Supplementary Methods

Motifs

A set of overrepresented motifs is *de novo* identified from the set of three hundred "*significant peaks*". This number was chosen so that the number of peaks would be sufficient to identify all strong motifs and so that a run of MEME[17] would not take too much time. The set of "*significant peaks*" includes both the highest peaks and rather low ones since it is possible that lower peaks contain motifs different from those in the highest peaks. Starting from chromosome 1, we continue until we have 300 selected peaks: for each chromosome, we add to our set peaks with DNA fragment coverage greater than threshold *t*. Here *t* is selected for each chromosome so that in the control dataset there would be no peaks with DNA fragment coverage greater than *t*.

As we believe that real binding motifs occur in the central area of peaks, we select only central areas of 300 peaks to identify motifs.

We run MEME on the set *S* with the following parameters:

-revcomp (search on both strands)

-dna (use DNA 4 letters alphabet)

-mod zoops (expect to find motif zero or one times in a sequence)

-evt 0.5 (maximal E-value for a motif is 0.5)

-minw 7 (minimal motif length is 7)

-maxw 24 (maximal motif length is 24)

We run MEME three times to identify up to three motifs for dataset *D*, each time we keep the best identified motif.

The first run of MEME results in set $\{H_i\}$ of sequences that constitute the most overrepresented motif M_1 in *D*. Using the set $\{H_i\}$ we construct a position-specific scoring matrix (PSSM): $PSSM[i][\alpha] = ln\left(\frac{counts[i][\alpha] + pseudoScore * p_\alpha}{N + pseudoScore} / p_\alpha\right)$, where $counts[i][\alpha]$ is the number of sites where the nucleotide α is observed in position *i*, p_α is a background probability of nucleotide α, *N* is a number of sites, $pseudoScore = \ln(N)$ is a pseudo-score.

The set *D* of 300 *significant peaks* is then searched again for occurrences of M_1 with its PSSM and minimal score threshold *minT*. The minimal threshold *minT* for the PSSM score is

selected as $minT = minThr - \frac{4 \cdot minThr}{maxThr - minThr}, minThr = \min_i PSSM(H_i),\ maxThr = \max_i PSSM(H_i)$. Here $PSSM(H_i)$ is a score of word H_i calculated with the PSSM of motif M_1. The correction $\frac{4 \cdot minThr}{maxThr - minThr}$ is introduced to weaken strong motifs.

Sequences from D which do not contain such occurrences form the second set D_2. They are subject to the second run MEME. If MEME finds the second overrepresented motif M_2, the set D_3 of sequences which do not contain motif M_2 will be constructed. Then, MEME is run for the third time to identify the third motif if this motif exists.

Classes

Below, we consider separately each chromosome. The whole set of peaks for each chromosome is divided into classes so that each class C_i contains peaks with the same number of overlapping DNA fragments i. For example, in Supplementary Figure 5, nine DNA fragments overlap, forming a peak which belongs to class C_9. We believe that all peaks belonging to the same class have the same properties; below, we use the same statistical parameters for all peaks from the same class.

Initial FDR

Since we have two datasets: one from ChIP and the other from the control experiment, we can calculate the *initial false discovery rate* (FDR) for each class C_i, which is a ratio:

