

Analysis of ChIP-seq Data with ‘mosaics’ Package

Dongjun Chung¹, Pei Fen Kuan² and Sündüz Keleş^{1,3}

¹Department of Statistics, University of Wisconsin
Madison, WI 53706.

²Department of Biostatistics, University of North Carolina at Chapel Hill
Chapel Hill, NC 27599.

³Department of Biostatistics and Medical Informatics, University of Wisconsin
Madison, WI 53706.

January 16, 2012

1 Overview

This vignette provides an introduction to the analysis of ChIP-seq data with the ‘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”. It implements a flexible parametric mixture modeling approach for detecting peaks, i.e., enriched regions, in one-sample (ChIP sample) or two-sample (ChIP and control samples) ChIP-seq data. It accounts for mappability and GC content biases that arise in ChIP-seq data.

The package can be loaded with the command:

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

2 Getting started

We assume that you already have the aligned read files for your samples (ChIP and/or control) such as aligned read files obtained from the ELAND or bowtie aligner. R package ‘mosaics’ analyzes the data after converting aligned read files into bin-level files for modeling and visualization purposes. These bin-level data can easily be generated from the aligned read files with the command:

```
R> constructBins(infileLoc = "/scratch/eland/", infileName = "STAT1_eland_results.txt",  
+   fileFormat = "eland_result", outfileLoc = infileLoc, byChr = FALSE,  
+   fragLen = 200, binSize = fragLen, capping = 0)
```

You can specify the directory, name, and file format of the aligned read file in ‘infileLoc’, ‘infileName’, and ‘fileFormat’, respectively. ‘constructBins’ method currently allows the following aligned read file formats: Eland result (“eland_result”), Eland extended (“eland_extended”), Eland export (“eland_export”), default Bowtie (“bowtie”), SAM (“sam”), BED (“bed”), and CSEM BED (“csem”). This method assumes that these aligned read files are obtained from single-end tag (SET) experiments. If input file format is neither BED nor CSEM BED, it retains only reads mapping uniquely to the reference genome (uni-reads).

Even though ‘constructBins’ retains only uni-reads for most aligned read files, reads mapping to multiple locations on the reference genome (multi-reads) can still be easily incorporated

into bin-level data by utilizing our multi-read allocator, CSEM (ChIP-Seq multi-read allocator using Expectation-Maximization algorithm). Galaxy tool for CSEM is available in Galaxy Tool Shed (<http://toolshed.g2.bx.psu.edu/>). Please check “csem” under “Next Gen Mappers”. Stand-alone version of CSEM is also available at <http://www.stat.wisc.edu/~keles/Software/multi-reads/>. CSEM exports uni-reads and allocated multi-reads into standard BED file and corresponding bin-level files can be constructed by applying ‘constructBins’ to this BED file with the argument ‘fileFormat="csem"’.

If ‘byChr=FALSE’, bin-level data for all chromosomes are exported to one file named as ‘[infileName]_fragL[fragLen]_bin[binSize].txt’. If ‘byChr=TRUE’, bin-level data for each chromosome is exported to a separate file named as ‘[chrID]_[infileName]_fragL[fragLen]_bin[binSize].txt’, where [chrID] is chromosome ID that reads align to. By default, constructed bin-level files are exported to the directory that the aligned read file is located at. If you prefer to export them to another directory, you can specify it in ‘outfileLoc’. Bin-level files are named as These chromosome IDs are extracted from the aligned read file.

In addition, you can set average fragment length and bin size in ‘fragLen’ and ‘binSize’, respectively, and these arguments control the resolution of bin-level ChIP-seq data. By default, average fragment length is set to 200 bp, which is the common fragment length for Illumina sequences, and bin size equals to average fragment length. ‘capping’ argument indicates maximum number of reads allowed to start at each nucleotide position and it is to exclude extremely large read counts that might correspond to PCR amplification artifacts. ‘capping’ is not used if it is set to some non-positive value. Small value (e.g., 3) are recommended only for the ChIP-seq data with low sequencing depth and ‘capping’ is not used by default.

You might also need to have bin-level mappability, GC content, and sequence ambiguity score files for the reference genome you are working with, depending on the arguments of the methods ‘readBins’ and ‘mosaicsFit’ (read the next section for more details). If you are working with organisms such as human (HG18 and HG19), mouse (MM9), rat (RN4), and Arabidopsis (TAIR9), you can download their corresponding preprocessed mappability, GC content, and sequence ambiguity score files at <http://www.stat.wisc.edu/~keles/Software/mosaics/>. If your reference genome of interest is not listed on our website, you can inquire about it at our Google group, http://groups.google.com/group/mosaics_user_group, and we would be happy to add your genome of interest to the list. The companion website also provides all the related scripts and easy-to-follow instructions to prepare these files. Please check <http://www.stat.wisc.edu/~keles/Software/mosaics/> for more details. We encourage questions or requests regarding ‘mosaics’ package to be posted on our Google group http://groups.google.com/group/mosaics_user_group.

