Local Enrichment of Genomic Annotations

Jakub Drobný[1], Broňa Brejová[1,*]

[1]Faculty of Mathematics, Physics and Informatics, Comenius University in Bratislava, Slovakia

Abstract

Genomic annotation identifies functional elements within a genome. Many annotations can be expressed as sets of non-overlapping intervals. To find a putative biological connection between two annotations, one may choose to compute how much their intervals overlap. Even in randomly generated annotations some overlap can be expected, and therefore we need to compare the observed overlap with a suitable null model. Many tools perform such analyses on the whole genome. In this work, we extend one of these algorithms to also identify regions with significant overlap within the genome. Our tool eMCDP is able to perform such analyses efficiently on any set of genomic windows, even overlapping ones. We use it to analyze human genome annotations, successfully identifying significantly enriched and depleted regions. The tool is available at https://github.com/jakubdrobny/e-mcdp.

Keywords

genome annotation, statistical significance, Markov model

1. Introduction

In this work, we provide a new tool for computing statistical significance of genome annotation overlaps in individual regions of the genome. A *genome* is a collection of genetic information of an organism. In bioinformatics, it is represented as a set of sequences over the DNA alphabet $\{A, C, G, T\}$. The process of *genome annotation* seeks to identify functional elements or other important positions within a genome. Examples include genes, gene regulatory sequences, regions conserved in evolution, mutations within or between species, binding sites of various proteins, and chemical epigenetic marks on the DNA or associated proteins.

In this article, we represent each annotation (corresponding to one of these functional classes) as a set of non-overlapping intervals of a genomic sequence. To find a potential biological connection between two annotations, we may choose to compute the amount of overlap between the regions they cover and to determine if this overlap is significantly larger or smaller than would be expected by chance in a suitable null model of independent annotations.

Several different null models were explored in the literature, providing a tradeoff between faithfulness to the observed data and computational efficiency [1, 2]. The simplest model characterizes an annotation just by the fraction p of the genome it covers, assuming that in the null model each position is covered by the annotation independently with probability p [3, 4]. This model allows simple computation of p-values for example by the Fisher exact test, but it does not capture the fact that annotation intervals may be long, making the independence assumption unrealistic. The other extreme are null models, which keep the number and individual lengths of intervals fixed and randomly shuffle the intervals along the chromosomes and potentially also permute their order [5, 6, 7]. With a fixed order of intervals, it is possible to compute the p-value by dynamic programming with pseudo-polynomial running time, which is slow in practice [8]. When we allow reordering, the problem of p-value computation becomes NP hard [9], and in practice it is solved by sampling from the model.

Here, we consider a Markov chain null hypothesis [9], which lies between these two extremes. In this hypothesis, one of the annotations is generated by a two-state Markov chain. This model can capture mean lengths of the annotation intervals and gaps between them, while allowing efficient algorithms

ITAT'25: Information Technologies – Applications and Theory, September 26–30, 2025, Telgárt, Slovakia
*Corresponding author.
brejova@fmph.uniba.sk (B. Brejová)
0000-0002-9483-1766 (B. Brejová)

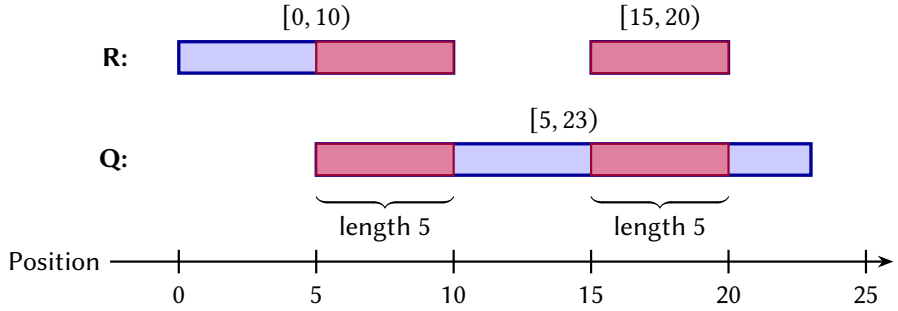

Figure 1: An example of colocalization statistics for reference annotation $R = \{[0, 10), [15, 20)\}$ and query annotation $Q = \{[5, 23)\}$. Both intervals in R intersect the only interval in Q, so $K(R, Q) = 2$. The intersection of R and Q are two intervals $[5, 10)$ and $[15, 20)$, both of length 5. The sum of lengths of these intervals is $B(R, Q) = 10$.

