the size of data used in the analysis.

```
R> bin1
```

```
Summary: bin-level data (class: BinData)
```

```
-----
- # of coordinates = 697101
- total effective tag counts = 1634452
  (sum of tag counts of all bins)
- input sample is incorporated
- mappability is incorporated
- GC content is incorporated
- uniquely matched tags are assumed
-----
```

‘print’ method returns the bin-level data in data frame format.

```
R> print(bin1)[1:10, ]
```

	coord	tagCount	mappability	gcContent	input
1	14429800	0	0.06	0.48	0
2	14429850	0	0.19	0.51	0
3	14429900	0	0.31	0.53	0
4	14429950	0	0.44	0.49	0
5	14430000	0	0.57	0.49	0
6	14430050	0	0.69	0.46	0
7	14430100	0	0.82	0.44	0
8	14430150	0	0.94	0.43	0
9	14430200	0	1.00	0.41	0
10	14430250	0	1.00	0.39	0

‘plot’ method provides exploratory plots of mean ChIP tag counts. ‘plotType’ controls which plot to be drawn. ‘plotType=“M”’ and ‘plotType=“GC”’ generate a plot of mean tag counts versus mappability score and GC content score, respectively. If ‘plotType’ is not specified, this method plots histogram of mean ChIP tag count. Figure 1, 2, and 3 show examples. One can see that mean tag count increases as mappability score increases. As GC content score increases, mean tag count increases to certain value of GC content score (about 0.6) and then it decreases.

```
R> plot(bin1, plotType = "M")
R> plot(bin1, plotType = "GC")
R> plot(bin1)
```

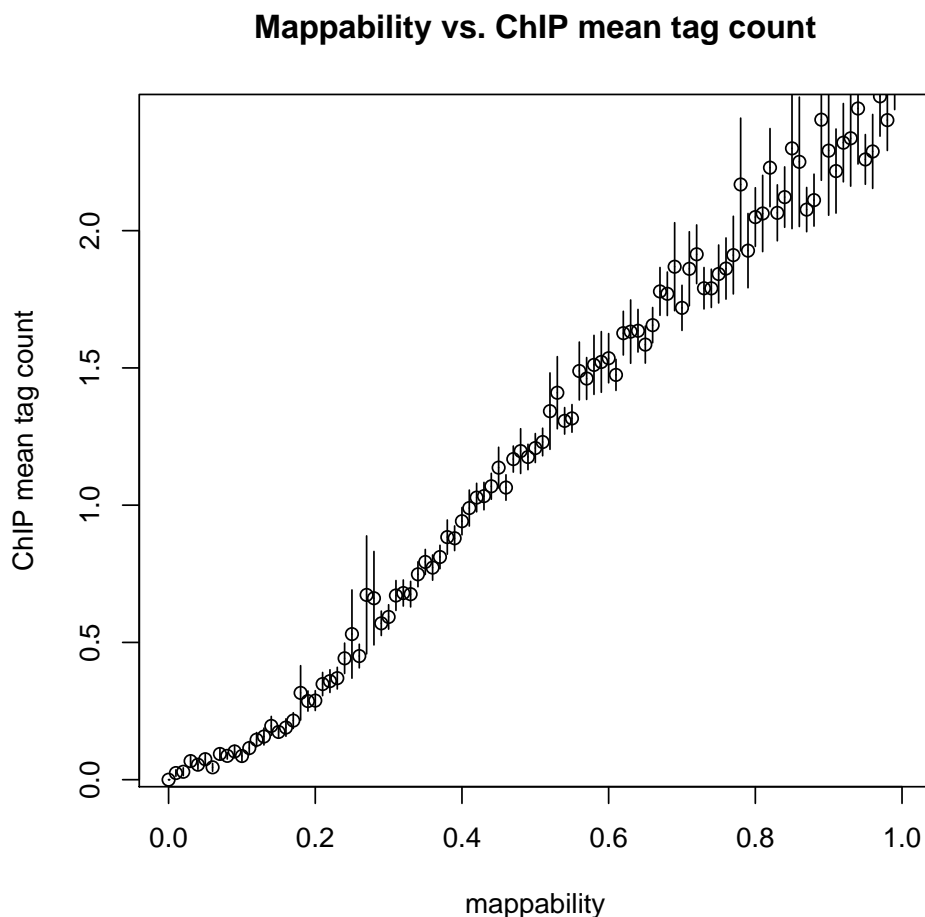


Figure 1: Plot of mean ChIP tag counts versus mappability score.

