Finding Optimal Sets of Enriched Regions in ChIP-Seq Data

Andreas Gogol-Döring* and Wei Chen
Berlin Institute for Medical Systems Biology,
Max Delbrück Center for Molecular Medicine, Berlin, Germany
andreas.doering@mdc-berlin.de

Abstract: The main challenge when analyzing ChIP-Seq data is the identification of DNA-protein binding sites by finding genomic regions that are enriched with sequencing reads. We present a new tool called qips especially suited for processing ChIP-Seq data containing broader enriched regions. Our tool certainly finds all enriched regions that are not exceeded by higher significant alternatives.

1 Introduction

Chromatin immunoprecipitation followed by high-throughput DNA sequencing (ChIP-Seq) [JMMW07] is a common method for genome-wide profiling protein-DNA interactions. In ChIP-Seq antibodies specifically select the proteins of interest together with any piece of randomly fragmented DNA bound to them, and the origins of the selected DNA fragments are then determined by sequencing and mapping to a reference genome. Protein binded regions feature an increased number of mapped sequencing reads. Several software packages were recently published for finding enriched regions in ChIP-Seq sequencing data; a good survey can be found in [PWM09]. Most tools, for example SISSRs [JCB+08], F-Seq [BGCF08], or QuEST [V+08], concentrate on finding short peaks indicating nearly punctate protein bindings as it would be typical for transcription factors, whereas nucleosomes or polymerases bind to much broader regions. Some other tools like MACS [Z+08] and CisGenome [JJM+08] can also find longer enriched regions by merging overlapping short regions found in fixed-length sliding windows, but long regions are likely to be cut when a short sliding window is applied, and increasing the windows size would make it impossible to estimate the region boundaries precisely. SICER [ZSZ+09] tries to overcome this problem by partitioning the genome into non-overlapping windows and searching for sequences of succeeding enriched windows which may be interrupt by a limited number of non-enriched windows. However, the precision of this approach is limited the granularity of the applied window grid.

In this paper, we present a new algorithm that identifies enriched regions of arbitrary length and boundaries in ChIP-Seq data. Our method finds an optimal set of enriched regions, which means that it reports an enriched region if there is no better, i.e. more significant, alternative overlapping region. Note that prior approaches for analyzing ChIP-
Seq data do not guarantee to find optimal region sets.

Our tool \textit{qips} (quantification of IP-Seq) also features a new way for estimating the average DNA fragment length from single-end sequencing data; this is discussed in Section 2.1. In Section 2.2 we describe how \textit{qips} estimates the statistical background from mappability information or, if available, a control data set. A quick approximation formula for scoring candidate regions is presented in Section 2.3. The algorithm for finding enriched regions is described in Section 2.4. We discuss our results in Section 3.1.

2 Methods

The analysis of ChIP-Seq data starts with the mapping of the sequenced reads to a reference genome \( G \) using a tool like Bowtie [LTPS09] or RazerS [WER+09]. The position of a read in \( G \) is the center of the subsequence of \( G \) to which the read matches. A read is only used for the following analysis if there is a single ‘best’ match of it in \( G \), because otherwise we cannot infer its true origin. However, this also means that it is hard to detect protein bindings in repetitive genomic regions. This is a general limitation of the ChIP-Seq technology, and our statistical model explicitly take it into account; see Section 2.2.

For avoiding biases due to PCR artifacts, we retained only one read at the same position and the same strand orientation. To identify the centers of the ChIP-Seq fragments, which are usually much longer than the sequenced reads, we shift each read downstream by \( s = (f - q) / 2 \), where \( f \) is the average fragment length and \( q \) the read length. Our method for accurately estimating \( f \) from single-end reads is described in Section 2.1.

In the final set of uniquely mapped, non-redundant, and shifted reads we then search for \textit{enriched regions}, namely for intervals \([a, b] \subset G\) containing significantly more read positions than expected by chance, which means that the p-value \( p[a, b] \) relative to a given background model is below a certain user-defined threshold \( \alpha \). Our background model assumes an uniform distribution of the reads over all (mappable) positions \( i \in G \) or, if available, it accounts for a second control data set obtained, e.g., from a ChIP-Seq using Immunoglobulin G (IgG); see Section 2.2. As \textit{qips} was designed to find enriched regions of arbitrary size, we must also decide whether two neighboring enriched regions \([a, b]\) and \([a', b']\) could in fact be a single enriched region \([a, b']\). We prefer \([a, b']\) instead of \([a, b]\) and \([a', b]\), if and only if the combined region is more significant than the two sub-regions, i.e. if \( p[a, b'] < \min\{p[a, b], p[a', b]\} \leq \alpha \). \textit{qips} computes an \textit{optimal set of enriched regions}, which is defined as follows:

**Definition:** Let \( I = \{i_1, i_2, \ldots, i_n\} \) be a set of intervals in a genome \( G \). Two intervals \([a, b]\) and \([a', b']\) overlap if \( a \leq b' \) and \( a' \leq b \). A subset \( R \subseteq I \) is a \textit{set of enriched regions} if the intervals in \( R \) are pairwise disjoint and if \( p(r) \leq \alpha \) for all \( r \in R \). \( R \) is \textit{optimal} if for each interval \( i \in I \setminus R \) exists an interval \( r \in R \) overlapping with \( i \) and \( p(r) \leq p(i) \).

\( R \) is unique if \( p(i) \neq p(i') \) for any two overlapping intervals \( i \neq i' \in I \), otherwise there are several optimal sets. We assume that the differences between those alternative sets have only minor practical relevance and can therefore be ignored. Our tool finds an optimal set of enriched regions for \( n \) reads in time \( O(n^2) \) and linear space; see Section 2.4.
2.1 Estimating the Fragment Length

ChIP-Seq DNA fragments are usually sequenced only from a single end, so it is not possible to deduce their lengths directly from the data. On the other hand, measuring the fragment length using laboratory equipment does not account for a length bias introduced in the sequencing procedure. Several authors therefore described methods for estimating average fragment lengths from single-end data, either using shift distances between peaks with different strand orientation [Z+C08] [V+C08] [JMM+C08], or the distances between forward reads and their closest reverse read [JCB+C08]. All these approaches assume (nearly) punctate peaks in the data, so they are less appropriate for ChIP-Seq of proteins binding to broader regions. Some methods are also susceptible for noise in the data or could be affected by a locality bias, which is the preference for reads being mapped to genomic positions where they overlap to other reads; see Figure 1A. We found locality biases of varying intensities in numerous public available data sets from different labs, and since it is partly caused by the limited mappability of short reads to large genomes, it can hardly be avoided completely.

Our method for estimating the average fragment length \( f \) relies on the shift between read distributions of different strand orientation: For each forward read at position \( i \) we compute the frequency \( F_i(d) \) of forward reads and the frequency \( R_i(d) \) of reverse reads at position \( i + d \). We define the total read frequencies \( F(d) = \sum_i F_i(d)/k \) and \( R(d) = \sum_i R_i(d)/k \), where the normalization factor \( k \) is the total number of forward strand reads. Figure 1A illustrates that \( R \) typically resembles \( F \) shifted downstream. The mode of \( R \) added to the read length \( q \) would be a simple estimate for the fragment length \( f \). Here we use a different approach that is less prone to noise and yields more accurate results for skewed fragment length distributions. We computed for shift widths \( d > 0 \) the average \( A(d) \) of the squared difference \( (F(j) - R(j + d))^2 \) over all \( j \notin [-q, q] \cup [d - q, d + q] \), i.e. we exclude all \( j \) where \( F(j) \) or \( R(j) \) could be affected by a locality bias. Then the average fragment length is estimated by \( f = q + \arg\min_d A(d) \).

We tested our method on various data sets, and it yielded reasonable results even for data containing very broad or unspecific binding, like ChIP-Seq targeting H3K36me3 histone marks or using IgG. For proving the accuracy of our method, we also sampled single-end reads from a paired-end ChIP-Seq data set [WXZ+C10] and compared the estimates from different tools to the actual fragment lengths; see Figure 1B.

2.2 Modeling the Statistical Background

A \( q \)-mer is unique if it occurs only once in a genome \( G \), and its position in \( G \) is called a (uniquely) mappable position. Repetitive regions of \( G \) are characterized by a lower density of mappable positions and therefore contain less reads than regions with higher mappability. However, only few software tools for ChIP-Seq data analysis take mappability variations in genomes into account [REA+C09]. Let \( mp[a, b] \) be the number of mappable positions in \([a, b] \subseteq G\), then the maximal number of reads in \([a, b]\) after shifting the reads...
downstream by \( s \) is given by:

\[
\text{maplen}[a, b] = mp[a - s, b - s] + mp[a + s, b + s].
\]

Our background model assumes that reads in the data set \( S \) are spread independently over the genome \( G \) by a Poisson process, which means that, given an interval \([a, b] \subset G\), the reads may occur at any position \( i \in [a, b] \) with the same rate \( \mu = \lambda / \text{maplen}[a, b] \), where \( \lambda = \mathbb{E}(\text{count}_S[a, b]) \) is the expected number of reads in \([a, b]\). These assumptions may be questionable, especially the independence between the reads; nevertheless, this kind of model is very common because there is a lack of better alternatives.