for computing or approximating p-values [9, 10]. However, these algorithms compute a single p-value for the whole genome. In this work, we consider computing p-values for individual windows of the genome to explore if enrichment of one annotation within the other happens throughout the whole genome or if it is concentrated in specific regions.

We adapt the original MCDP algorithm [9] for computing the probability mass function of the overlap size under the Markov chain null hypothesis to work on a single window. To compute p-values for a set of windows, we can run this algorithm on each window separately. However, if the input windows overlap, this approach repeatedly recomputes shared parts of the windows. Therefore we also provide two algorithms seeking to avoid this redundancy. The first one splits the genome into sections at window boundaries, computes necessary quantities for each section and then combines sections in each window. The second one uses the segment tree data structure [11] to precompute some joins and reuse them. We implemented these algorithms in a tool called eMCDP and compared their running time with the original MCDP implementation [9]. We also show the effect of window size on the ability of the model to detect significant overlaps. Lastly, we use our software to analyze real annotations of the human genome, identifying several regions of significant enrichment and depletion.

The Genomic HyperBrowser by Sandve et al. [7] also provides users with window-based analysis of statistical significance, but their windows are required to be non-overlapping, and the software does not support the Markov chain null model.

2. Preliminaries and problem definition

Basic notation. To simplify notation, we will consider a *single chromosome* of length L, which we represent as a sequence of nucleotides numbered $0, \ldots, L-1$. We will discuss multiple chromosomes at the end of this section. A *genomic interval* is a half-open interval $[\ell, r)$ with $0 \leq \ell < r \leq L$ describing a contiguous sequence of positions. An *annotation* A is a set of non-overlapping genomic intervals. Its size $|A|$ is defined as the number of its intervals and *weight(A)* denotes the sum of lengths of intervals in A.

Colocalization statistics and p-values. In colocalization analysis, we are given two genomic annotations, denoted reference R and query Q. We will measure their overlap by two statistics. The *overlap statistic* $K(R, Q)$ is defined as the number of intervals in R intersecting with at least one interval in Q. The *shared bases statistic* $B(R, Q)$ is defined as the sum of the lengths of the intervals in the intersection of R and Q. An example of both statistics can be seen in Figure 1.

To assess if the observed value k of a statistic is significant, we will use the null hypothesis testing. We first select a suitable *null hypothesis* $\mathscr{H}_0$, which assumes that there is no true association. Then we calculate the probability of the statistic achieving value k or even more extreme than k assuming the null hypothesis is true, also known as the *p-value*. We reject the null hypothesis if the p-value is lower

than some predefined threshold α, such as $\alpha = 0.01$. In this case, we assume that the overlap between the two annotations under $\mathscr{H}_0$ did not happen due to pure chance, and the two annotations might have an underlying biological connection.

The Markov chain null hypothesis. The Markov chain null hypothesis $\mathscr{H}_0^{MC}$ keeps the reference annotation R fixed and generates query annotation Q_{RAND} by a two-state Markov chain whose parameters are estimated from the observed query annotation Q [9]. State 0 of the Markov chain corresponds to nucleotides outside of the query annotation intervals and state 1 to those inside. For example, sequence of states $(0, 1, 1, 0, 1, 0, 0, 1, 1)$ represents annotation $\{[1, 3), [4, 5), [7, 9)\}$. The Markov chain uses the following transition matrix:

$$\begin{aligned} T_Q &= \begin{pmatrix} t_{00} & t_{01} \\ t_{10} & t_{11} \end{pmatrix} \\ &= \begin{pmatrix} 1 - t_{01} & \frac{|Q|}{L - weight(Q) - (1-b)} \\ \frac{|Q| - b}{weight(Q) - b} & 1 - t_{10} \end{pmatrix}, \end{aligned}$$

where t_{ij} is the probability of transitioning from state i to state j and b is a binary indicator with value 1 if the last nucleotide of the chromosome is covered by Q. Matrix T_Q maximizes the likelihood of the query annotation Q being generated by the Markov chain. It preserves the expected lengths of gaps and intervals [9]. We will set the starting distribution equal to the stationary distribution of the transition matrix T_Q as in the MCDP2 software [10]. This means that the probability of state 1 is the same at any position of the sequence (but states are not independent).