3 Workflow: Two-Sample Analysis

3.1 Reading Bin-Level Data into the R Environment

‘mosaics’ package assumes chromosome-wise analysis of ChIP-seq data. For the two-sample analysis, you need preprocessed bin-level ChIP data, control sample data, mappability score, GC content score, and sequence ambiguity score. In this vignette, we use chromosome 21 data from a ChIP-seq experiment of STAT1 binding in interferon- γ -stimulated HeLa S3 cells [2]. ‘mosaicsExample’ package provides this example dataset.

```
R> library(mosaicsExample)
```

Bin-level data can be imported to the R environment with the command:

```
R> exampleBinData <- readBins(type = c("chip", "input", "M", "GC",
+   "N"), fileName = c(system.file(file.path("extdata", "chip_chr21.txt"),
+   package = "mosaicsExample"), system.file(file.path("extdata",
+   "input_chr21.txt"), package = "mosaicsExample"), system.file(file.path("extdata",
+   "M_chr21.txt"), package = "mosaicsExample"), system.file(file.path("extdata",
+   "GC_chr21.txt"), package = "mosaicsExample"), system.file(file.path("extdata",
+   "N_chr21.txt"), package = "mosaicsExample")))
```

```
-----
Info: preprocessing summary
-----
```

```
- percentage of bins with ambiguous sequences: 27%
  (these bins will be excluded from the analysis)
- before preprocessing:
    first coordinates = 0, last coordinates = 46944350
- after preprocessing:
    first coordinates = 9719550, last coordinates = 46944250
-----
```

For the ‘type’ argument, “chip”, “input”, “M”, “GC”, and “N” indicate bin-level ChIP data, control sample data, mappability score, GC content score, and sequence ambiguity score, respectively. You need to specify the corresponding file names in ‘fileName’. ‘mosaics’ package assumes that each file name in ‘fileName’ is provided in the same order as in ‘type’.

R package *mosaics* provides functions for generating simple summaries of the data. The following command prints out basic information about the bin-level data, such as number of bins and total “effective tag counts”. “Total effective tag counts” is defined as the sum of the tag counts of all bins. This value is usually larger than the sequencing depth since tags are counted after extension to average fragment length and an extended fragment can contribute to multiple bins.

```
R> exampleBinData
```

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

```
- # of chromosomes in the data: 1
- total effective tag counts: 1637819
  (sum of ChIP tag counts of all bins)
- control sample is incorporated
- mappability score is incorporated
- GC content score is incorporated
- uni-reads are assumed
-----
```

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

```
R> print(exampleBinData)[51680:51690, ]
```

	chrID	coord	tagCount	mappability	gcContent	input
51680	chr21	15353100	10	1.00	0.36	4
51681	chr21	15353150	25	1.00	0.38	3

51682	chr21	15353200	61	1.00	0.39	5
51683	chr21	15353250	105	1.00	0.39	5
51684	chr21	15353300	125	1.00	0.39	6
51685	chr21	15353350	124	1.00	0.38	6
51686	chr21	15353400	109	1.00	0.38	7
51687	chr21	15353450	72	1.00	0.36	4
51688	chr21	15353500	30	0.99	0.36	2
51689	chr21	15353550	10	0.99	0.36	1
51690	chr21	15353600	6	0.99	0.36	1

‘plot’ method provides exploratory plots for the ChIP data. Different type of plots can be obtained by varying the ‘plotType’ argument. ‘plotType="M"' and ‘plotType="GC"' generate plots of mean ChIP tag counts versus mappability and GC content scores, respectively. ‘plotType="input"' generates a plot of mean ChIP tag counts versus control tag counts. Moreover, ‘plotType="M|input"' and ‘plotType="GC|input"' generate plots of mean ChIP tag counts versus mappability and GC content scores, respectively, conditional on control tag counts. If ‘plotType’ is not specified, this method plots the histogram of ChIP tag counts.

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

Figures 1, 2, 3, 4, 5, and 6 display examples of different types of plots. As discussed in [1], we observe that mean ChIP tag count increases as mappability score increases (Figure 2). Mean ChIP tag count depends on GC score in a non-linear fashion (Figure 3). The relationship between mean ChIP tag counts and control tag counts seems to be linear, especially for small control tag counts (Figure 4). When we condition on control tag counts (Figures 5 and 6), mean ChIP tag count versus mappability and GC content relations exhibit similar patterns to that of marginal plots given in Figures 2 and 3. MOSAiCS incorporates this observation by modeling ChIP tag counts from non-peak regions with a small number of control tag counts as a function of mappability, GC content, and control tag counts.

3.2 Fitting MOSAiCS

We are now ready to fit a MOSAiCS model using the bin-level data above (`exampleBinData`) with the command:

```
R> exampleFit <- mosaicsFit(exampleBinData, analysisType = "TS")
```

‘analysisType="TS"' indicates implementation of the two-sample analysis. ‘mosaicsFit’ fits both one-signal-component and two-signal-component models. When identifying peaks, you can choose the number of signal components to be used for the final model. The optimal choice of the number of signal components depends on the characteristics of data. In order to support users in the choice of optimal signal model, `mosaics` package provides Bayesian Information Criterion (BIC) values and Goodness of Fit (GOF) plots of these signal models.

