Girimurugan *et al. BMC Bioinformatics* (2018) 19:131
https://doi.org/10.1186/s12859-018-2140-3

BMC Bioinformatics

METHODOLOGY Open Access

iSeg: an efficient algorithm for segmentation of genomic and epigenomic data

Senthil B. Girimurugan[1†], Yuhang Liu[2†], Pei-Yau Lung[2†], Daniel L. Vera[3], Jonathan H. Dennis[4], Hank W. Bass[4] and Jinfeng Zhang[2*]

Abstract

Background: Identification of functional elements of a genome often requires dividing a sequence of measurements along a genome into segments where adjacent segments have different properties, such as different mean values. Despite dozens of algorithms developed to address this problem in genomics research, methods with improved accuracy and speed are still needed to effectively tackle both existing and emerging genomic and epigenomic segmentation problems.

Results: We designed an efficient algorithm, called iSeg, for segmentation of genomic and epigenomic profiles. iSeg first utilizes dynamic programming to identify candidate segments and test for significance. It then uses a novel data structure based on two coupled balanced binary trees to detect overlapping significant segments and update them simultaneously during searching and refinement stages. Refinement and merging of significant segments are performed at the end to generate the final set of segments. By using an objective function based on the *p*-values of the segments, the algorithm can serve as a general computational framework to be combined with different assumptions on the distributions of the data. As a general segmentation method, it can segment different types of genomic and epigenomic data, such as DNA copy number variation, nucleosome occupancy, nuclease sensitivity, and differential nuclease sensitivity data. Using simple *t*-tests to compute *p*-values across multiple datasets of different types, we evaluate iSeg using both simulated and experimental datasets and show that it performs satisfactorily when compared with some other popular methods, which often employ more sophisticated statistical models. Implemented in C++, iSeg is also very computationally efficient, well suited for large numbers of input profiles and data with very long sequences.

Conclusions: We have developed an efficient general-purpose segmentation tool and showed that it had comparable or more accurate results than many of the most popular segment-calling algorithms used in contemporary genomic data analysis. iSeg is capable of analyzing datasets that have both positive and negative values. Tunable parameters allow users to readily adjust the statistical stringency to best match the biological nature of individual datasets, including widely or sparsely mapped genomic datasets or those with non-normal distributions.

Background

High throughput genomic assays, such as microarrays and next-generation sequencing, are powerful tools for studying genetic and epigenetic functional elements at a genome scale [1]. A large number of approaches have been developed to exploit these technologies to identify and characterize the distribution of genomic and epigenomic features, such as nucleosome occupancy, chromatin accessibility, histone modifications, transcription-factor binding, replication timing, and DNA copy-number variations (CNVs). These approaches are often applied to multiple samples to identify differences in such features among different biological contexts. When detecting changes for such features, one needs to consider a very large number of segments that may undergo changes, and robustly calculating statistics for all possible segments is usually not feasible. As a result, heuristic algorithms are often needed to find the optimal solution for the objective function adopted by an approach. This problem is often called segmentation problem in the field of genomics, and change-point problem in other scientific disciplines.

* Correspondence: jinfeng@stat.fsu.edu
†Equal contributors
[2]Department of Statistics, Florida State University, Tallahassee, FL, USA
Full list of author information is available at the end of the article

Solving the segmentation problem typically involves dividing a sequence of measurements along the genome such that adjacent segments are different for a predefined criterion. For example, if segments without changes have a mean value of zero, then the goal could be to identify those segments of the genome whose means are significantly above or below zero. A large number of such methods have been developed for different types of genomic and epigenomic data [2–17]. Many methods are designed for specific data types or structures, but it is challenging to find versatile programs for data with different properties and different underlying statistical assumptions. The previous methods fall into several categories including change-point detection [2, 3, 9, 10, 12, 14, 18–25], Hidden Markov models [5, 15, 26–28], Dynamic Bayesian Network (DBN) models [29, 30], signal smoothing [31–34], and variational models [35, 36]. For review and comprehensive comparison, please refer to [16, 37–40].

Many currently available segmentation tools have poor performance, run slowly on large datasets, or are not straightforward to use. To address these challenges, we developed a general method for segmentation of sequence-indexed genome-wide data. Our method, called iSeg, is based on a simple formulation of the optimization problem [7]. Assuming the significance (i.e. *p*-value) of segments can be computed based on certain parametric or non-parametric models, iSeg identifies the most significant segments, those with smallest *p*-values. Once the segment with the smallest *p*-value is found, it will be removed from the dataset and segment with the second smallest p-value will be searched in the remaining of the data. The procedure repeats until no segments whose significance levels pass a predefined threshold. This simple objective function is intuitive from a biological perspective since the most statistically significant segments often biologically significant.

iSeg has several noteworthy features. First, the simple formulation allows it to serve as a general framework to be combined with different assumptions of underlying probability distributions of the data, such as Gaussian, Poisson, negative Binomial, or non-parametric models. As long as *p*-values can be calculated for the segments, the corresponding statistical model can be incorporated into the framework. Second, iSeg is a general segmentation method, able to deal with both positive and negative signals. Many of the existing methods, cannot deal with negative values in the data, because they are designed specifically for certain data types, such as genome-wide read densities, assuming data values with only zero or non-negative values. Negative values in genomic datasets can occur when analyzing data pair relationships, such as difference values or log2 ratios commonly used with fold-change analysis. Currently, most methods segment profiles separately and compare the resulting segmentations. The drawback of such treatment is that peaks with different starting and ending positions from different segmentations cannot be conveniently compared, and peaks with different magnitudes may not always be distinguished. Taking differences from two profiles to generate a single profile overcomes these drawbacks. Third, to deal with cases where segments are statistically significant, but the biological significance may be weak, we apply biological significance threshold to allow practitioners the flexibility to incorporate their domain knowledge when calling "significant" segments. Fourth, iSeg is implemented in C++ with careful design of data structures to minimize computational time to accommodate, for example, multiple or very long whole genome profiles. Fifth, iSeg is relatively easy to use with few parameters to be tuned by the users.

Here we describe the method in detail, followed by performance analysis using multiple data types including simulated, benchmark, and our own. The data types include DNA copy number variations (CNVs) for microarray-based comparative genomic hybridization (aCGH) data, copy number variations from next generation sequencing (NGS) data, and nucleosome occupancy data from NGS. From these tests, we found that iSeg performs at least comparably with popular, contemporary methods.

Methods

Problem formulation

We adopted a formulation of the segmentation problem from previous methods [7]. The goal was to find segments with statistical significance higher than a predefined level measured by *p*-values under a certain probability distribution assumption. The priority was given to segments with higher significance, meaning that the segment with highest significance was identified first, followed by the one with the second highest significance, and so on. We implemented the method using Gaussian-based tests (i.e. *t*-test and *z*-test), as used it in other existing methods [3, 7, 9]. Our method achieved satisfactory performance for both microarray and next generation sequencing data without modifying the hypothesis test. A more common formulation of the change-point problems is given in [41].

Consider a sample consisting of N measurements along the genome in a sequential order, $X1, X2, \ldots, XN$, and

$$X_k \sim N\left(\mu_0, \sigma^2\right), \forall k \in L$$

$$X_k \sim N\left(\mu_i, \sigma^2\right), \forall k \notin L$$

for some set of locations L (i.e. background, regions with no changes, etc.). The common assumption is that there are M non-overlapping segments with mean $\mu_1, \mu_2, \ldots, \mu_i, \ldots, \mu_M$, where $\mu_i \neq \mu_0$, and the union of these segments will form the complement of the set L. If the background level, μ_0, is non-zero, the null hypothesis can be the corresponding non-zero means. According to this model, it is possible for multiple segments with means different from μ_0 to be adjacent to each other. In addition, all the measurements are assumed to be independent. This assumption has been employed in many existing methods [9, 23]. A summary of existing methods that use such an i.i.d. assumption and its properties are discussed in [42]. The goal of a segmentation method is to detect all the M segments with means different from μ_0.

To illustrate, Figure 2a shows segments generated from Normal distributions with non-zero means where the rest of the data is generated from a standard Normal distribution. There are two computational challenges associated with the approach we are taking that also manifest in many previous methods. First, the number of segments that are examined is very large. Second, the overlaps among significant segments need to be detected so that the significance of the overlapping segments can be adjusted accordingly. To deal with the first challenge, we applied dynamic programming combined with exponentially increased segment scales to speed up the scanning of a large sequence of data points. To deal with the second challenge, we designed an algorithm coupling two balanced binary trees to quickly detect overlaps and update the list of the most significant segments. Segment refinement and merging allow iSeg to detect segments of arbitrary length.