## 2.2 Fit MOSAiCS

We are now ready to fit a MOSAiCS model. The bin-level data above (say, `bin1`) is used as an input. A MOSAiCS model is fitted with the command:

```
R> fit1 <- mosaicsFit(bin1, analysisType = "OS")
```

‘`analysisType="OS"`’ indicates implementation of the one-sample analysis. Without input sample, only the one-sample analysis is permitted. ‘`mosaicsFit`’ fits both one-signal-component and two-signal-component models. When calling peaks, user can choose the number of signal components used for the final model. The optimal choice of the number of signal components depends on the characteristics of data. In order to support user in choice of optimal signal model, `mosaics` package provides Bayesian Information Criterion (BIC) values and Goodness of Fit (GOF) plot of these signal models.

The following command shows BIC of signal models, with information about the parameters used in fitting the null (non-enriched) distribution. Lower BIC indicates better model fit and we can see that two-signal-component model has lower BIC in our case.

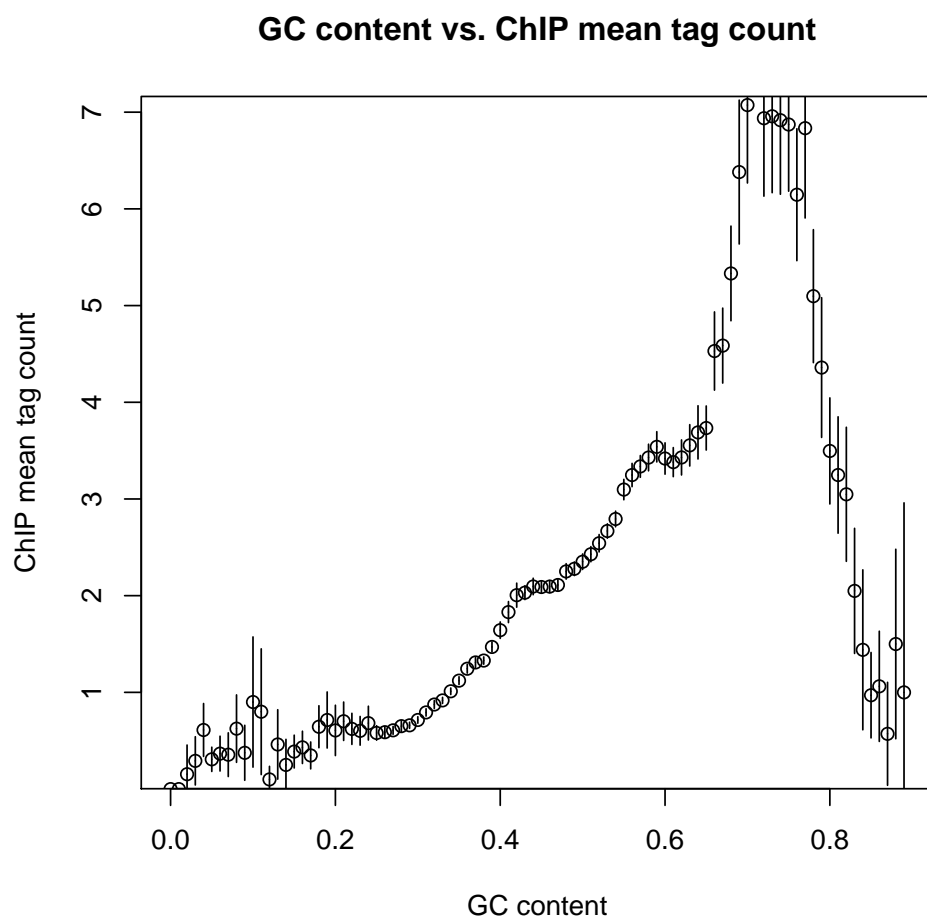


Figure 2: Plot of mean ChIP tag counts versus GC content score.

```
R> fit1
```

```
Summary: MOSAiCS model fitting (class: MosaicsFit)
```

```
-----
analysis type: one-sample analysis
```

```
parameters used: k = 3, meanThres = 0
```

```
BIC of one-signal-component model = 1257229
```

```
BIC of two-signal-component model = 1243432
-----
```