We apply two ways for estimating the \( \lambda \) in a given interval \([a, b]\):

1. The expected number \( \lambda_M \) of reads in \([a, b]\) assuming an uniform distribution of all \( \text{count}_S(G) \) reads in \( S \) to the \( \text{maplen}(G) \) mappable positions in \( G \) is:

\[
\lambda_M = \frac{\text{maplen}[a, b]}{\text{maplen}(G)} \text{count}(G).
\]

In order to get a more local estimation of \( \lambda_M \), one could also use a region \( G_{\text{part}} \subset G \) containing \([a, b]\) instead of the whole genome \( G \).

2. A second estimation \( \lambda_C \) is done if a control data set \( C \) is available. We calculate the density of reads in \( C \) by:

\[
\mu_C = \frac{\text{count}_C[a - s, b + s]}{\text{maplen}[a - s, b + s]},
\]

i.e. we enlarge \([a, b]\) in both sides by the shift width \( s \) to avoid clipping effects due to variations in the fragment lengths. The read density \( \mu_S \) in \( S \) can be estimated
given $\mu_C$ after normalizing for the different read quantities in both data sets, so we get:

$$\lambda_C = \mu_S \text{maplen}[a, b] = \mu_C \frac{\text{count}_S(G)}{\text{count}_C(G)} \text{maplen}[a, b].$$

If both estimates are available, then we use the maximum $\lambda = \max(\lambda_M, \lambda_C)$.

### 2.3 Computing p-Values

Let $T = \{t_1, t_2, \ldots, t_n\}$ be a set of different read positions, $t_i < t_j$ for $i < j$. Regarding all intervals containing exactly the reads at the positions $t_i, \ldots, t_j$ is $[t_i, t_j]$, obviously $[t_i, t_j]$ is the interval with maximum read density, so we can restrict the search for enriched regions on intervals starting and ending at read positions. This is a great saving of time, because the typical number of reads in a ChIP-Seq data set is two to three orders of magnitude smaller than the genome length.

Starting with a fixed read position $t_i$, the probability for finding the next $k = \text{count}[t_i + 1, t_j]$ reads within the interval $[t_i + 1, t_j]$ is given by an Erlang distribution:

$$f(x; k, \mu) = \frac{\mu^k x^{k-1} e^{-\mu x}}{(k-1)!},$$

where $x = \text{maplen}[t_i + 1, t_j]$. The p-value is defined by the cumulative density function:

$$p[t_i, t_j] = \sum_{x \leq \text{maplen}[t_i+1,t_j]} f(x; k, \mu) = \frac{\gamma(k, \lambda)}{(k-1)!},$$

where $\gamma$ is the lower incomplete gamma function. Note that $p[t_i, t_j]$ only depends on the actual number $k$ and expected number $\lambda$ of reads in $[t_i + 1, t_j]$.

In practice, it is often more convenient to deal with logarithmic scores than with p-values, so we further define $\text{score}[t_i, t_j] = -\log(p[t_i, t_j])$. Since Algorithm 1 has to calculate a huge amount of scores, we substituted the time consuming computation of the function $\gamma$ by the following approximation formula:

$$\text{score}[t_i, t_j] \approx \lambda - k \log(\lambda) + \log(k!) - 0.08 \log(k)^{1.6}$$

This way we speed up our program by more than 50 times compared to a direct computation of $\gamma$ using the GNU Scientific Library (GSL) [GDT+10]. For scores $\geq 10$, the approximations diverge by less than 5% from the exact values; see Figure 2A.

### 2.4 Finding Optimal Sets of Enriched Regions

Let $T = \{t_1, \ldots, t_n\}$ be a sorted set of interval boundaries. \textsc{FindOptimalSet} (see Algorithm 1) calculates an optimal set $R$ of enriched regions in two steps: First, it determines for each start position $t_i \in T$ the optimal end position $t_{E[i]}$, where $[t_i, t_{E[i]}]$ must
Figure 2: A: Comparison between exact and approximated scores. B: Fraction of detected regions depending on the required minimum overlap between actual and predicted regions for \textit{qips} (this paper), MACS, and SICER with different parameter settings.

not overlap with any higher scoring region starting at \( t_k > t_i \). Second, the algorithm selects intervals \([t_i, t_{E[i]}]\) with increasing starting positions \( t_i \). Obviously, the resulting set \( R \) is a set of non-overlapping enriched regions. We show that \( R \) is optimal as follows: Let \( \mathcal{M} \) be the set of intervals with maximum score in \( I = \{[t_i, t_j] | t_i, t_j \in T \} \), let \( \mathcal{M}' \subset \mathcal{M} \) be the intervals in \( \mathcal{M} \) with maximum start position, and \([t_i, t_j] \in \mathcal{M}' \) the interval with minimum \( t_j \). The array \( E \) is constructed such that \( E[t_i] = j \) and \( E[t_k] < i \) for all \( k < i \), hence \( i \) is not skipped in line 18 of Algorithm 1, and therefore \([t_i, t_j] \in R \). The optimality of \( R \) follows by applying structural induction to the remaining sets of boundaries \( \{t_1, \ldots , t_{i-1}\} \) and \( \{t_{j+1}, \ldots , t_n\} \).