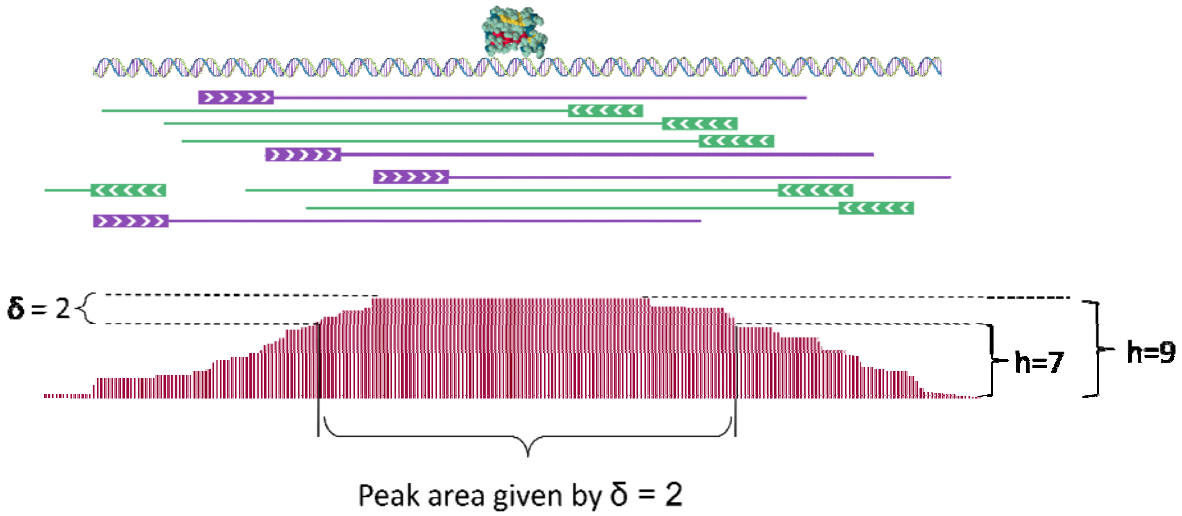
$inFDR_i = \frac{K_i}{M_i}$, where K_i is the number of peaks with depth of DNA fragment coverage equal to or greater than i in the control dataset, and M_i is the number of peaks with the depth of DNA fragment coverage equal to or greater than i in the ChIP dataset.

The FDR value represents the probability that a peak in $\{C_j\}_{j \geq i}$ is a result of random events and is not a real binding site.

Optimization.

The last step of the MICSA pipeline is the optimization procedure. It aims to maximize the total number of reported peaks for each class so that the total number of expected false positives does not exceed the user-defined threshold (F). Below, we focus the explanation on the case of one motif M and one chromosome.

For each class of peaks C_i, we optimize the following parameters: $T_{M,i}$ which is a PSSM score threshold for a given motif M, and $\delta_{M,i}$ which is a parameter specifying the length of the central area of the peak (Supplementary Figure 5).



Supplementary Figure 5. Example of area provided by δ=2. Given this δ, the peak will be filtered if it does not contain a motif occurrence in the central area provided by δ. Here h is a number of overlapping DNA fragments.

For each class C_i and each pair ($T_{M,i}$, $\delta_{M,i}$) we can calculate $S_i(T_{M,i}, \delta_{M,i})$ which is the number of peaks that we select when we only keep peaks with at least one occurrence of motif M (PSSM score ≥ $T_{M,i}$) in the central area specified by $\delta_{M,i}$. Independently, we can estimate the number of false peaks selected by chance using the procedure:

$$F_i(T_{M,i}, \delta_{M,i}) = \sum_{peak\ j\ in\ C_i} (inFDR_i)(mp_j),$$

where $inFDR_i$ is the *initial FDR* of class C_i defined above and mp_j is a motif p-value of peak j described in the following.

Each term in the sum is the probability of two events considered to be independent: that a peak has DNA fragment coverage equal to or greater than i just by chance and that by chance this peak contains a motif occurrence.

The *motif p-value* for peak j (mp_j) is a probability to observe a motif by chance in a sequence of given length. In our case the motif is represented by its PSSM with the threshold $T_{M,i}$, and the sequence length $L(\delta_{M,i}, j)$ is equal to the length of central area of peak j provided by $\delta_{M,i}$. We use the Poisson approximation to calculate the motif p-value: $motif\ p\text{-}value \approx 1 -$

$(1 - P(M))^{L(\delta_{M,i},j)-MotifLength+1}$. Here, the *motif probability* $P(M)$ is the probability of observing a motif occurrence with a PSSM score above a given threshold $T_{M,i}$ at a given position. We consider it equal to the genomic frequency of the motif with threshold $T_{M,i}$. Since it is very time-consuming to evaluate motif frequencies in a whole genome, in our approach we use only chromosome 1.