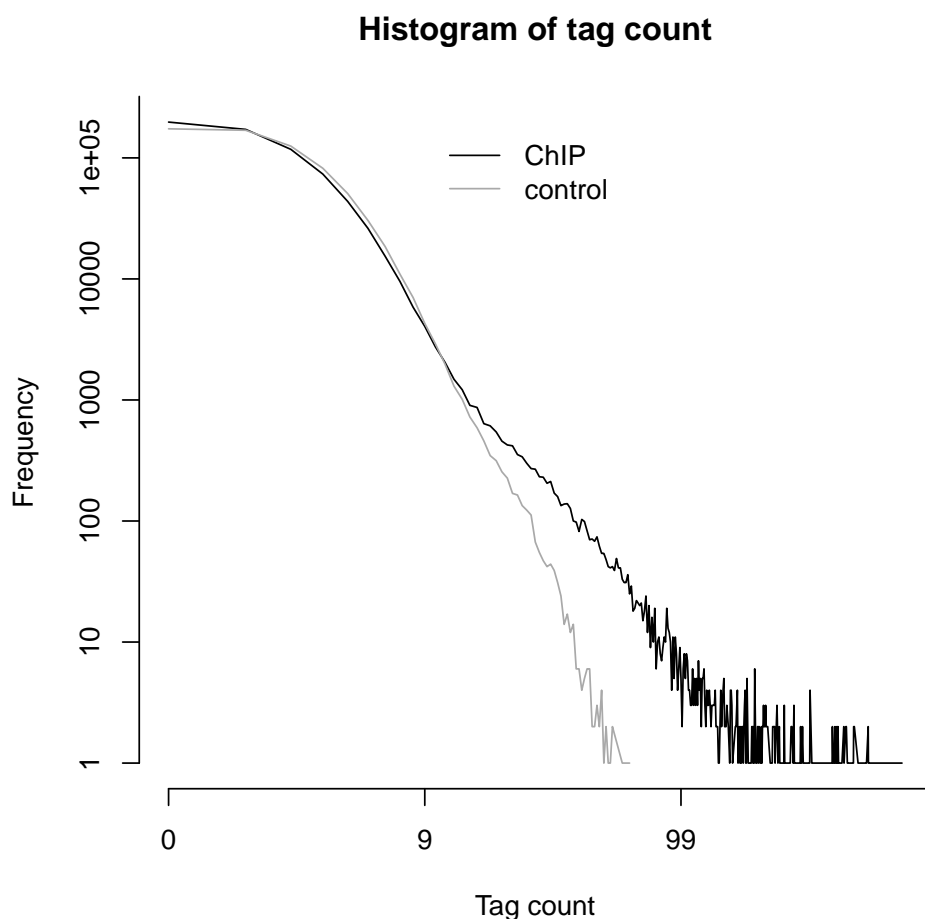


Figure 1: Histograms of the count data from ChIP and control samples.

The following command prints out BIC values of one-signal-component and two-signal-component models, with additional information about the parameters used in fitting the background (non-enriched) distribution. A lower BIC value indicates a better model fit. For this dataset, we conclude that the two-signal-component model has a lower BIC and hence it provides a better fit.

```
R> exampleFit
```

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

```
-----
analysis type: two-sample analysis (with mappability & GC content)
```

```
parameters used: k = 3, meanThres = 1, s = 2, d = 0.25
```

```
BIC of one-signal-component model = 1137784
```

```
BIC of two-signal-component model = 1135762
-----
```

'plot' method provides the GOF plot. This plots allows visual comparisons of the fits of the background, one-signal-component, and two-signal-component models with the actual data.

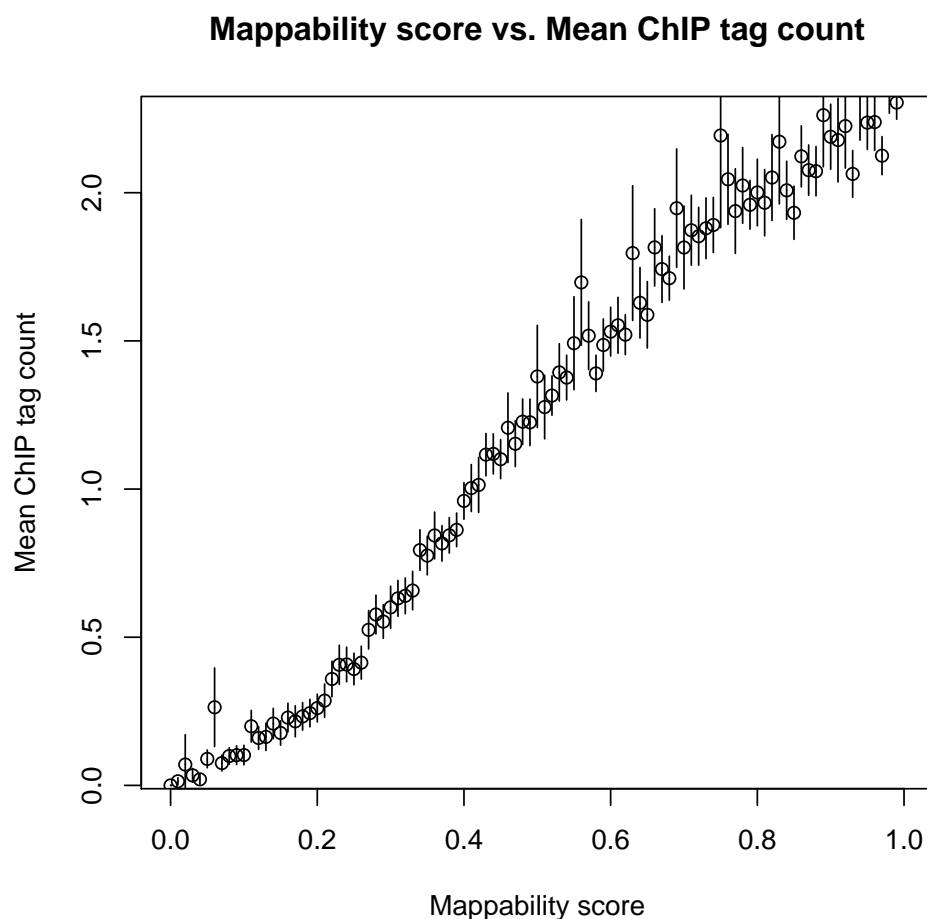


Figure 2: Mean ChIP tag count versus Mappability.