Computing *p*-values using dynamic programming

iSeg scans a large number of segments starting with a minimum window length, W_{min}, and up to a maximum window length, W_{max}. The minimum and maximum window lengths have default values of 1 and 300, respectively. This window length increases by a fixed multiplicative factor, called power factor (ρ), with every iteration. For example, the shortest window length is W_{min}, and the next shortest window length would be ρW_{min}. The default value for ρ is 1.1. When scanning with a particular window length, W, we use overlapping windows with a space of $W/5$. When 'W' is not a multiple of 5, numerical rounding (**ceiling**) is applied. The aforementioned parameters can be changed by a user. We found the default parameters work robustly for all the datasets we have worked with. The algorithm computes p-values for candidate segments and detects a set of non-overlapping segments most significant among all possible segments.

Given the normality assumption, a standard test for mean is the one-sample student's t-test, which is commonly found among many existing methods. The test statistic for this test is,

$$t = \frac{\overline{x}\sqrt{n}}{s}$$

where $\overline{x}$ is the sample mean, s is the sample standard deviation, and n is the sample size. A drawback of this statistic is that it cannot evaluate segments of length 1. This may be the reason that some of the previous methods are not good at detecting segments of length 1. Although we can derive a test statistic separately for segments of length 1, the two statistics may not be consistent. To solve this issue, we first estimate the sample standard deviation using median absolute deviation (**MAD**), assuming that the standard deviation is known. Specifically, for a series of data points $x_1, x_2, \ldots, x_n$, the MAD is defined as the median of the absolute deviations from the data's median:

$MAD = median(|\, x_i - median(\boldsymbol{x})|\,)$.

This is a robust summary statistic of the variability of the data. This allows us to use z-statistic instead of t-statistic and the significance of single points can be evaluated based on the same model assumption as longer segments. To calculate sample means for all segments to be considered for significance, the number of operations required by a brute force approach is 'C_b'.

$$C_b = \sum_{i=0}^{k} \left(N - \rho^i W_{\min}\right) \rho^i W_{\min}$$

where, $\rho k W min \leq W max$ and $\rho k + 1 W min > W max$.

Computation of these parameters (means and standard deviations) for larger segments can be made more efficiently by using the means computed for shorter segments. For example, the running sum of a shorter segment of length 'm' is given by,

$$S_m = \sum_{i=l}^{m} X_i.$$

If this sum is retained, the running sum of a longer segment of length r ($r > m$) in the next iteration can be obtained as,

$$S_r = S_m + \sum_{i=m+1}^{r} X_i,$$

and the means for all the segments can be computed using these running sums. Now, the total number of operations (C_b^*) is

$$C_b^* = N + \sum_{i=0}^{k} \left(N - \rho^i W_{\min}\right),$$

which is much smaller in practice than the number of operations (C_b) without using dynamic programming. Computation of standard deviations is sped up using a similar process.

Detecting overlapping segments and updating significant segments using coupled balanced binary trees

When the *p*-values of all the segments are computed, we rank the segments by their *p*-values from the smallest to the largest. All the segments with p-values smaller than a threshold value, p_s, are kept in a balanced binary tree (BBT1). The default value of p_s is set as 0.001. Assuming a significance level (α) of 0.1, 100 simultaneous tests will maintain a family-wise error rate (FWER) bounded by 0.001 with Bonferroni and Sidak corrections. Thus, the cut-off is an acceptable upper bound for multiple testing. It can be changed by a user if necessary. The procedure for overlapping segment detection is described below as a pseudo-code. The set BBT1 stores all significant segments passing the initial significance level cutoff (default value 0.001). The second balanced binary tree (BBT2) stores the boundaries for significant segments. After the procedure, SS contains all the detected significant segments. The selection of segments using balanced binary tree makes sure that segments with small *p*-values will be kept, while those overlapping ones with bigger *p*-values will be removed.

Refinement of significant segments

The significant segments are refined further by expansion and shrinkage. Without loss of generality, in the procedure (see SegmentExpansion text box) we describe expansion on left side of a segment only. Expansion on the right side and shrinkage are done similarly. When performing said expansion and shrinkage, a condition to check for overlapping segments is applied so the algorithm results in only disjoint segments.

Merging adjacent significant segments

When all the significant non-overlapping segments are detected and refined in the previous steps, iSeg performs a final merging step to merge adjacent segments (no other significant segments in between). The procedure is straightforward. We check each pair of adjacent segments. If the merged segment, whose range is defined by the left boundary of the first segment and the right boundary of the second segments, has a *p*-value smaller than those of individual segments, then we merge the two segments. The new segment will then be tested for merging with its

```
procedure SelectSignificantSegments
initialize BBT2 // BBT2 is empty at the beginning
while(BBT1 not empty)
        $S$ = top ranked segment in BBT1 (smallest p-value among all segments in BBT1)
        delete $S$ from BBT1
        $l$ = left boundary of $S$
        $r$= right boundary of $S$
        if(checkoverlap (BBT2, $l$, $r$) == FALSE) // no overlapping
                insert pair($l$, $r$) into BBT2
                insert $S$ to set SS
```

```
procedure SegmentExpansion ($S_{l,r}$)
/* $S_{l,r}$: the segment to be expanded. Its left boundary is $l$, and right boundary is $r$. */
$S = S_{l,r}$
while()
        $p$ = p-value of $S$
        $L$ = length of $S$
        $l_0$ = left boundary of $S$
/* expand the segment by 1/K of its current length, and compute its p-value. The default value
for K is 10. */
        $l' = l_0 - ceiling(L/K)$
        $p'$ = p-value of segment $S_{l',r}$
        if $p' < p$
                $S = S_{l',r}$
        Else
                compute p-values for all segments with left boundary in ($l'$, $l_0$) and right boundary
                $r$. let $p_m$ be the minimum p-value of these segments, and $l_m$ be the corresponding
                left boundary
                if $p_m < p$
                        $S = S_{lm,r}$
                Break
Update $S_{l,r}$ with boundaries of $S$.
```

adjacent segments iteratively. The procedure continues until no segments can be merged. With refinement and merging, iSeg can detect segments of arbitrary length—long and short. We added an option to merge only segments whose distances are no more than certain threshold, where distances are measured by the difference of the ending position of the first segment and the starting position of the second segment.

Multiple comparisons

In iSeg, *p*-values for potentially significant segments are calculated. Using a common p-value cutoff, for example 0.05, to determine significant segments can suffer from a large number of false positives due to multiple comparisons. To cope with the multiple comparisons issue, which can be very serious when the sequence of measurements is long, we use a false discovery rate (FDR) control. Specifically, we employ the Benjamini-Hochberg (B-H) procedure [43] to obtain a cutoff value for a predefined false discovery rate (α), which has a default value of 0.01, and can also be set by a user. Other types of cutoff values can be used to select significant segments, such as a fixed number of most significant segments.

Biological cutoff

Often in practice, biologists prefer to call signals above a certain threshold. For example, in gene expression analysis, a minimum of two-fold change may be applied to call differentially expressed genes. Here we add a parameter, *bc*, which can be tuned by a user to allow more flexible and accurate calling of significant segments. The default output gives four *bc* cutoffs: 1.0, 1.5, 2.0 and 3.0. Biological cutoff value 1.0 means that the height of a segment has to be greater than 1.0*standard deviation of the data for it to be called as significant, regardless of the length of the segment. The biological cutoff parameter allows users to select significant segments, whichever are more likely to be biological significant based on their knowledge of the problem they are studying.

In Fig. 1, we provide a schematic illustration of the iSeg workflow.

Processing of the raw NGS data from maize

Raw fastq files were clipped of 3′ illumina adapters with cutadapt 1.9.1. Reads were, aligned to B73 AGPv3 [44] with bowtie2 v2.2.8 [45], alignments with a quality < 20 were removed, and fragment intervals were generated with bedtools (v2.25) bamtobed [46]. Fragments were optionally subset based on their size. Read counts in 20-bp nonoverlapping windows across the entire genome were calculated with bedtools genomecov and normalized for sequencing depth (to fragments-per-million, FPM). Difference profiles were calculated by subtracting heavy FPM from light FPM. Quantile normalization was performed using the average score distributions within a given combination of digestion level (or difference) and fragment size class.