‘plot’ method provides GOF plot. Using the GOF plot, user can compare null model, one-signal-component model, and two-signal-component model, with the actual data. Specifically, user can check which signal model fits the actual data better. Figure 4 shows an example GOF plot. One can see that two-signal-component model provides better fit for our data.

```
R> plot(fit1)
```

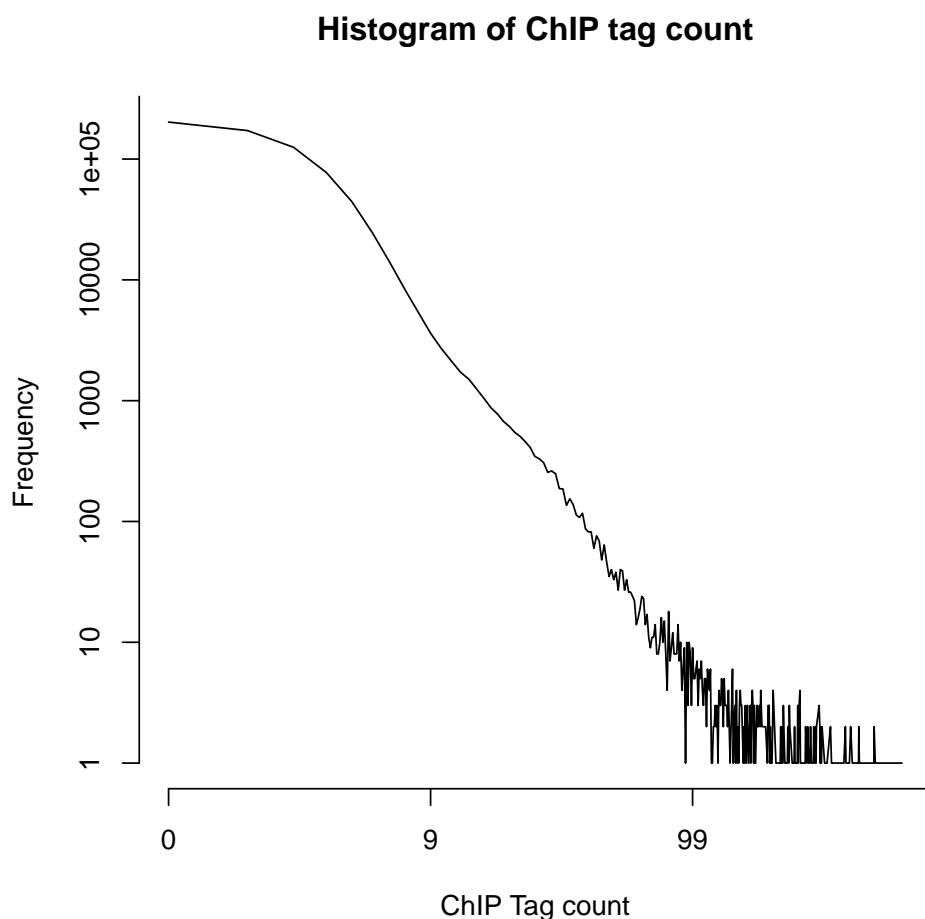


Figure 3: Histogram of mean ChIP tag counts.