Figure 7 displays the GOF plot for our dataset and we conclude that the two-signal-component model provides a better fit as is also supported by its lower BIC value compared to the one-signal component model.

```
R> plot(exampleFit)
```

In addition to ‘analysisType’, ‘mosaicsFit’ method provides parameters to tune the background distribution of the MOSAiCS model. We specified appropriate default values for these parameters based on computational experiments and analysis of diverse ChIP-seq datasets. Default values work well in general but some tuning might be required for some cases. You may need to consider parameter tuning if the fitted background model is too similar to the actual data in the GOF plot or you encounter some warning or error messages while running ‘mosaicsFit’ method. Section 6 provides basic guidelines on parameter tuning. If you encounter a fitting problem you need help with, feel free to contact us at our Google group, http://groups.google.com/group/mosaics_user_group.

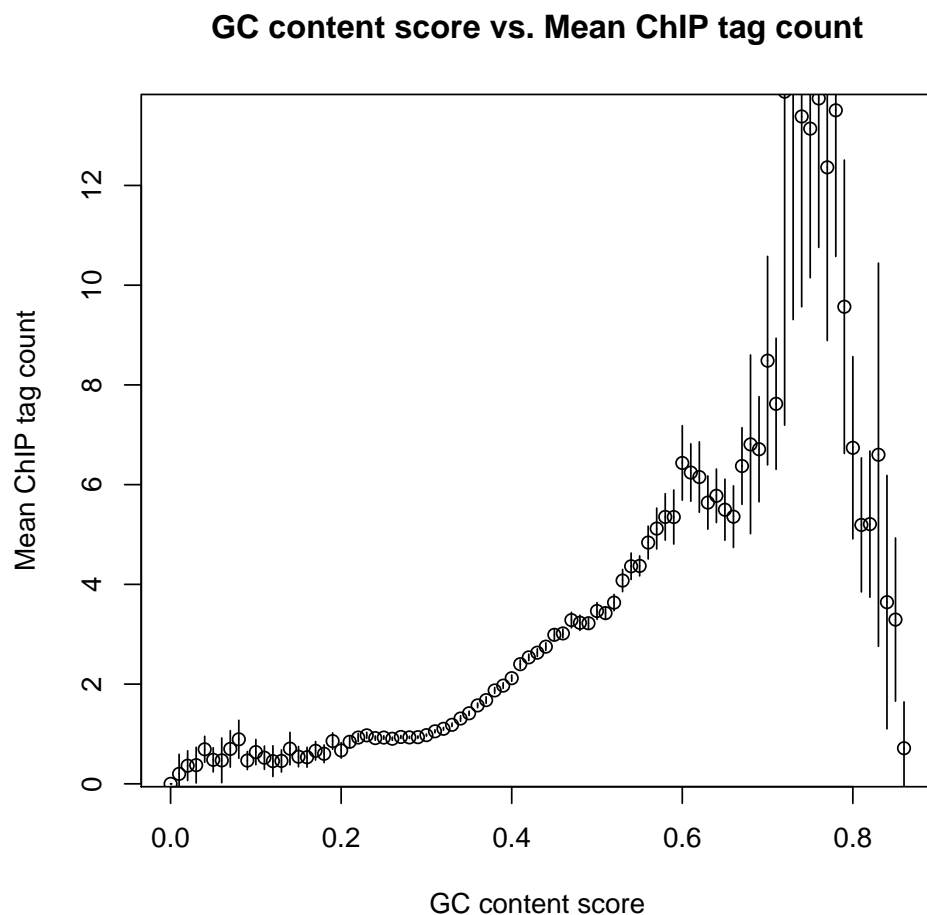


Figure 3: Mean ChIP tag count versus GC content.

3.3 Identifying Peaks Based on the Fitted Model

Using BIC values and GOF plots in the previous section, we concluded that two-signal-component model fits our data better. Next, we will identify peaks with the two-signal-component model at a false discovery rate (FDR) of 0.05 using the command:

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

‘signalModel="2S"’ indicates two-signal-component model. Similarly, one-signal-component model can be specified by ‘signalModel="1S"’. FDR can be controlled at the desired level by specifying ‘FDR’. In addition to these two essential parameters, you can also control three more parameters, ‘maxgap’, ‘minsize’, and ‘thres’. These parameters are for refining initial peaks called using specified signal model and FDR. Initial nearby peaks are merged if the distance (in bp) between them is less than ‘maxgap’. Some initial peaks are removed if their lengths are shorter than ‘minsize’ or their ChIP tag counts are less than ‘thres’.