The MCDP software [9] computes the full probability mass function (pmf) of the $K(R, Q)$ statistic under this null hypothesis, i.e. the values $\Pr_{Q_{RAND} \sim \mathscr{H}_0^{MC}}[K(R, Q_{RAND}) = i]$ for $i = 0, \ldots, |R|$. It uses a dynamic programming algorithm running in time $\mathcal{O}(|R|^2)$. The required p-value is then computed as the sum of the tail of this distribution.

The newer version MCDP2 [10] includes several extensions. First, it also computes the pmf for the $B(R, Q)$ statistics by splitting each interval of R into adjacent intervals of size 1 and using the algorithm for $K(R, Q)$. However, this makes the running time quadratic in $weight(R)$ rather than $|R|$, which is significantly slower for longer intervals. Second, MCDP2 can approximate the pmf by the normal distribution. The exact mean and variance of this distribution can be computed in the running time linear in $|R|$ for the $K(R, Q)$ statistic and quasi-linear for the $B(R, Q)$ statistic, allowing fast computation of approximate p-values. Finally, it supports a richer parameterization of the null model, where the genome is segmented by the user into several classes of intervals called contexts and a separate transition matrix is estimated for each context. In this work, we will not consider such context-dependent models, and we will consider only the exact algorithm from MCDP.

Problem statement. Our goal is to provide algorithms for computing the p-values under the Markov chain null hypothesis not only for the genome as a whole but also for a collection of windows to explore local colocalization of R and Q. Each *window* is a genomic interval $w = [\ell, r)$. *Applying window* w on annotation A results in a new annotation $A' = \{[x, y) \mid \exists [b, e) \in A : [x, y) = [b, e) \cap w \wedge x < y\}$. In other words, we remove the intervals that are outside w and resize the intervals that are only partially intersecting it. Now we can define the problem we are going to solve in this work.

Problem statement 1. *We are given the length of the chromosome L, reference annotation R, query annotation Q and windows $w_0, \ldots, w_{t-1}$. The goal is to compute for each window w_i the p-value of the overlap $K(R_i, Q_i)$ under $\mathscr{H}_0^{MC}$, where annotations R_i and Q_i are the result of applying window w_i on annotations R and Q respectively. For enrichment, the p-value is $\Pr_{Q_{RAND} \sim \mathscr{H}_0^{MC}}[K(R_i, Q_{RAND}) \geq K(R_i, Q_i)]$, and for depletion, it is $\Pr_{Q_{RAND} \sim \mathscr{H}_0^{MC}}[K(R_i, Q_{RAND}) \leq K(R_i, Q_i)]$. The null hypothesis $\mathscr{H}_0^{MC}$ uses the same transition matrix T_Q estimated from the entire Q for all windows.*

We can easily adapt the algorithm of MCDP to run on a single window, but our goal is to provide algorithms that work more efficiently for overlapping windows by reusing some computations.

A typical example of overlapping windows are *sliding windows* defined by a window size W and a step S resulting in a set of windows $\{[0, W), [S, S+W), [2S, 2S+W), \ldots, [kS, kS+W), [(k+1)S, L)\}$, where $k = \lceil (L-W)/S \rceil - 1$. In contrast to non-overlapping bins used in previous work [7], sliding windows can better pinpoint locations exhibiting significant enrichment or depletion of overlaps between annotations. Our algorithms work for arbitrary window sets. In addition to sets with a regular structure, one may consider biologically-motivated window sets, such as topologically associating domains (TADs) [12] which have a hierarchical structure or regions delineated by recombination hotspots [13].