Results

We compared our method with several previous methods for which we were able to obtain executable programs: HMMSeg [5], CGHSeg [10], DNAcopy [9, 24], fastseg [3], cghFLasso [34], BioHMM-snapCGH [27], mBPCR [12], SICER [47], PePr [48] and MACS [49]. Among them, CGHSeg, DNAcopy, BioHMM-snapCGH, mBPCR and cghFLasso are specifically designed for DNA copy number variation data; MACS, SICER and PePr are designed for ChIP-seq data; and HMMSeg is a general method for segmentation of genomic data. Each method has some parameters that can be tuned by a user to achieve better performance. In our comparative study, we carefully selected parameters on the basis of the recommendations provided by the authors of the methods. For each method

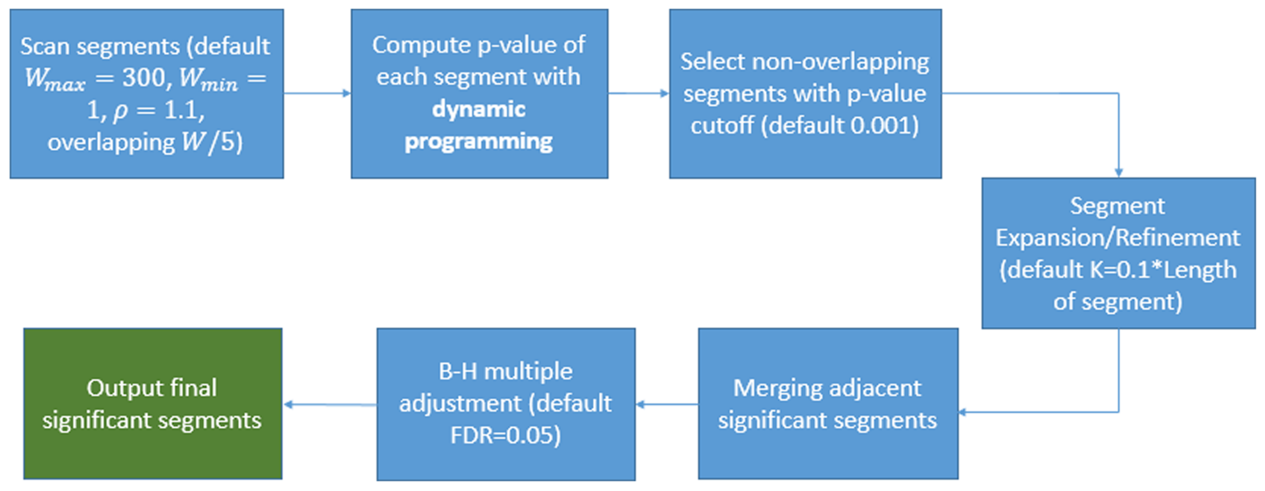

Scan segments (default $W_{max} = 300, W_{min} = 1, \rho = 1.1$, overlapping $W/5$)
Compute p-value of each segment with **dynamic programming**
Select non-overlapping segments with p-value cutoff (default 0.001)
Segment Expansion/Refinement (default K=0.1*Length of segment)
Merging adjacent significant segments
B-H multiple adjustment (default FDR=0.05)
Output final significant segments

Fig. 1 A schematic illustration of the workflow of iSeg

including iSeg, a single set of parameters is used for all data sets except where specified. Post-processing is required by some of the methods to identify significant segments.

In our analysis, performance is measured using F_1-scores[46] for all methods. F_1-scores are considered as a robust measure for classifiers because they account for both precision and recall in their measurement. The F_1-score is defined as,

$$F_1 = (2pr)/(p + r),$$

where p is precision and r is recall for a classifier. In terms of the true (TP) and false (FP) positives,

$$p = TP/(TP + FP),$$
$$r = TP/(TP + FN).$$

The methods CGHSeg, DNAcopy, and fastseg depend on random seeds given by a user (or at run-time automatically), and the F_1-score at different runs are very similar but not the same. These methods were run using three different random seeds. The averages of the F_1-score were used to measure their performance.

Performance on simulated data

The simulated profiles were generated under varying noise conditions, with signal to noise ratios (SNR) of 0.5, 1.0 and 2.0, which correspond to poor, realistic and best case scenarios, respectively. Ten different profiles of length 5000 were simulated.

For each profile, five different segments of varying lengths were predefined at different locations. Data points outside of these segments were generated from normal distribution with mean zero. The five segments were simulated with non-zero means and varying amplitudes, or ease of detection, in order to assess the robustness of the methods. Because this set of simulated data resembles more of the DNA copy-number variation data, we used it to compare iSeg to methods designed for DNA copy number data. Figure 2 shows an example of the simulated data and the segments identified by iSeg and other existing methods. Figure 3a shows the performance of iSeg and other methods on simulated data with SNR = 1.0. We can see that iSeg, DNACopy and CGHSeg perform similarly well, with HMMseg and CGHFLasso performing a little worse while fastseg did not perform as well as the other methods. iSeg is also tested using a set of 10 longer simulated

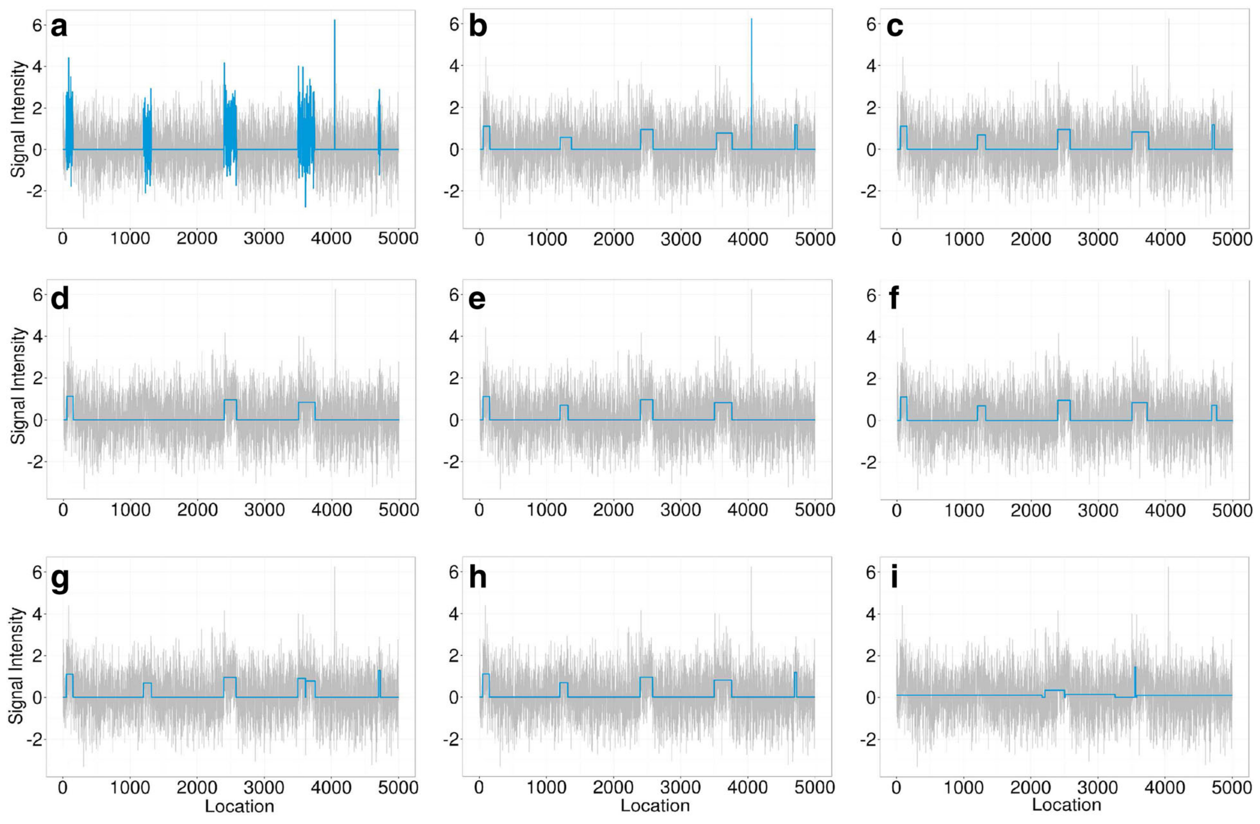


Fig. 2 One of the simulated profiles and its detected segments obtained using iSeg. **a** The actual data with background noise and predefined segments. The segments with non-zero means are normally distributed with unit variance and means 0.72, 0.83, 0.76, 0.9, 0.7, and 0.6 respectively. The profiles shown here are normalized for an approximate signal to noise ratio of 1.0. The segments detected by iSeg (**b**) and other existing methods: snapCGH (**c**), mBPCR (**d**), cghseg (**e**), cghFLasso (**f**), HMMSeg (**g**), DNAcopy (**h**) and fastseg (**i**)

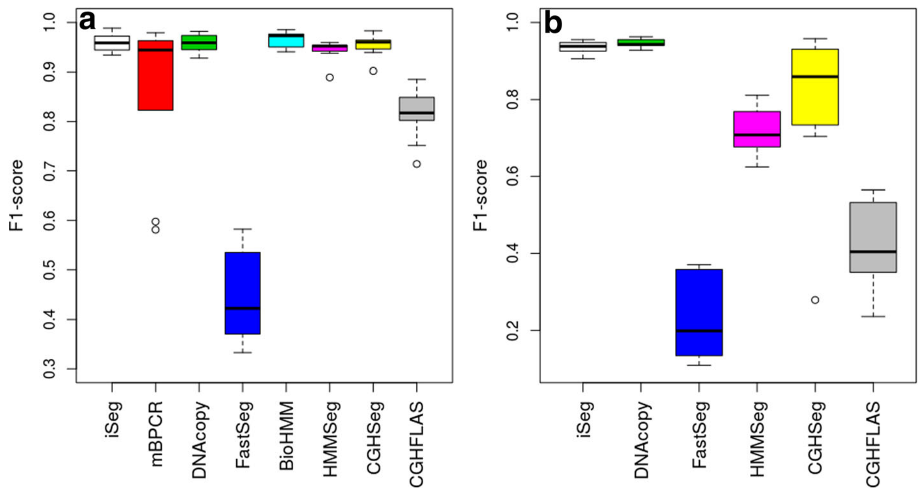


Fig. 3 Comparison of F_1-scores of various methods in analyzing different types of profiles. **a** 10 Simulated Profiles; **b** 10 Simulated Long Profiles ($n = 100$ K). BioHMM: BioHMM-SnapCGH; CGHFLAS: CGHFLasso

profiles, each with length 100,000. Seven segments are introduced at varying locations along the profiles. iSeg still performs quite well in these very long profiles. The performance of these methods on long sequences is shown in Fig. 3b. In Fig. 4, we plotted Venn Diagrams of the overlapping called locations for several methods with better performance in terms of F_1-scores, for both the simulated short profiles (Fig. 4a) and long profiles (Fig. 4b). We can see that in the simulated profiles, there is a substantial overlap among most methods, while iSeg can detect more segments for relatively short profiles. In simulated long profiles, all methods, except HMMSeg, have similar segmentation results.