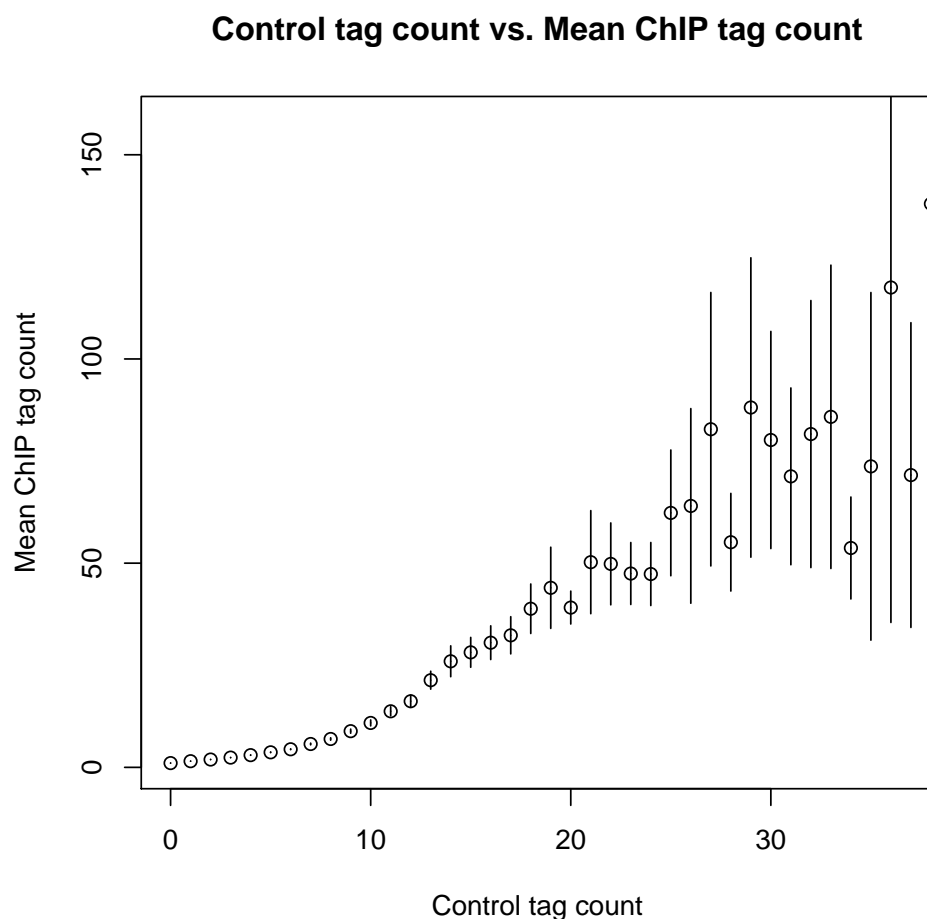


Figure 4: Mean ChIP tag count versus Control tag count.

If you use a bin size shorter than the average fragment length in the experiment, we recommend to set `'maxgap'` to the average fragment length and `'minsize'` to the bin size. This setting removes peaks that are too narrow (e.g., singletons). If you set the bin size to the average fragment length (or maybe bin size is larger than the average fragment length), we recommend setting `'minsize'` to a value smaller than the average fragment length while leaving `'maxgap'` the same as the average fragment length. This is to prevent filtering using `'minsize'` because initial peaks would already be at a reasonable width. `'thres'` is employed to filter out initial peaks with very small ChIP tag counts because such peaks might be false discoveries. Optimal choice of `'thres'` depends on the sequencing depth of the ChIP-seq data to be analyzed. If you don't wish to filter out initial peaks using ChIP tag counts, you can set `'thres'` to an arbitrary negative value.

The following command prints out a summary of identified peaks including the number of peaks identified, median peak width, and the empirical false discovery rate (FDR).

```
R> examplePeak
```