The aim of the optimization is to find such values of ($T_{M,i}$, $\delta_{M,i}$) that maximize $\Sigma S_i(T_{M,i}$, $\delta_{M,i})$ so that $\Sigma F_i(T_{M,i}$, $\delta_{M,i})$ stays below the user-defined threshold F. Since it is very time consuming to do it exactly, we discretize all parameters.

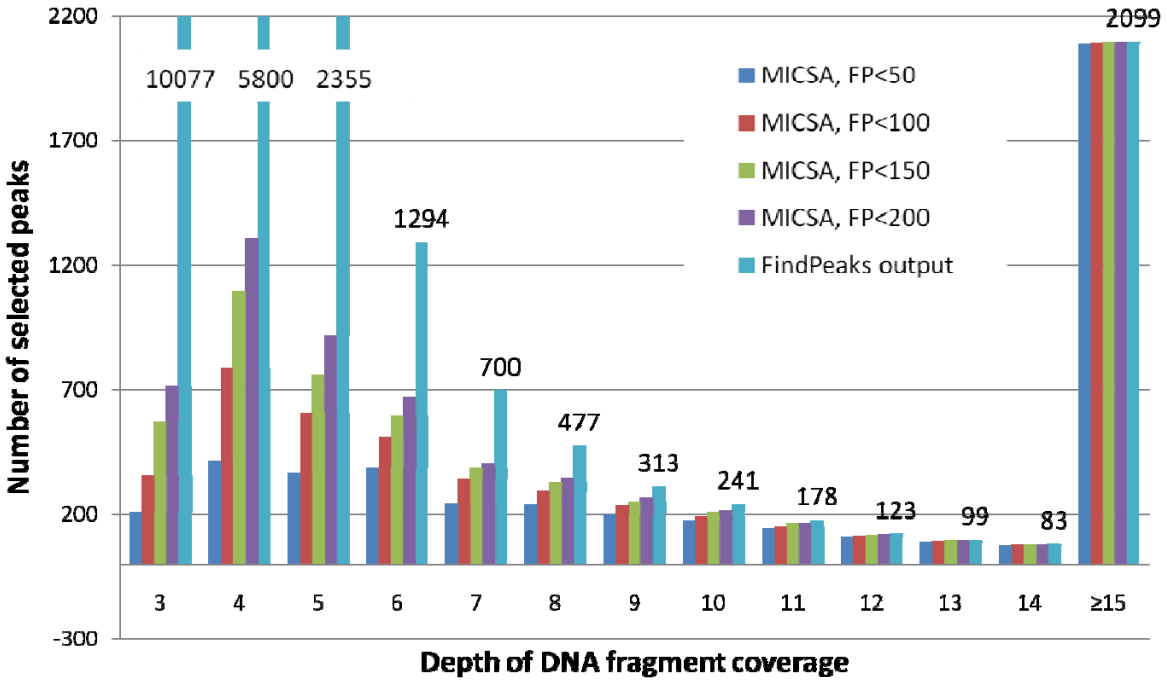
We divide segment]0, F] into very small sub-segments:]0, ε],]ε, 2ε],… ,]F-2ε, F- ε],]F- ε, F]. We set $\varepsilon = F/1000$. This gives 1000 sub-fragments. For the first class C_1 and for each segment]$k\varepsilon$, $(k+1)\varepsilon$] we find such ($T_{M,1}$, $\delta_{M,1}$) so that $k\varepsilon < F_1(T_{M,i}$, $\delta_{M,i}) \leq (k+1)\varepsilon$ and $S_1(T_{M,1}$, $\delta_{M,1})$ are maximal. We fill array B_1 with corresponding maximized values of $S_1(T_{M,1}$, $\delta_{M,1})$. We remember the choice of ($T_{M,1}$, δ_{M1}) for each sub-fragment. We create an array A with 1000 elements that should contain corresponding values of $S_i(T_{M,i}$, $\delta_{M,i})$ at each following step (this array is called A_i in step i). In the first step we fill array $A_1$with values from B_1: $A_1[j] = B_1[j]$. Then, for each following class C_i we repeat the same procedure, i.e., for each segment]$k\varepsilon$, $(k+1)\varepsilon$] we find such ($T_{M,i}$, δ_{Mi}) so that $k\varepsilon < F_i(T_{M,i}$, $\delta_{M,i}) \leq (k+1)\varepsilon$ and $S_i(T_{M,i}$, $\delta_{M,i})$ would be maximal. Corresponding maximized values of $S_i(T_{M,i}$, $\delta_{M,i})$ are held in array B_i. Then, we fill A_i as follows: $A_i[j] = \max_{k=\overline{1,j}}(A_{i-1}(k) + B_i(j-k))$. After the last step i_{last}, the value $A_{i_{last}}[1000]$ will contain a value close to $\max\left(\Sigma S_i(T_{M,i}, \delta_{M,i}) \middle| F_i\left(T_{M,i}, \delta_{M,i}\right) \leq F\right)$.