## 2.3 Call Peaks

[Note: We may be going to include explanation of peak refinement and the procedure to implement it in this section.] We can call peaks with the command:

```
R> peak1 <- mosaicsPeak(fit1, signalModel = "2S", FDR = 0.05, maxgap = 200,
+   minsize = 50, thres = 10)
```

```
Info: use two-signal-component model.
Info: calculating posterior probabilities...
Info: calling peaks...
Info: empirical FDR (before thresholding) = 0.05
Info: empirical FDR (after thresholding) = 0.032
Info: done!
```

Using BIC and GOF plot in the previous section, we can conclude that two-signal-component model fits our data better. ‘signalModel="2S"’ indicates two-signal-component model. Similarly,

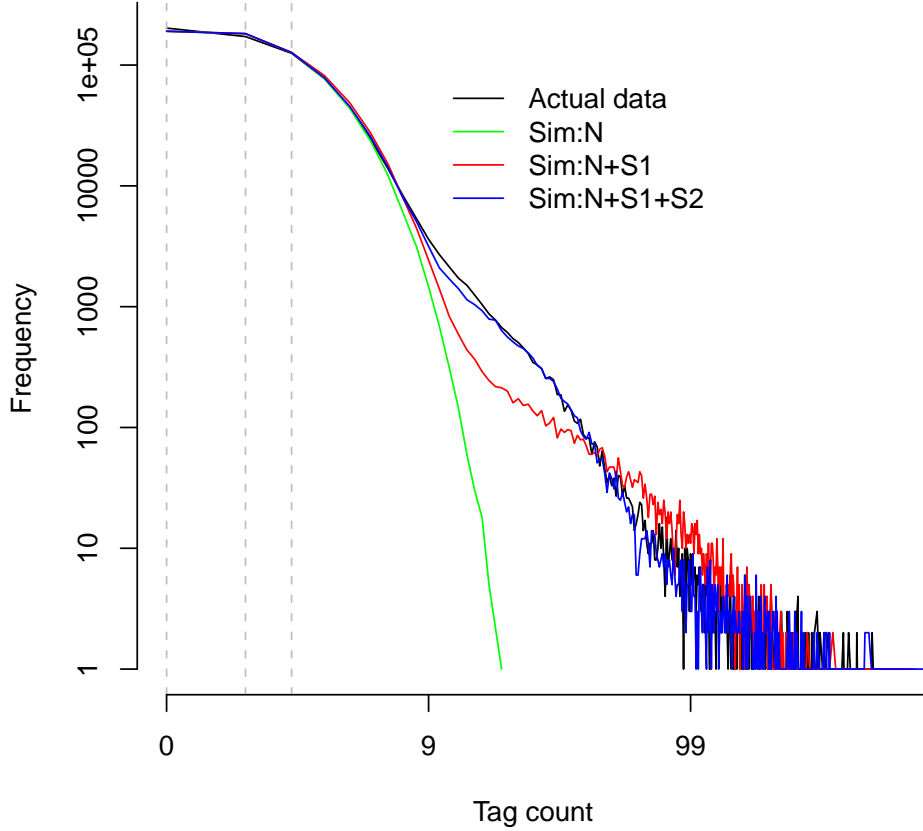


Figure 4: Goodness of Fit (GOF) plot. The plot shows null model (Sim:N), one-signal-component model (Sim:N+S1), and two-signal-component model (Sim:N+S1+S2), with the actual data.

one-signal-component model can be specified by `'signalModel="1S"`. Additionally, user can control false discovery rate (`'FDR'`), distance (in bp) to combine initial peaks (`'maxgap'`), minimum size (in bp) of the initial peak so that it is claimed as a peak (`'minsize'`), and minimum ChIP tag count in each bin so that the bin is claimed as a peak (`'thres'`). User needs to choose these parameters in order to call peaks accurately. We recommend to set `'maxgap'` and `'minsize'` as the fragment length and bin size, respectively. Initial nearby peaks are merged if the distance between them is less than `'maxgap'`. This is because a fragment covers several bins when bin size is chosen to be shorter than the fragment length. The initial peaks of which length are shorter than `'minsize'` are removed. This is because very small initial peak (e.g., singleton) is less likely to be a real peak when bin size is chosen to be shorter than the fragment length. When bin size is same as the fragment length, `'minsize'` should be smaller than the fragment length. This is because there are many (potentially real) peaks of which size is the same as bin size. `'thres'` is employed to filter out initial peaks with small tag counts because these initial peaks might correspond to false discoveries. Optimal choice depends on the sequencing depth of the ChIP-Seq data. If negative value is provided for `'thres'`, thresholding is not applied.