Performance on experimental data

DNA copy-number variation (CNV) data

To assess the performance of iSeg on experimental data, we use three different datasets. They were the Coriell dataset [50] with 11 profiles, the BACarray dataset [51] with three profiles, and the dataset from The Cancer Genome Atlas (TCGA) with two profiles. The 11 profiles in Coriell datasets correspond to 11 cell lines: GM03563, GM05296, GM01750, GM03134, GM13330, GM01535, GM07081, GM13031, GM01524, S0034 and S1514. We constructed "gold standard" annotations using a consensus approach. We first ran all the methods using several different parameter settings for each method. The resulting segments from all the different parameter settings of all the methods were combined to give an initial set of potential segments. The test statistics and p-values are then calculated for all the segments using the same probability distribution assumption described in Method. Benjamini-Hochberg procedure was then used to correct for multiple comparison. The 0.05 adjusted p-value cutoff was then used to select the set

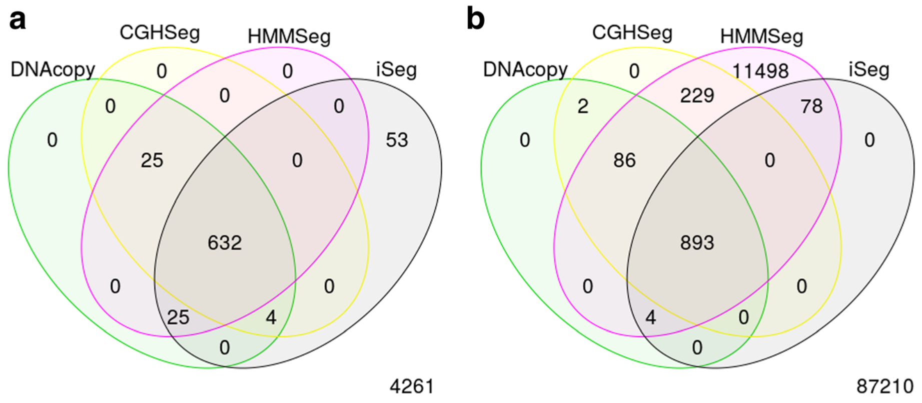


Fig. 4 Venn Diagram of overlapping regions (in number of biological coordinates) called by well-performed methods (in terms of F_1-scores) in different profiles. **a** Simulated Profiles; **b** Simulated Long Profiles ($n = 100$ K)

of segments as the gold standard. The annotations derived using the consensus approach are provided as Additional file 1.

The 11 profiles from the Coriell dataset were segmented using iSeg and the other methods. Segmentation result for one of the profiles is shown in Fig. 5, and the F_1-scores are shown in Fig. 6a. The performance of iSeg is robust with accuracy above 0.75 for all the profiles from this dataset and it was found to be comparable to or better than other methods. For HMMSeg, both no-smoothing and smoothing were used. The best smoothing scale for HMMSeg was found to be 2 for the Coriell dataset. In Fig. 5, we found that iSeg identified most of the segments. DNAcopy, fastseg, HMMSeg and cghseg missed single-point peaks, whereas cghFLasso, mBPCR and snapCGH missed some of the longer segments. The segmentation results for other profiles in Coriell dataset can also be found in the Additional file 1. We generated annotations using the consensus method for BACarray dataset similar to the Coriell dataset. The comparison of segmentation results for one profile of the BACarray dataset is shown in Fig. 7, and the comparison of F_1-scores is shown in Fig. 6b. iSeg returned better F1-scores than the other methods, consistent with the conclusions based on visual inspection.

For the TCGA datasets, since the profiles are rather long, we did not generate annotations using the consensus approach. Instead, we applied some of the methods to this dataset and compared their segmentation results visually (Fig. 8). Again, we found that iSeg identified most of the significant peaks. In this test, DNACopy performed well overall, but tended to miss some of the single-point peaks, whereas other methods performed even less well.

We compared the computational time of iSeg, shown in Table 1, with those of the other methods, and found that iSeg is the fastest for the three test datasets. Notably, iSeg took much less time than the other methods for very long profiles (length 100,000). This speed is achieved in part through dynamic programming and a power factor that provides rapid initial scanning of the profiles. The long profiles contain similar amount of data points that are signals (as opposed to background or noise) as the shorter profiles. The time spent on dealing with potentially significant segments is roughly the same between the two types of profiles. As a result, the overall running time of iSeg for the long profiles did not

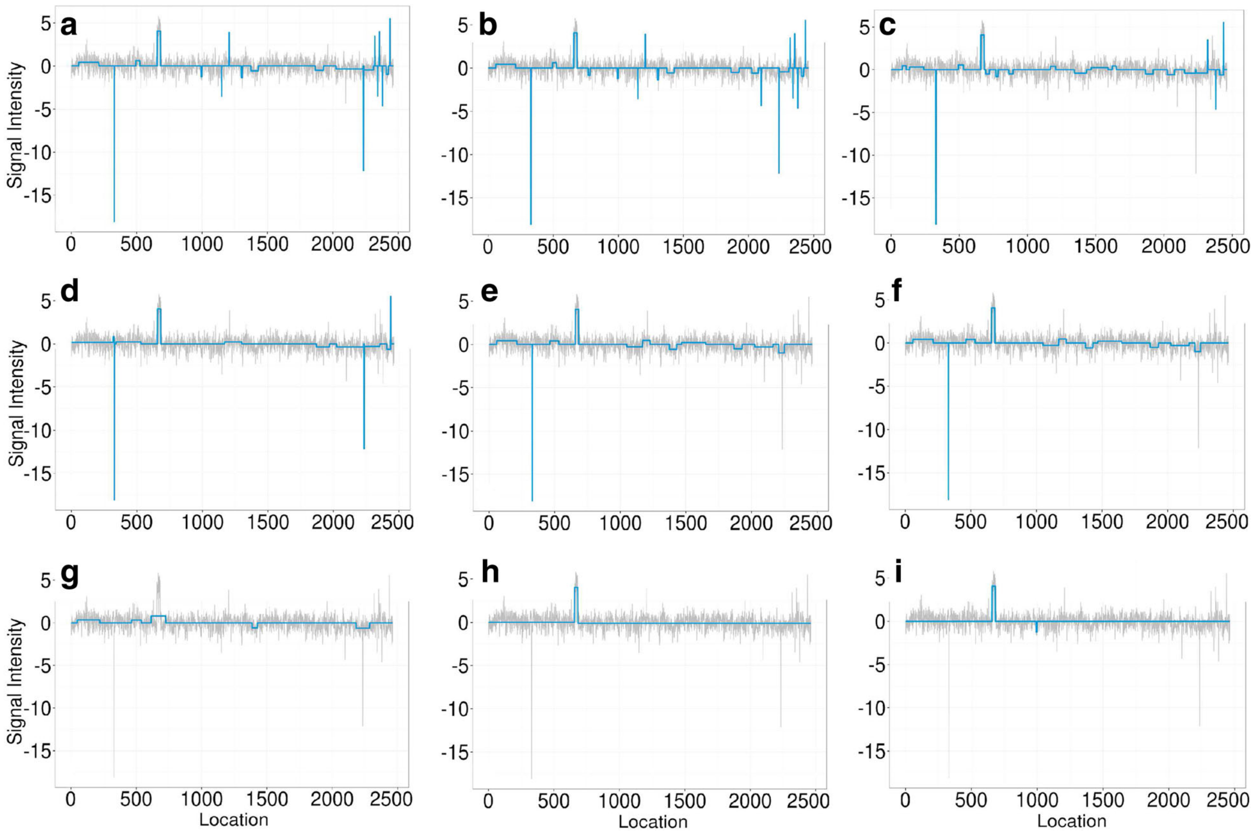


Fig. 5 Comparison of segmentation results for one of the Coriell datasets. **a** The gold standard segmentation obtained using a consensus approach. Segmentation results of iSeg (**b**) and other existing methods: snapCGH (**c**), mBPCR (**d**), cghseg (**e**), cghFLasso (**f**), HMMSeg (**g**), DNAcopy (**h**) and fastseg (**i**)