The algorithm can easily be modified such that it restricts the search to a subset of \( I \). For example, \textit{qips} allows to set the minimum and maximum length as well as the minimum number of reads in a candidate region. Moreover, it is possible to exclude any interval containing a \textit{drop}, which we define here as an interval \([t_i, t_j]\) having a certain minimum length and either contains less reads than expected by chance, or has a mappability \( \text{map}[t_i, t_j]/(t_j - t_i) \) below a minimum threshold. A drop cuts the search space into two parts, hence the run time of the algorithm gets linear after choosing appropriate drop parameters.
Algorithm 1: Finding an optimal set of enriched regions. $T$ is a sorted set of interval boundaries, i.e. read positions, and $\alpha$ the p-value threshold. $\text{score}[t_i, t_j] = -\log(p[t_i, t_j])$. 

\begin{verbatim}
\begin{algorithm}[H]
\caption{Finding an optimal set of enriched regions ($T = \{t_1 \ldots t_n\}$)}
$S[j] \leftarrow 0$ for all $j \in \{1, \ldots, n\}$

for $i \leftarrow n$ down to 1 do
  $s_{\text{min}} \leftarrow -\log(\alpha)$
  $s_{\text{opt}} \leftarrow 0$
  $j_{\text{opt}} \leftarrow \text{nil}$
  for $j \leftarrow i$ to $n$ do
    if $\text{score}[t_i, t_j] > \max(s_{\text{min}}, s_{\text{opt}})$ then
      $s_{\text{opt}} \leftarrow \text{score}[t_i, t_j]$
      $j_{\text{opt}} \leftarrow j$
      $s_{\text{min}} \leftarrow \max(s_{\text{min}}, S[j])$
  $S[i] \leftarrow s_{\text{opt}}$
  $E[i] \leftarrow j_{\text{opt}}$
  $R \leftarrow \{\}$
  $i \leftarrow 1$

while $i < n$ do
  if $E[i] \neq \text{nil}$ then
    $R \leftarrow R \cup \{[t_i, t_{E[i]}]\}$
    $i \leftarrow E[i] + 1$
  else
    $i \leftarrow i + 1$

return $R$
\end{algorithm}
\end{verbatim}
3 Results and Discussion

3.1 Results

We simulated threefold enriched regions, each of length $10\, kb$, on a $\mu = 1\%$ read density background. This data set was used to compare qips with SICER, which is a tool especially designed for searching long enriched regions, and with the popular peak finder MACS. The output of SICER depends very much on the input parameters, so we tried several settings. MACS was started with the `--nolambda` command line option for finding longer enriched regions. qips detected all enriched regions in the data set, whereas MACS totally missed about $12\%$ of them. MACS and SICER (for some settings) also splitted some of the enriched regions into smaller parts.

We measured the overlap between two regions by the number of common bases divided by the length of the longer region. An enriched region was counted among the detected regions, if its overlap to one region in the tool output file was above a certain threshold. Figure 2B shows the sensitivity of the three tools depending on this overlap threshold. It can be seen that qips detects enriched regions more precisely than the competitors.

3.2 Discussion

Our approach performs an exhaustive search of all possibly enriched regions and, consequently, should have better chances to detect enriched regions than a heuristical approach that limits the search space. The results presented above illustrate that, at least in some cases, qips indeed has some advantages compared to previously published tools like SICER or MACS. On the other hand, our algorithm takes quadratic run time and is therefore significantly more time-consuming than other tools. Applying a relaxed drop condition can significantly improve the run time, but this also increases the risk for missing high scoring enriched regions. A thorough test of our tool both for simulated and real ChIP-Seq data would help to find a good balance between the sensitivity and the performance of our software. This is future work.

3.3 Implementation

We implemented qips in C++ and Python, using the GNU Scientific Library [GDT+10] and SeqAn [DWRR08]. The program is controlled by a make file, so it can simply be parallelized by specifying the GNU make `-f` command line option, or distributed to a computer cluster using the Sun Grid Engine qmake tool.

Our software will be published free and open source.
References