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

```
-----
```

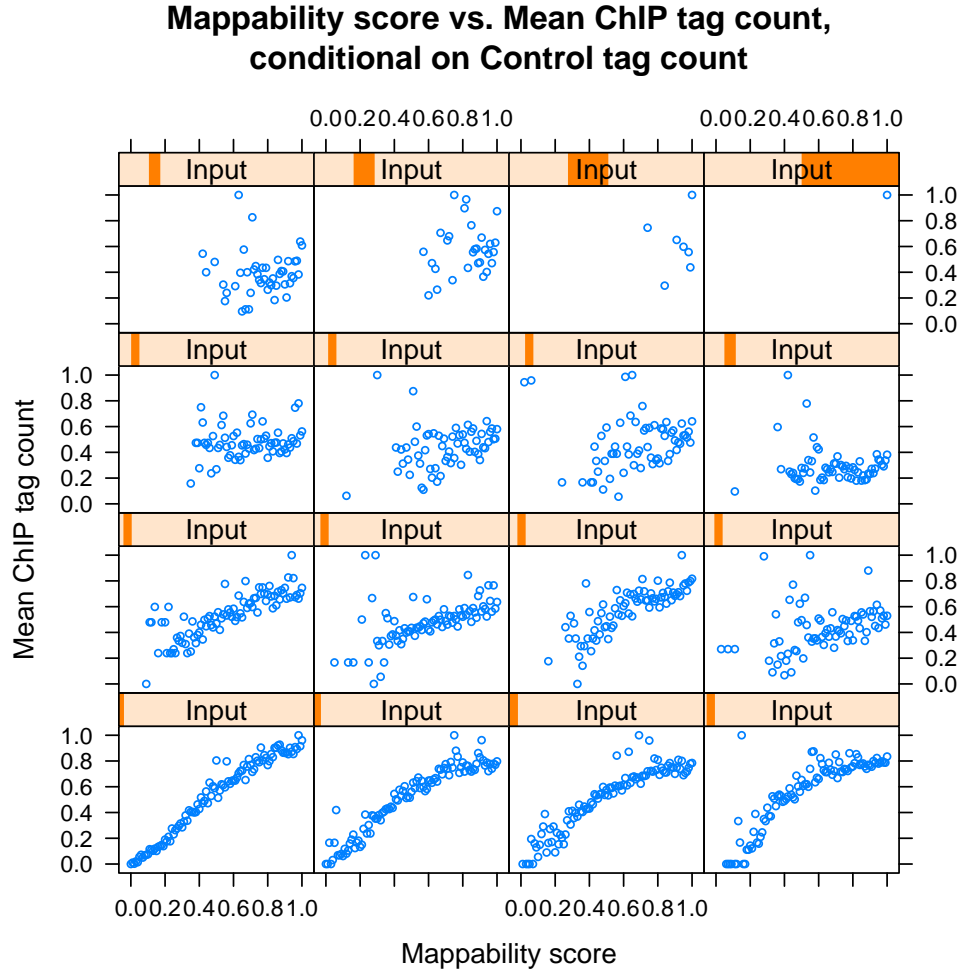



Figure 5: Mean ChIP tag count versus Mappability, conditional on control tag counts.

```
final model: two-sample analysis (with M & GC) with two signal components
setting: FDR = 0.05, maxgap = 200, minsize = 50, thres = 10
# of peaks = 520
median peak width = 250
empirical FDR = 0.05
```

‘print’ method returns the peak calling results in data frame format. This data frame can be used as an input for downstream analysis such as motif finding. This output might have different number of columns, depending on ‘analysisType’ of ‘mosaicsFit’. For example, if ‘analysisType=“TS”’, columns are peak start position, peak end position, peak width, averaged posterior probability, minimum posterior probability, averaged ChIP tag count, maximum ChIP tag count, averaged control tag count, averaged control tag count scaled by sequencing depth, averaged log base 2 ratio of ChIP over input tag counts, averaged mappability score, and averaged GC content score for each peak. Here, the posterior probability of a bin refers to the probability that the bin is not a peak conditional on data. Hence, smaller posterior probabilities provide more

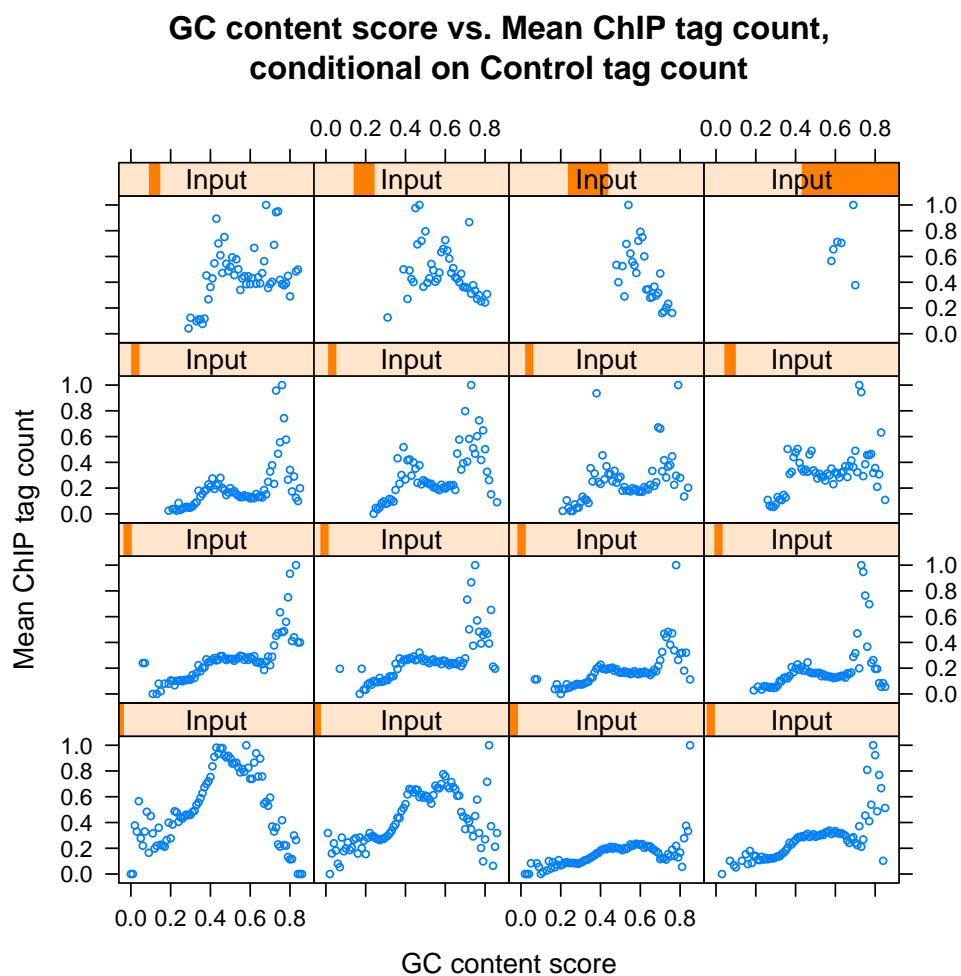


Figure 6: Mean ChIP tag count versus GC content, conditional on control tag counts.