The following command prints out basic information of called peaks such as number of called peaks, median peak length, and empirical false discovery rate.

```
R> peak1
```

```
Summary: MOSAiCS peak calling (class: MosaicsPeak)
```

```
-----
final model: one-sample analysis with two signal components
setting: FDR = 0.05, maxgap = 200, minsize = 50, thres = 10
# of peaks = 1690
median peak length = 349
empirical FDR = 0.05
-----
```

‘print’ method returns the peak calling results in data frame format. This data frame can contain different number of columns, depending on ‘analysisType’ of ‘mosaicsFit’. For example, if ‘analysisType=“OS”’, columns are peak start position, peak stop position, peak size, averaged posterior probabilities, minimum posterior probability, averaged ChIP tag counts, and maximum of ChIP tag counts in each peak. Here, posterior probability of a bin means the probability that the bin is not a peak, conditional on data. Hence, small posterior probability indicates more evidence that the bin is actually a peak. This output is intended as an input for downstream analysis such as motif analysis.

```
R> print(peak1)[1:10, ]
```

	peakStart	peakStop	peakSize	aveP	minP	aveChipCount
1	15228250	15228599	349	1.246238e-02	1.901677e-05	11.28571
2	15228850	15229249	399	1.815465e-05	4.122910e-10	12.00000
3	15229750	15230399	649	5.103986e-05	5.809272e-15	16.46154
4	15231200	15231449	249	2.798188e-05	2.409283e-07	12.80000
5	15233850	15234099	249	2.351081e-05	1.247978e-07	12.80000
6	15234400	15234499	99	2.400522e-04	1.164164e-05	12.50000
7	15236050	15236999	949	1.297906e-03	3.619806e-10	11.84211
8	15238400	15239699	1299	7.363318e-03	3.473284e-10	11.84615
9	15241600	15242099	499	1.484500e-04	1.045810e-14	16.90000
10	15712300	15712549	249	4.371991e-08	4.042070e-11	18.40000

	maxChipCount	map	GC
1	13	0.8242857	0.3657143
2	15	0.5225000	0.3600000
3	23	0.7369231	0.3500000
4	14	0.6240000	0.3780000
5	15	0.7860000	0.3360000
6	14	0.8800000	0.3450000
7	19	0.7315789	0.3484211
8	20	0.7319231	0.3519231
9	25	0.6750000	0.3720000
10	21	0.6380000	0.4100000

User can also export peak calling results to text files in diverse file formats. Currently, ‘mosaics’ package support TXT, BED, and GFF file formats. In the exported file, TXT file format (‘type=“txt”’)

includes all the columns that ‘print’ method returns. ‘type="bed"’ and ‘type="gff"’ export peak calling results in standard BED and GFF file format, respectively, where score is summed tag counts in each peak. Peak calling results can be exported in TXT, BED, and GFF file formats, respectively, with the commands:

```
R> export(peak1, type = "txt", fileLoc = "./", fileName = "OSpeakList.txt",
+       chrID = "chr22")
R> export(peak1, type = "bed", fileLoc = "./", fileName = "OSpeakList.bed",
+       chrID = "chr22")
R> export(peak1, type = "gff", fileLoc = "./", fileName = "OSpeakList.gff",
+       chrID = "chr22")
```

‘fileLoc’ and ‘fileName’ indicate the directory and the name of the exported file. ‘chrID’ means chromosome ID. ‘mosaics’ package assumes chromosome-wise analysis and does not require user to supply chromosome ID during the analysis. In the exported file, the character specified in ‘chrID’ is used as chromosome ID, which will be the first column in standard BED and GFF file formats.

## 3 Workflow: Two-Sample Analysis

### 3.1 Analysis using Mappability and GC Content

When input data is additionally provided, the two-sample analysis can be implemented. The procedure to implement the two-sample analysis is analogous to the procedure to implement the one-sample analysis. Bin-level data for the two-sample analysis can be imported to the R environment with the command:

```
R> bin2 <- readBins(type = c("chip", "M", "GC", "N", "input"), fileName = c("./chip_chr22.txt",
+       "./M_chr22.txt", "./GC_chr22.txt", "./N_chr22.txt", "./Input_chr22.txt"))
```