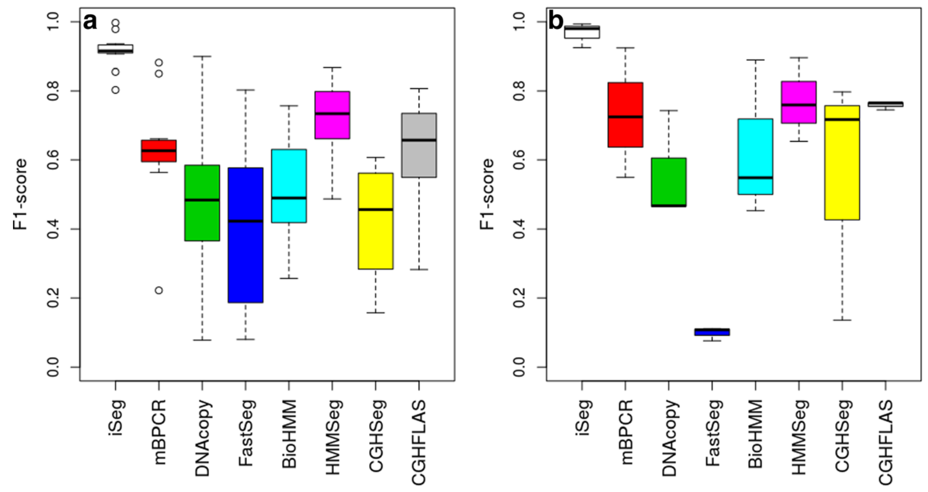


Fig. 6 Comparison of F_1-scores of various methods in analyzing different types of profiles. **a** Coriell (Snijders et al.) profiles; **b** BACarray profiles

increase as much as that for the other methods. In summary, we observed that iSeg ran faster than the other methods, especially for profiles with sparse signals.

Differential nuclease sensitivity profiling (DNS-seq) data

We then tested our method on next generation sequencing data, for which discrete probability distribution models have been used in most of the previous methods. The dataset profiles were genome-wide reads from light or heavy digests (with zero or positive values) or difference profiles (light minus heavy, with positive or negative values) [51]. The difference plots are also referred to as sensitivity or differential nuclease sensitivity (DNS) profiles [51].

Segmentation of single nuclease sensitivity profiles

Figure 9 shows the significant segments (peaks) called by iSeg together with those of two other methods, MACS and SICER. Visual inspection revealed that iSeg successfully segmented clear peaks and their boundaries. The performance of iSeg was at least comparable to MACS and SICER. Follow-up analyses on the segmentation results also demonstrated its capability in identifying biological interesting functional regions [51, 52]. iSeg results with different biological significance cutoffs (BC) are displayed as genome browser tracks beside the input profile data to guide inspection of the segmentation results. The default is to output three BCs: 1.0, 2.0, and 3. 0, which are sufficient for most applications.

Segmentation of difference profiles with both positive and negative values

Difference profiles between two conditions can be generated by subtracting one profile from the other at each genomic location. Pairwise comparisons are of great interest in genomics, as they allow for tests of differences within replicates, or across treatments, tissues, or

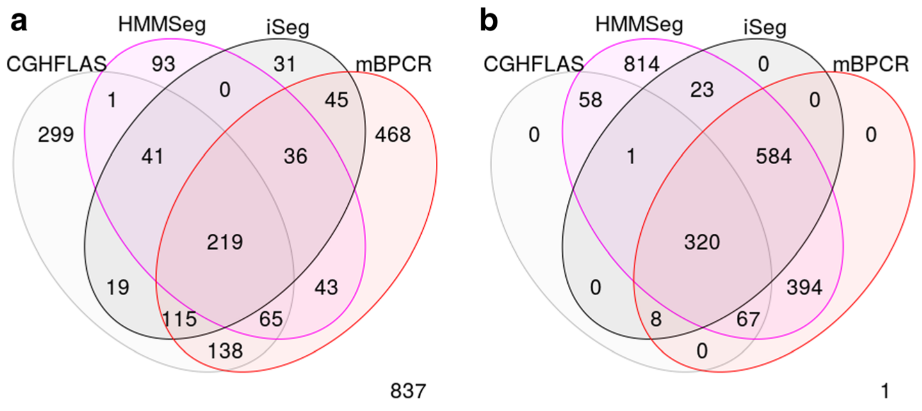


Fig. 7 Venn Diagram of overlapping regions (in number of biological coordinates) called by well-performed methods (in terms of F_1-scores) in different profiles. **a** Coriell (Snijders et al.) profiles; **b** BACarray profiles

Table 1 Comparison of computational times (in seconds) on simulated data and Coriell data. These are total times required to process 10 simulated and 11 Coriell profiles

Method	Simulation (SNR ≃ 1.0, $n = 5000$)	Simulation (SNR ≃ 1.0, $n = 100$ K)	Coriell
iSeg (C++)	0.164	1.223	0.294
DNAcopy (R)	2.267	60.343	3.098
Fastseg (R)	0.647	48.139	0.630
CGHSeg (R)	54.480	157.626	24.36
HMMSeg (Java)	0.543	160.790	0.552

genotypes. Analyzing such profiles can preserve the range or magnitude of differences, adding power to detect subtle differences between two profiles, compared to approaches that rely on calling peaks in the two files separately. Figure 10 shows the segmentation of iSeg on a typical DNS-seq profile. When analyzing these sets of DNS data, we merge segments only when they are consecutive, meaning the gap between the two segments is zero. The length of gaps between adjacent segments that can be merged is a parameter tunable by users. iSeg successfully identified both positive (peak, positive peak) and negative (valley, negative peak) segments, as shown in Fig. 11. Most existing ChIP-seq data analysis methods do not accommodate this type of data as input. To run MACS, SICER, and PePr, we assigned the light and heavy digestion read profiles as the treatment and control files respectively, as a fair way of comparison. Since the true biological significant segments are unknown, we compared the methods through careful visual inspections by domain experts. We found that iSeg performed satisfactorily and select it as the method of choice for analyzing the data from our own labs. The choice of parameters of competing methods is somehow subjective. We always start from the default values, and then do some tuning of each parameter while making the others fixed. Each time we will do a careful visualization until the final set of calls look reasonably satisfactory.

Discussion

In this study, we designed an efficient method, iSeg, for the segmentation of large-scale genomic and epigenomic profiles. When compared with existing methods using both simulated and experimental data, iSeg showed comparable or improved accuracy and speed. iSeg performed equally well when tested on very long profiles, making it suitable for deployment in real-time, including online webservers able to handle large-scale genomic datasets.

In this study, we have assumed that the data follow a Gaussian (normal) distribution. The algorithm is not limited, however, to this distribution assumption. Other hypothesis tests, such as Poisson, negative binomial, and non-parametric tests, can be used to compute *p*-values for the segments. Data generated by next-generation sequencing

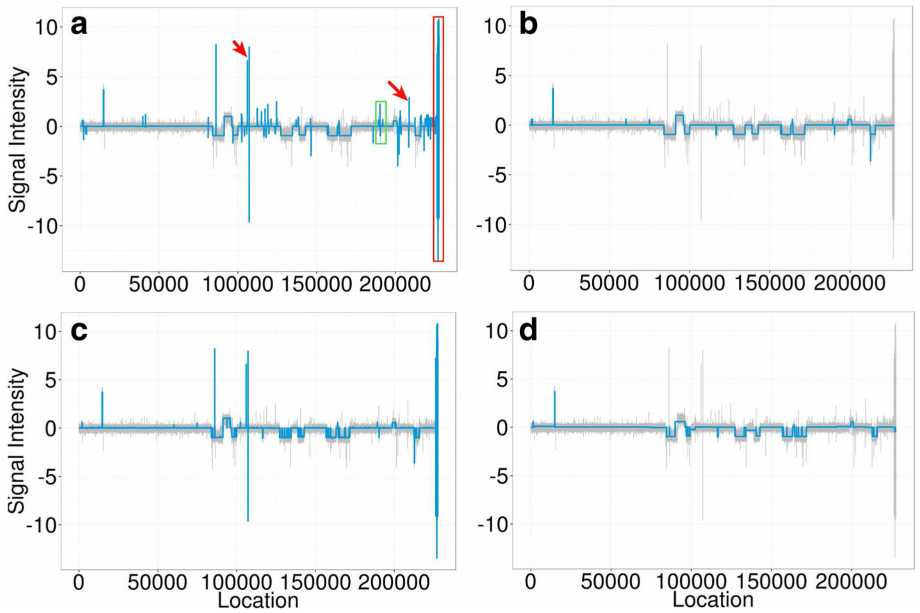


Fig. 8 Comparison of segmentations for the TCGA dataset. The patient profile ID is TCGA-02-0007 and the data is supplied by the Harvard Medical School 244 Array CGH experiment (HMS). Segmentation results of iSeg (**a**) and other existing methods: DNAcopy (**b**), cghFLasso (**c**), and cghseg (**d**). The peaks pointed by the arrows and the region labeled by the red, and green squares are identified by iSeg, but not all of them are detected by the other three methods. Overall, iSeg consistently identifies all the significant peaks. Other methods often miss peaks or regions which are more significant than those identified