evidence that the bin is actually a peak.

```
R> print(examplePeak)[1:15, ]
```

	chrID	peakStart	peakStop	peakSize	aveP	minP	aveChipCount
1	chr21	14538100	14538499	400	2.316159e-02	1.732184e-11	32.00000
2	chr21	14828000	14828449	450	4.683745e-02	4.537161e-05	21.77778
3	chr21	14901550	14901849	300	1.120661e-02	4.268164e-05	20.00000
4	chr21	15032250	15032499	250	2.122025e-02	4.995990e-04	15.00000
5	chr21	15068000	15068099	100	9.315207e-02	7.948542e-02	13.50000
6	chr21	15175200	15175299	100	8.070661e-02	3.541864e-02	15.50000
7	chr21	15177350	15177599	250	1.030375e-01	7.688995e-03	16.80000
8	chr21	15353150	15353549	400	2.190450e-06	9.188126e-26	81.37500
9	chr21	15362700	15362849	150	1.498505e-01	7.948542e-02	12.33333
10	chr21	15374650	15375349	700	6.933481e-05	2.139220e-65	88.28571
11	chr21	15378850	15379049	200	9.072711e-02	7.649026e-04	21.00000
12	chr21	15486500	15486799	300	3.124502e-02	5.137427e-04	33.00000

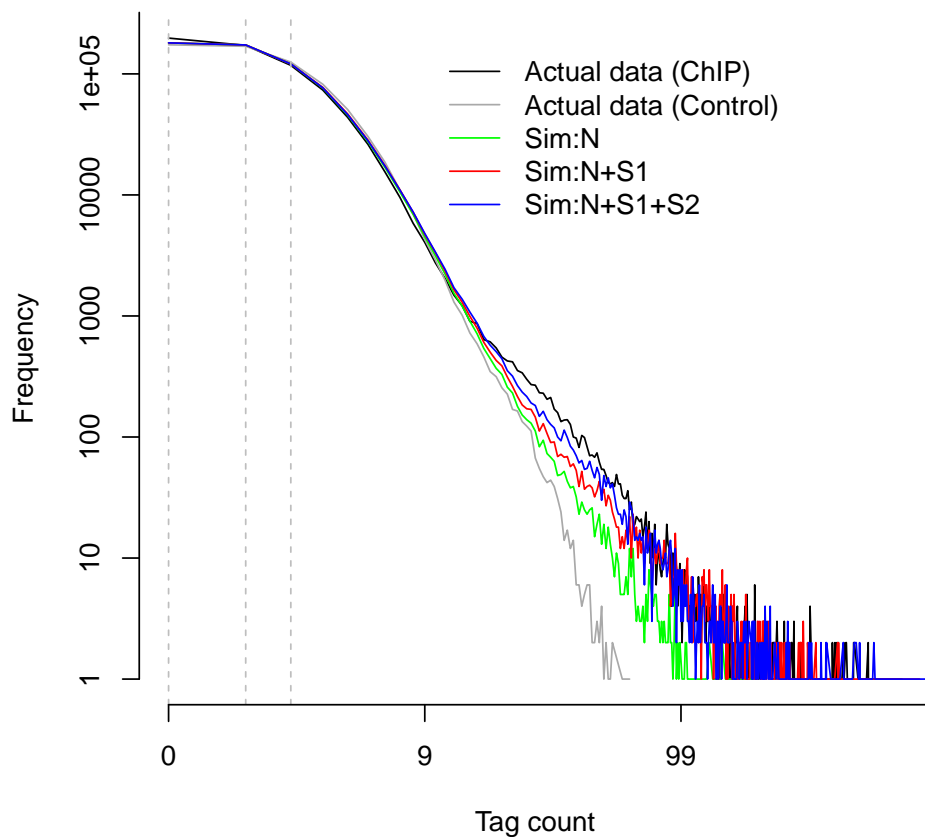


Figure 7: Goodness of Fit (GOF) plot. Depicted are actual data for ChIP and control samples with simulated data from the following fitted models: (Sim:N): Background model; (Sim:N+S1): one-signal-component model; (Sim:N+S1+S2): two-signal-component model.

13	chr21	15498300	15499149	850	3.892446e-02	5.263370e-12	48.23529
14	chr21	15501950	15502249	300	2.826662e-01	7.688995e-03	14.33333
15	chr21	15502950	15503399	450	2.188426e-03	2.830815e-49	90.66667
		maxChipCount	aveInputCount	aveInputCountScaled	aveLog2Ratio	map	
1		48	2.375000	2.662827	3.140374	0.9925000	
2		31	3.777778	4.235608	2.183571	1.0000000	
3		25	2.833333	3.176706	2.358010	1.0000000	
4		17	2.000000	2.242380	2.466431	0.9940000	
5		14	2.000000	2.242380	2.160069	1.0000000	
6		17	3.000000	3.363571	1.962167	0.9850000	
7		20	1.800000	2.018142	2.601233	0.9800000	
8		125	4.750000	5.325654	3.560025	0.9987500	
9		13	1.666667	1.868650	2.240231	1.0000000	
10		180	2.714286	3.043231	4.127678	1.0000000	