This procedure guarantees that at the end we will find such ($T_{M,i}$, $\delta_{M,i}$), so that

$$F - \varepsilon \cdot NumberOfClasses \leq \Sigma F_i\left(T_{M,i}, \delta_{M,i}\right) \leq \mathrm{F}, \text{ and}$$

$$\Sigma S_i\left(T_{M,i}, \delta_{M,i}\right) \geq (\text{real max}\, \Sigma S_i \mid \Sigma F_i \leq \mathrm{F} - \varepsilon \cdot NumberOfClasses).$$

Supplementary Fig. 6 shows results of the described optimization procedure for different values of F.



Supplementary Figure 6. Numbers of peaks reported by FindPeaks (default parameters) and by MICSA (different values of maximal number of false positives *F*) for the NRSF dataset[2].

Program execution

The tutorial on how to use the graphical interface of MICSA (Supplementary Fig. 7) can be found on the MICSA website http://bioinfo-out.curie.fr/projects/micsa/tutorial.html.

Below are instructions on how to run the MICSA pipeline from the command line.

1. Run FindPeaks for ChIP and control data:

```
java -Xmx2G -jar FindPeaks.jar -aligner <aligner> -eff_frac
<eff_frac> -duplicatefilter -input <ChIP input files> -name chip -
output <existing output directory> -dist_type <dist_type> -minimum 3

java -Xmx2G -jar FindPeaks.jar -aligner <aligner> -eff_frac
<eff_frac> -duplicatefilter -input <control input files> -name
control -output <existing output directory> -dist_type <dist_type> -
minimum 1
```

   More details on FindPeaks parameters can be found at http://vancouvershortr.wiki.sourceforge.net/FindPeaks4

2. Filter out peaks in repetitive or ambiguous regions of genome:

```
java DeleteRegions -f chip_triangle_standard.peaks -r <file with
positions to mask>

java deleteRegions -f control_triangle_standard.peaks -r <file with
positions to mask>
```

3. Create summary about peak distribution in ChIP and control data:

```
java Summary -f chip_triangle_standard.peaks -c
control_triangle_standard.peaks -r <ratio>
```

4. Filter out peaks occurring both in ChIP and Control data:

```
java FilterPeaks -f chip_triangle_standard.peaks -c
control_triangle_standard.peaks -t <coverage_threshold>
```

5. Run MICSA.jar :

```
java -jar -Xmx2G micsa.jar -name <name> -f
chip_triangle_standard.peaks -n <max_false_positive> -o <output_dir>
-l <file with summary> -g <genome_dir> -w <wig_file>
```

See the MICSA tutorial http://bioinfo-out.curie.fr/projects/micsa/tutorial.html for more details on parameters and an example of MICSA run on the NRSF dataset[2].

References are numbered according to the main text.