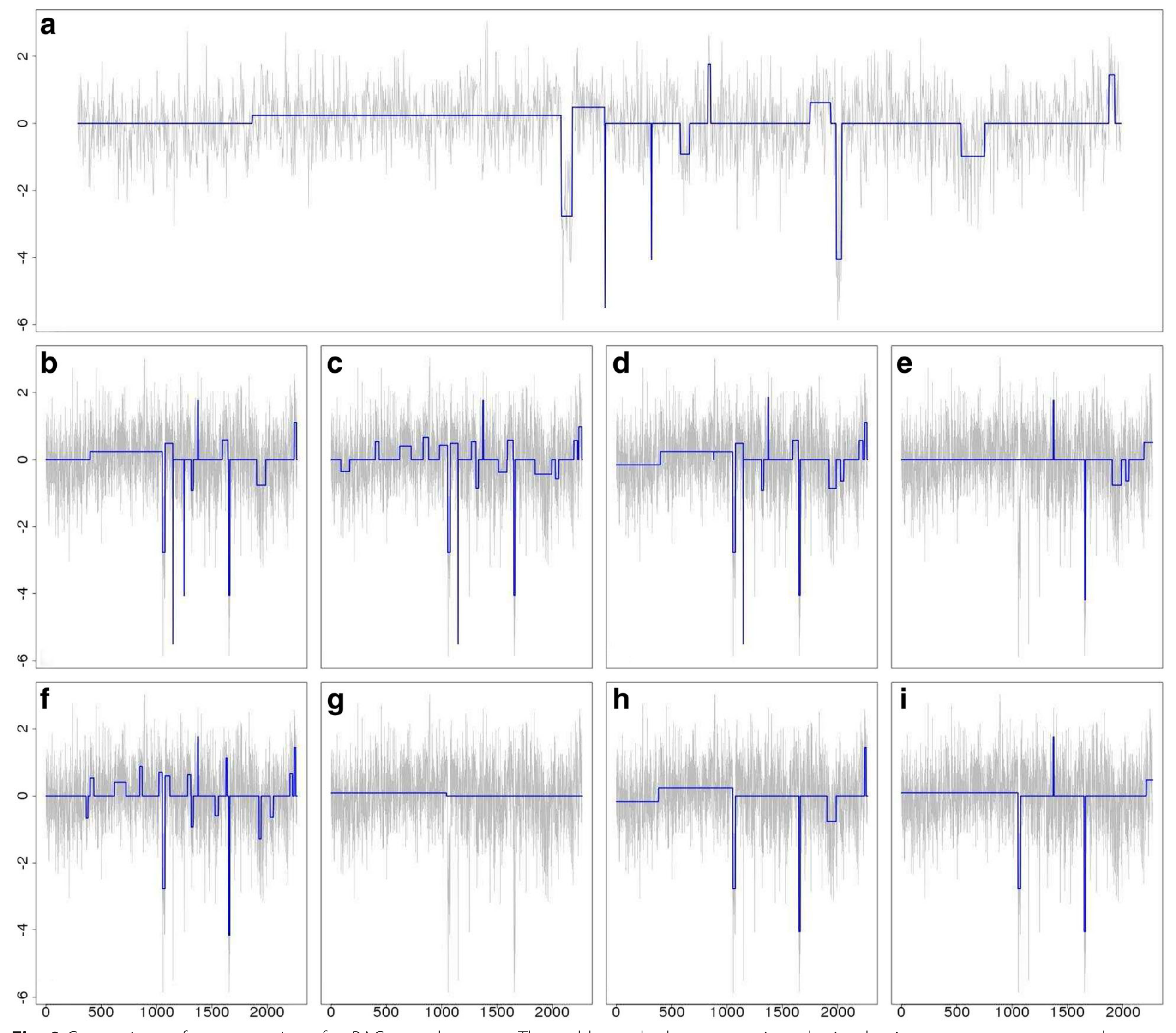


Fig. 9 Comparison of segmentations for BACarray dataset. **a** The gold standard segmentation obtained using a consensus approach. Segmentation results of iSeg (**b**) and other existing methods: snapCGH (**c**), mBPCR (**d**), cghseg (**e**), cghFLasso (**f**), HMMSeg (**g**), DNAcopy (**h**) and fastseg (**i**)

(NGS) are often assumed to follow either a Poisson or a negative binomial distribution [53, 54]. However, a recent study by us (to be published) showed that normality-based tests such as *t*-tests or Welch's *t*-test can perform equally well for NGS data after certain normalization, indicating Normal approximation of NGS data is likely adequate when analyzing NGS data, at least for some applications. For segmentation problems, when the segments tend to have relatively large sizes, the averages of signals within segments can be well approximated by normal distributions. This likely explains why iSeg performed well on NGS data. However, p-values of the segments can be computed based on either Poisson or negative binomial distributions, which can be directly used in the rest of the segmentation procedures. Such flexibility and general applicability make iSeg especially useful for segmenting a variety of genomic and epigenomic data.

The use of dynamic programming and a power factor makes iSeg computationally more efficient when analyzing very large data sets, especially with sparse signals as is typical for many types of NGS data sets. The refinement step identifies the exact boundaries of segments found by the scanning step. Merging allows iSeg to detect segments of any length. Together, these steps make iSeg an accurate and efficient method for segmentation of sequential data.

The statistic used in our method is very similar to that described for optimal sparse segment identification [7]. However in that study, the segments are identified using an exhaustive approach, which will not be efficient for segmenting very large profiles. To speed up computation, the

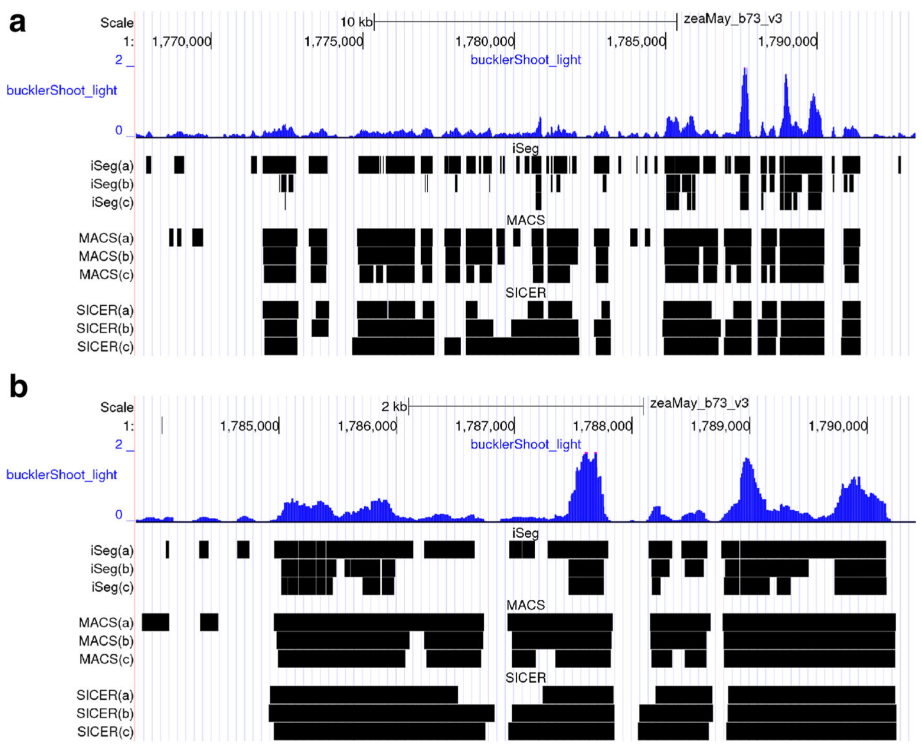


Fig. 10 Peak calling performance on Maize Epigenomics B73v3 Buckler Shoot DNS-seq data [55]. Top browser window (**a**) shows the biological range Chr1: 1,770,000–1,790,000. Bottom window (**b**) is a zoom in of range Chr1: 1,785,000–1,790,000. In each window, from top to bottom: first signal track is light digestion group (small fragment data only, color = blue). Segment tracks are enriched regions called by: iSeg(a): iSeg calls with bc = 1.0; iSeg(b): iSeg calls with bc = 2.0; iSeg(c): iSeg calls with bc = 3.0; MACS(a): MACS calls on the light digestion group, FDR = 0.05; MACS(b): MACS calls on the light digestion group, FDR = 0.01; MACS(c): MACS calls on the light digestion group, FDR = 0.001; SICER(a): SICER calls on the light digestion group with window size = 20 and gap size = 40; SICER(b): SICER calls on the light digestion group with window size = 30 and gap size = 60; SICER(c): SICER calls on the light digestion group with window size = 50 and gap size = 100