In other words, we need one more element, “input”, in ‘type’. The file name of input data also needs to be specified in ‘fileName’. When input data is provided, ‘plotType="input"’ generates a plot of mean ChIP tag counts versus input tag count. Moreover, ‘plotType="M|input"’ and ‘plotType="GC|input"’ generate plots of mean ChIP tag counts versus mappability score and GC content score, respectively, conditional on input tag count.

```
R> plot(bin1, plotType = "input")
R> plot(bin1, plotType = "M|input")
R> plot(bin1, plotType = "GC|input")
```

In order to fit MOSAiCS model for the two-sample analysis, when calling ‘mosaicsFit’ method, user should specify ‘analysisType="TS"’ instead of ‘analysisType="OS"’.

```
R> fit2 <- mosaicsFit(bin2, analysisType = "TS")
```

Peak calling can be done exactly in the same way as before. Also all the other methods can still be used as in the one-sample analysis case.

```
R> peak2 <- mosaicsPeak(fit2, signalModel = "2S", FDR = 0.05, maxgap = 200,
+       minsize = 50, thres = 10)
```



### 3.2 Analysis without using Mappability and GC Content

Application of MOSAiCS methods to the real data showed that consideration of mappability and GC content in the model improves sensitivity and specificity of peak calling [1]. However, `mosaics` package still permits the two-sample analysis without considering mappability and GC content. In order to fit MOSAiCS model for the two-sample analysis without considering mappability and GC content, `'analysisType="IO"'` needs to be specified when calling `'mosaicsFit'` method.

```
R> fit3 <- mosaicsFit(bin2, analysisType = "IO")
```

Furthermore, in this case, mappability score, GC content score, and sequence ambiguity score might not be incorporated when importing bin-level data. User can import bin-level data and fit MOSAiCS model for the two-sample analysis without mappability and GC content, with the commands:

```
R> bin4 <- readBins(type = c("chip", "input"), fileName = c("./chip_chr22.txt",  
+      "./Input_chr22.txt"))  
R> fit4 <- mosaicsFit(bin4, analysisType = "IO")
```

Note that with these commands `'mosaics'` package skips initial processing of bin-level data using mappability score, GC content score, and sequence ambiguity score. We recommend to incorporate mappability score, GC content score, and sequence ambiguity score as long as they are available to users.

## 4 Tuning of MOSAiCS Parameters

For the two-sample analysis, `'mosaics'` package permits user to tune the parameters used in fitting the null distribution of MOSAiCS model. When both mappability score and GC content score are available (i.e., the two-sample analysis using mappability and GC content), user can control two tuning parameters, `'s'` and `'d'`. `'s'` means the cut-off to determine lower and higher values of input data. `'d'` means the order used in the transformation of input data. Please see [1] for further details about model parameters. For the two-sample analysis using mappability and GC content, `'mosaics'` package uses the following parameter setting as default:

```
R> fit3_ts <- mosaicsFit(bin2, analysisType = "TS", s = 2, d = 0.25)
```

When mappability score and GC content score are not available (i.e., the two-sample analysis without using mappability and GC content), user can tune only one parameter, `'d'`. For the two-sample analysis without using mappability and GC content, `'mosaics'` package uses the following parameter setting as default:

```
R> fit3_io <- mosaicsFit(bin2, analysisType = "IO", d = 0.25)
```

## 5 Conclusion and Ongoing Development

R package `mosaics` provides effective tools to read and investigate ChIP-seq data, fit MOSAiCS, and claim peaks. We are continuously working on improving `mosaics` package further in more friendly user interface and more effective and diverse investigation tools.

## References

- [1] Kuan, PF, D Chung, JA Thomson, R Stewart, and S Keleş (2010), “A Statistical Framework for the Analysis of ChIP-Seq Data”, submitted ([http://works.bepress.com/sunduz\\_keles/19/](http://works.bepress.com/sunduz_keles/19/)).
- [2] Rozowsky, J, G Euskirchen, R Auerbach, D Zhang, T Gibson, R Bjornson, N Carriero, M Snyder, and M Gerstein (2009), “PeakSeq enables systematic scoring of ChIP-Seq experiments relative to controls”, *Nature Biotechnology*, 27, 66-75.