11	23	3.750000	4.204463	2.178285	0.9975000
12	40	6.666667	7.474602	2.008631	0.9833333
13	64	7.705882	8.639760	2.346157	1.0000000
14	18	3.000000	3.363571	1.829847	1.0000000
15	144	5.111111	5.730528	3.907724	1.0000000

GC

```

1 0.4250000
2 0.3788889
3 0.3933333
4 0.3320000
5 0.3900000
6 0.3800000
7 0.4540000
8 0.3787500
9 0.3800000
10 0.3557143
11 0.4325000
12 0.4016667
13 0.4311765
14 0.4333333
15 0.4644444

```

You can export peak calling results to text files in diverse file formats. Currently, ‘mosaics’ package supports 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 formats, respectively, where score is the averaged ChIP tag counts in each peak. Peak calling results can be exported in TXT, BED, and GFF file formats, respectively, by the commands:

```

R> export(examplePeak, type = "txt", fileLoc = ".", fileName = "TSpeakList.txt")
R> export(examplePeak, type = "bed", fileLoc = ".", fileName = "TSpeakList.bed")
R> export(examplePeak, type = "gff", fileLoc = ".", fileName = "TSpeakList.gff")

```

‘fileLoc’ and ‘fileName’ indicate the directory and the name of the exported file.

4 One-Sample Analysis

When control sample is not available, ‘mosaics’ package accommodates one-sample analysis of ChIP-seq data. Implementation of the MOSAiCS one-sample model is very similar to that of the two-sample analysis. Bin-level data for the one-sample analysis can be imported to the R environment with the command:

```

R> OneSampleBinData <- readBins(type = c("chip", "M", "GC", "N"),
+   fileName = c(system.file(file.path("extdata", "chip_chr21.txt"),
+     package = "mosaicsExample"), system.file(file.path("extdata",
+     "M_chr21.txt"), package = "mosaicsExample"), system.file(file.path("extdata",
+     "GC_chr21.txt"), package = "mosaicsExample"), system.file(file.path("extdata",
+     "N_chr21.txt"), package = "mosaicsExample")))

```

```
-----  
Info: preprocessing summary  
-----
```

```
- percentage of bins with ambiguous sequences: 27%  
  (these bins will be excluded from the analysis)  
- before preprocessing:  
    first coordinates = 0, last coordinates = 46944350  
- after preprocessing:  
    first coordinates = 9719550, last coordinates = 46944250  
-----
```

Note that you don't need to provide `"input"` in `'type'` and the file name of a control dataset in `'fileName'` here. In order to fit a MOSAiCS model for the one-sample analysis, you need to specify `'analysisType="OS"'` instead of `'analysisType="TS"'` when calling the `'mosaicsFit'` method.

```
R> OneSampleFit <- mosaicsFit(OneSampleBinData, analysisType = "OS")
```

Peak identification can be done exactly in the same way as in the case of the two-sample analysis.

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

5 Two-Sample Analysis Without Mappability and GC Content

Application of MOSAiCS to multiple case studies showed that consideration of mappability and GC content in the model improves sensitivity and specificity of peak identification even in the presence of a control sample [1]. However, `mosaics` package accommodates a two-sample analysis without mappability and GC content by specification of `'analysisType="IO"'` when calling the `'mosaicsFit'` method.

```
R> inputOnlyFit <- mosaicsFit(exampleBinData, analysisType = "IO")
```

You can import bin-level data (for ChIP and control sample only) and fit MOSAiCS model for the two-sample analysis without mappability and GC content with the commands:

```
R> inputOnlyBinData <- readBins(type = c("chip", "input"), fileName = c(system.file(file.path(  
+   "chip_chr21.txt"), package = "mosaicsExample"), system.file(file.path("extdata",  
+   "input_chr21.txt"), package = "mosaicsExample")))
```

```
R> inputOnlyFit <- mosaicsFit(inputOnlyBinData, analysisType = "IO")
```

6 Tuning of MOSAiCS Parameters

In the two-sample analysis, users can control three tuning parameters: `'s'`, `'d'`, and `'meanThres'`. `'s'` and `'d'` are parameters of the background distribution and control the functional form used for the control data. Please see [1] for further details on these two model parameters. `'meanThres'` controls the number of strata used at the robust linear regression modelling step of the background distribution fitting. `'mosaics'` package uses the following parameter setting as default:

```
R> exampleFit <- mosaicsFit(exampleBinData, analysisType = "TS",
+   meanThres = 1, s = 2, d = 0.25)
```

Users might need to consider parameter tuning especially when the fitted background model is too similar to the actual data, resulting in too few peaks. If such cases are detected or predicted, ‘`mosaicsFit`’ prints out warning or error messages. You may also be able to detect this case using the GOF plot. Using a higher ‘`s`’ value and lower ‘`meanThres`’ often solves the problem, e.g., ‘`s = 6`’ and ‘`meanThres = 0`’.

7 Conclusion and Ongoing Work

R package `mosaics` provides effective tools to read and investigate ChIP-seq data, fit MOSAiCS model, and identify peaks. We are continuously working on improving `mosaics` package further, especially in supporting more diverse genomes, automating fitting procedures, developing more friendly and easy-to-use user interface, and providing more effective data investigation tools. Please post any questions or requests regarding ‘`mosaics`’ package at http://groups.google.com/group/mosaics_user_group. Updates and changes of ‘`mosaics`’ package will be announced at our Google group and the companion website (<http://www.stat.wisc.edu/~keles/Software/mosaics/>).

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/).
- [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.