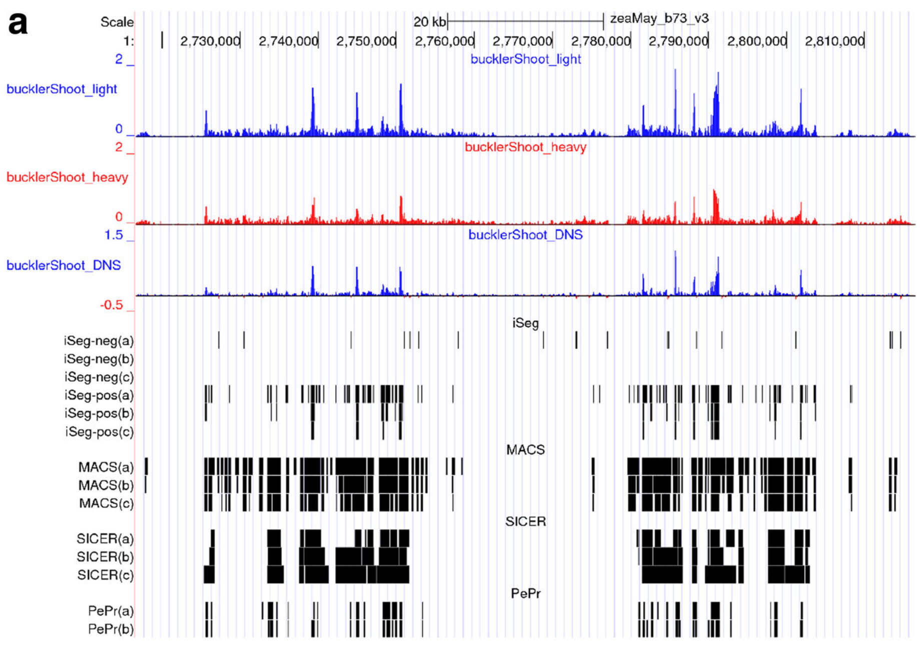
optimal sparse segment method [7] employs the assumption that the segments have relatively short length, which is not true for some datasets. In contrast, the algorithm designed in this study allowed us to detect segments of any length with demonstrably greater efficiency.

The gold standard approach is desirable but the derivation of a true gold standard is complex and possibly subjective. A gold standard generated using a consensus approach does not guarantee that the true optimal segments will be identified. In addition, the F_1-scores may favor iSeg more as the test statistic used to generate the gold standard is not employed by the other methods. However, the statistic we used is based on model assumptions used by many existing methods. Visual inspection of segmentation results clearly shows how iSeg performs well in direct comparisons with other popular, contemporary methods. We expect that future research will benefit from the application of iSeg to compare multiple profiles simultaneously.

We have designed the method to make it flexible and versatile. This resulted in a number of parameters that users can tune. However, the default values work well for all the simulated and experimental datasets. In practice, to obtain satisfactory results, users are not expected to modify any parameters. The speed of iSeg would allow us and fellow researchers to implement it as an online tool to deliver segmentation results in real-time.

Conclusion

In this work, we developed a new method, iSeg, for the segmentation of large-scale genomic and epigenomic profiles. The performance of iSeg was demonstrated by comparing with existing segmentation methods using both simulated and real datasets. The computation framework utilized in iSeg can readily accommodate different assumptions on the underlying probability distribution of the data. The computational algorithms designed in iSeg make it able to search for, grow, and refine segments comprehensively for very long profiles without compromising too much of the computational speed. iSeg also provides a biological cutoff parameter, allowing researchers to select segments which are more likely to be biological significant by setting a proper biological cutoff value based on the problems at hand and/or their prior knowledge. We believe that iSeg will be a very useful tool for segmentation problems in biology.



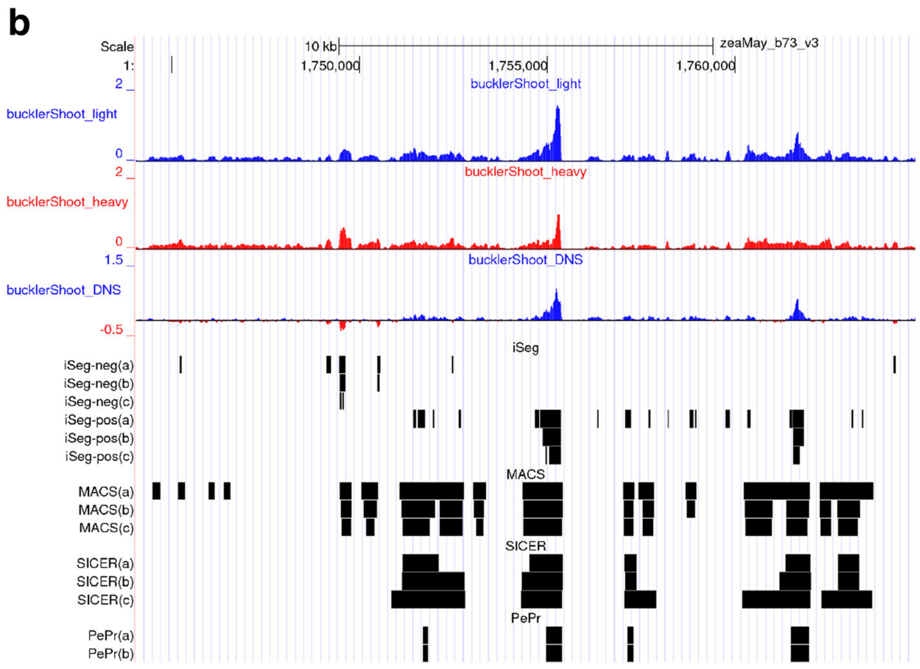


Fig. 11 Peak calling performance on Maize Epigenomics B73V3 nuprime Buckler Shoot DNS-seq data. Top browser window (**a**) shows the biological range Chr1: 2,730,000–2,810,000. Bottom window (**b**) is another range Chr1: 1,745,000–1,760,000 on a larger scale. In each window, from top to bottom: first signal track is light digestion group (treatment, color = blue); second signal track is heavy digestion group (control, color = red); Third signal track is difference track (light minus heavy, color code: positive = blue, negative = red). Segment tracks are enriched regions called by: iSeg-neg(a): iSeg calls of negative regions on the difference track with bc = 1.0; iSeg-neg(b): iSeg calls of negative regions on the difference track with bc = 2.0; iSeg-neg(c): iSeg calls of negative regions on the difference track with bc = 3.0; iSeg-pos(a): iSeg calls of positive regions on the difference track with bc = 1.0; iSeg-pos(b): iSeg calls of positive regions on the difference track with bc = 2.0; iSeg-pos(c): iSeg calls of positive regions on the difference track with bc = 3.0; MACS(a): MACS calls with light as treatment and heavy as control, FDR = 0.05; MACS(b): MACS calls with light as treatment and heavy as control, FDR = 0.01; MACS(c): MACS calls with light as treatment and heavy as control, FDR = 0.001; SICER(a): SICER calls with light as treatment and heavy as control, window size = 20 and gap size = 40; SICER(b): SICER calls with light as treatment and heavy as control, window size = 30 and gap size = 60; SICER(c): SICER calls with light as treatment and heavy as control, window size = 50 and gap size = 100; PePr(a): PePr calls with light as treatment and heavy as control, broad peak mode; PePr(b): PePr calls with light as treatment and heavy as control, sharp peak mode

Additional file

Additional file 1: The zipped supplementary file submitted with this manuscript contains Coriell (Snijders et al.) profiles, BACarray profiles, and simulated data files. (ZIP 1970kb)

Abbreviations

aCGH: (micro)array-based comparative genomic hybridization; BBT: Balanced binary tree; B-H: Benjamini-Hochberg; CNV: Copy-number variation; DBN: Dynamic Bayesian Network; FDR: False discovery rate; FPM: Fragment per million; FWER: Family-wise error rate; MACS: Model-based Analysis for ChIP-Seq; NGS: Next generation sequencing; SNR: Signal-to-noise ratio; TCGA: The Cancer Genome Atlas

Acknowledgements

We would like to thank the researchers who have generated the data used in this study.

Funding

This work was supported by a grant from NSF (Plant Genome Research Program, IOS Award 1444532). JZ was also supported by the National Institute of General Medical Sciences of the National Institute of Health under award number R01GM126558.

Availability of data and materials

The data used in this study are available publicly or in the Additional file 1. The executable file of the iSeg program can be downloaded at http://ani.stat.fsu.edu/~jinfeng/programs/iseg. A server has been built and will be released soon.