Multiple chromosomes. Our notation and problem statement assumed that the annotations only contain intervals from a single chromosome, but in many cases, the annotations cover multiple chromosomes. To compute the overall p-value for the whole genome, it is usually assumed that the chromosomes are independent under the null model. The pmfs computed for individual chromosomes can be, therefore, combined by a straightforward convolution [8, 9]. In our problem, each genomic window is located on a single chromosome. Therefore we can apply our algorithms for each set of windows located on a single chromosome to obtain their p-values.

3. Algorithms

In our software, we implemented three algorithms. The `naive` algorithm runs an adapted version of the original MCDP algorithm for each window separately. The two other algorithms first split the chromosome into disjoint *sections*, with section boundaries at the starts and ends of all windows. Then we compute some quantities called *multiprobs* for each section, and we provide a procedure for joining two adjacent sections into a single section. The `section-based` algorithm then takes sections of each window and combines them sequentially to obtain the pmf for the whole window. The `tree-based` algorithm always does at most logarithmic number of section joins per window by applying the segment tree data structure [11].

The original MCDP and MCDP2 algorithms. For completeness, we start by summarizing the original MCDP algorithm [9] with some minor modifications from MCDP2 [10]. At its core is a dynamic programming algorithm which gets as inputs a Markov chain transition matrix T, the initial state distribution π applied at position -1 and the reference annotation $R = \{[b_1, e_1), \ldots, [b_m, e_m)\}$ ordered by interval starts. We will define $e_0 = 0$. Denote the state sequence generated by the Markov chain as $S_0, \ldots, S_{L-1}$.

The algorithm computes quantities $P_{DP}[j, \kappa, s]$ for $0 \leq j \leq m$ and $0 \leq \kappa \leq m$ and $s \in \{0, 1\}$. This quantity is defined as the probability that if we run the Markov chain to generate $S_0, \ldots, S_{e_j - 1}$ (i.e. finishing at the end of the j-th interval of R), it will intersect exactly κ intervals of R and finish in state s.

The base cases are defined for $j = 0$ as $P_{DP}[0, 0, s] = \pi_s$ and $P_{DP}[0, \kappa, s] = 0$ for $\kappa > 0$. To simplify the main recurrence, $P_{DP}[j, -1, s]$ is also defined as 0. The main recurrence for $j > 0$ is computed as follows:

$$\begin{aligned} P_{DP}[j, \kappa, s] = {} & P_{DP}[j-1, \kappa, 0] P_{nohit}(j, 0, s) \\ & + P_{DP}[j-1, \kappa, 1] P_{nohit}(j, 1, s) \\ & + P_{DP}[j-1, \kappa-1, 0] P_{hit}(j, 0, s) \\ & + P_{DP}[j-1, \kappa-1, 1] P_{hit}(j, 1, s), \end{aligned}$$

where $P_{nohit}(j, s, t)$ is defined as the probability that $S_{e_j - 1} = t$ and $S_x = 0$ for all x such that $b_j \leq x < e_j$, given that $S_{e_{j-1} - 1} = s$. This corresponds to the case that the generated sequence of states does not hit the j-th reference interval. Similarly, $P_{hit}(j, s, t)$ is defined as the probability that $S_{e_j - 1} = t$ and $S_x = 1$ for at least one x such that $b_j \leq x < e_j$, given that $S_{e_{j-1} - 1} = s$. This corresponds to the case that the generated sequence of states hits the j-th reference interval. Quantities $P_{nohit}(j, s, t)$

and $P_{hit}(j,s,t)$ can be computed in $\mathcal{O}(1)$ time [9], leading to the overall $\mathcal{O}(|R|^2)$ running time of the whole dynamic programming algorithm. The desired pmf for the whole chromosome is obtained as $\Pr(K(R,Q_{RAND})=\kappa) = P_{DP}[m,\kappa,0] + P_{DP}[m,\kappa,1]$.

The MCDP2 algorithm [10] computes the mean and variance of the selected statistic without computing the full pmf. It computes a data structure called *plumbus* for each reference interval and for the gaps between them. A plumbus stores the mean and variance for all combinations of the starting and ending states of the Markov chain at the endpoints the interval. It also stores the probability distribution of the ending state for different values of the starting state. Such plumbuses can be then combined in constant time into plumbuses for longer intervals, leading to overall linear running time for the $K(R,Q)$ statistic.