Authors' contributions

JZ designed the core iSeg algorithm. JZ, HWB, DLV, and JHD were responsible for the study design and result interpretation. SBG, YL, and PYL performed data analyses. SBG wrote the first draft of the manuscript. All authors revised the manuscript and approved the final version.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Author details

[1]Department of Mathematics, Florida Gulf Coast University, Fort Myers, FL, USA. [2]Department of Statistics, Florida State University, Tallahassee, FL, USA. [3]Center for Genomics and Personalized Medicine, Florida State University, Tallahassee, FL, USA. [4]Department of Biological Science, Florida State University, Tallahassee, FL, USA.

Received: 5 September 2017 Accepted: 26 March 2018
Published online: 11 April 2018

References

1. Consortium EP. An integrated encyclopedia of DNA elements in the human genome. Nature. 2012;489(7414):57–74.
2. Baldi P, Brunak S. Bioinformatics: the machine learning approach. Cambridge: MIT press; 2001.
3. Baldi P, Long AD. A Bayesian framework for the analysis of microarray expression data: regularized t-test and statistical inferences of gene changes. Bioinformatics. 2001;17(6):509–19.
4. David L, et al. A high-resolution map of transcription in the yeast genome. Proc Natl Acad Sci. 2006;103(14):5320–5.
5. Day N, et al. Unsupervised segmentation of continuous genomic data. Bioinformatics. 2007;23(11):1424–6.
6. Diskin SJ, et al. STAC: a method for testing the significance of DNA copy number aberrations across multiple array-CGH experiments. Genome Res. 2006;16(9):1149–58.
7. Jeng XJ, Cai TT, Li H. Optimal sparse segment identification with application in copy number variation analysis. J Am Stat Assoc. 2010;105(491):1156–66.
8. Kampa D, et al. Novel RNAs identified from an in-depth analysis of the transcriptome of human chromosomes 21 and 22. Genome Res. 2004; 14(3):331–42.
9. Olshen AB, et al. Circular binary segmentation for the analysis of array-based DNA copy number data. Biostatistics. 2004;5(4):557–72.
10. Picard F, et al. Joint segmentation of multivariate Gaussian processes using mixed linear models. Comput Stat Data Anal. 2011;55(2):1160–70.
11. Picard F, et al. A segmentation/clustering model for the analysis of array CGH data. Biometrics. 2007;63(3):758–66.
12. Rancoita PM, et al. Bayesian DNA copy number analysis. BMC Bioinform. 2009;10(1):10.
13. Tony Cai T, Jessie Jeng X, Li H. Robust detection and identification of sparse segments in ultrahigh dimensional data analysis. J R Stat Soc. 2012;74(5):773–97.
14. Venkatraman E, Olshen AB. A faster circular binary segmentation algorithm for the analysis of array CGH data. Bioinformatics. 2007;23(6):657–63.
15. Wang K, et al. PennCNV: an integrated hidden Markov model designed for high-resolution copy number variation detection in whole-genome SNP genotyping data. Genome Res. 2007;17(11):1665–74.
16. Wilbanks EG, Facciotti MT. Evaluation of algorithm performance in ChIP-seq peak detection. PLoS One. 2010;5(7):e11471.
17. Zhang NR, Siegmund DO. A modified Bayes information criterion with applications to the analysis of comparative genomic hybridization data. Biometrics. 2007;63(1):22–32.
18. Chen J, Wang Y-P. A statistical change point model approach for the detection of DNA copy number variations in array CGH data. IEEE/ACM Trans Comput Biol Bioinform. 2009;6(4):529–41.
19. Chen J, Yiğiter A, Chang K-C. A Bayesian approach to inference about a change point model with application to DNA copy number experimental data. J Appl Stat. 2011;38(9):1899–913.
20. Cleynen A, et al. Segmentor3IsBack: an R package for the fast and exact segmentation of Seq-data. Algorithms Mol Biol. 2014;9(1):6.
21. Killick R, Eckley I. Changepoint: an R package for changepoint analysis. J Stat Softw. 2014;58(3):1–19.
22. Niu YS, Zhang H. The screening and ranking algorithm to detect DNA copy number variations. Ann Appl Stat. 2012;6(3):1306.
23. Picard F, et al. A statistical approach for array CGH data analysis. BMC Bioinform. 2005;6(1):27.
24. Sen A, Srivastava MS. On tests for detecting change in mean. Ann Stat. 1975;3(1):98–108.
25. Yao Q. Tests for change-points with epidemic alternatives. Biometrika. 1993: 179–91.
26. Jaschek R, Tanay A. Spatial Clustering of Multivariate Genomic and Epigenomic Information. In: Batzoglou S. (eds) Research in Computational Molecular Biology. RECOMB 2009. Lecture Notes in Computer Science, vol 5541. Berlin, Heidelberg: Springer; 2009.
27. Marioni J, Thorne NP, Tavaré S. BioHMM: a heterogeneous hidden Markov model for segmenting array CGH data. Bioinformatics. 2006; 22(9):1144–6.
28. Stjernqvist S, et al. Continuous-index hidden Markov modelling of array CGH copy number data. Bioinformatics. 2007;23(8):1006–14.
29. Hoffman MM, et al. Unsupervised pattern discovery in human chromatin structure through genomic segmentation. Nat Methods. 2012;9(5):473–6.
30. Hoffman MM, et al. Integrative annotation of chromatin elements from ENCODE data. Nucleic Acids Res. 2012:gks1284.
31. Ben-Yaacov E, Eldar YC. A fast and flexible method for the segmentation of aCGH data. Bioinformatics. 2008;24(16):i139–45.
32. Hu J, et al. Exploiting noise in array CGH data to improve detection of DNA copy number change. Nucleic Acids Res. 2007;35(5):e35.
33. Hupé P, et al. Analysis of array CGH data: from signal ratio to gain and loss of DNA regions. Bioinformatics. 2004;20(18):3413–22.
34. Tibshirani R, Wang P. Spatial smoothing and hot spot detection for CGH data using the fused lasso. Biostatistics. 2008;9(1):18–29.
35. Morganella S, et al. VEGA: Variational segmentation for copy number detection. Bioinformatics. 2010;26(24):3020–7.

36. Nilsson B, et al. Ultrasome: efficient aberration caller for copy number studies of ultra-high resolution. Bioinformatics. 2009;25(8):1078–9.
37. Kharchenko PV, Tolstorukov MY, Park PJ. Design and analysis of ChIP-seq experiments for DNA-binding proteins. Nat Biotechnol. 2008;26(12):1351–9.
38. Lai WR, et al. Comparative analysis of algorithms for identifying amplifications and deletions in array CGH data. Bioinformatics. 2005;21(19):3763–70.
39. Willenbrock H, Fridlyand J. A comparison study: applying segmentation to array CGH data for downstream analyses. Bioinformatics. 2005;21(22):4084–91.
40. Park PJ. Experimental design and data analysis for array comparative genomic hybridization. Cancer Investig. 2008;26(9):923–8.
41. Brodsky E, Darkhovsky BS. Nonparametric methods in change point problems. Vol. 243: Springer Science & Business Media; 2013. https://play.google.com/store/books/details?id=GLvwCAAAQBAJ.
42. Roy S, Motsinger-Reif A. Evaluation of calling algorithms for array-CGH. Front Genet. 2013;4:217.
43. Benjamini Y, Hochberg Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. J R Stat Soc Ser B Methodol. 1995:289–300.
44. Schnable PS, et al. The B73 maize genome: complexity, diversity, and dynamics. Science. 2009;326(5956):1112–5.
45. Langmead B, et al. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol. 2009;10(3):R25.
46. Quinlan AR, Hall IM. BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics. 2010;26(6):841–2.
47. Xu S, et al. Spatial clustering for identification of ChIP-enriched regions (SICER) to map regions of histone methylation patterns in embryonic stem cells, Stem cell transcriptional networks: methods and protocols; 2014. p. 97–111.
48. Zhang Y, et al. PePr: a peak-calling prioritization pipeline to identify consistent or differential peaks from replicated ChIP-Seq data. Bioinformatics. 2014;30(18):2568–75.
49. Zhang Y, et al. Model-based analysis of ChIP-Seq (MACS). Genome Biol. 2008;9(9):R137.
50. Snijders AM, et al. Assembly of microarrays for genome-wide measurement of DNA copy number. Nat Genet. 2001;29(3):263–4.
51. Vera DL, et al. Differential nuclease sensitivity profiling of chromatin reveals biochemical footprints coupled to gene expression and functional DNA elements in maize. Plant Cell. 2014;26(10):3883–93.
52. Sexton BS, et al. The spring-loaded genome: nucleosome redistributions are widespread, transient, and DNA-directed. Genome Res. 2014;24(2):251–9.
53. Auer PL, Doerge R. Statistical design and analysis of RNA sequencing data. Genetics. 2010;185(2):405–16.
54. Marioni JC, et al. RNA-seq: an assessment of technical reproducibility and comparison with gene expression arrays. Genome Res. 2008;18(9):1509–17.
55. Rodgers-Melnick E, et al. Open chromatin reveals the functional maize genome. Proc Natl Acad Sci. 2016;113(22):E3177–84.