Our multiprobs data structure. We will now introduce the *multiprobs* data structure inspired by the plumbus data structure from MCDP2, but adapted for computing the full pmf.

Let us now consider a genomic interval $w=[b,e)$ and let R' be the annotation obtained by applying window w on R. The *multiprobs* data structure for interval w is a table $A^w_{s,t,\kappa}$ for $s,t \in \{0,1\}$ and $0 \le \kappa \le |R'|$. Value $A^w_{s,t,\kappa}$ is the probability $\Pr(K(R',Q_{RAND})=\kappa, S_{e-1}=t \mid S_{b-1}=s)$, i.e. the probability of Q_{RAND} overlapping κ intervals of R' and ending in state t at position $e-1$, given that we started in state s at position $b-1$.

To compute multiprobs, we run the MCDP algorithm twice, with starting distributions $(1,0)$ and $(0,1)$ at position $b-1$ to compute $A^w_{0,t,\kappa}$ and $A^w_{1,t,\kappa}$ respectively. We modify the base case to apply to position $b-1$ instead of -1, but we also need to add computation of the probability of ending in state t at position $e-1$, whereas MCDP ignores the states of the Markov chain after the end of the last interval, as they do not influence the overlap. In our case, these probabilities are needed to correctly join adjacent intervals. This extension can be easily done in $\mathcal{O}(1)$ time. The overall running time is $\mathcal{O}(|R'|^2)$.

To obtain the pmf for window w from its multiprobs A^w assuming the Markov chain starts in its stationary distribution π, we use the law of total probability to marginalize over the starting state s and ending state t:

$$\Pr(K(R',Q_{RAND})=\kappa) = \sum_{s\in\{0,1\}} \sum_{t\in\{0,1\}} \pi_s \cdot A^w_{s,t,\kappa}.$$

Joining adjacent multiprobs. Consider two adjacent intervals $w_1=[b,d)$ and $w_2=[d,e)$. Given multiprobs A^{w_1} and A^{w_2} of these intervals, we want to compute multiprobs A^w for interval $w=[b,e)$. Let R' be the annotation obtained by applying w on R and for $i\in\{1,2\}$ let R_i be the annotation obtained by applying w_i on R.

First let us assume that no interval of R spans the boundary point d. This means that for any Q', $K(R',Q') = K(R_1,Q') + K(R_2,Q')$. To compute $A^w_{s,t,\kappa}$, we marginalize over the state y at the boundary point d and over κ_1 which is the value of the statistics in w_1.

Thanks to the Markov property, events in window w_2 are independent of how the state S_{d-1} was reached, given that $S_{d-1}=y$. Namely, assuming $S_{d-1}=y$, events $K(R_2,Q_{RAND})=\kappa-\kappa_1$ and $S_{e-1}=t$ are independent of events $K(R_1,Q_{RAND})=\kappa_1$ and $S_{b-1}=s$. This allows us to obtain the following equation for $A^w_{s,t,\kappa}$:

$$A^w_{s,t,\kappa} = \sum_{y\in\{0,1\}} \sum_{\kappa_1=\max(0,\kappa-|R_2|)}^{\min(\kappa,|R_1|)} A^{w_1}_{s,y,\kappa_1} \cdot A^{w_2}_{y,t,\kappa-\kappa_1} \tag{1}$$

Now let us assume that there is an interval $[\ell,r)\in R'$ which spans the boundary point d between w_1 and w_2, i.e. $\ell<d<r$. Formula (1) cannot be directly applied in this case because $K(R',Q')$ is either $K(R_1,Q')+K(R_2,Q')$ or $K(R_1,Q')+K(R_2,Q')-1$, depending on whether Q' overlaps with $[\ell,r)$.

To overcome this issue, we split w into three parts: $w_b=[b,\ell)$, $w_d=[\ell,r)$ and $w_e=[r,e)$. Notice that no interval in annotation R crosses either of the boundary points ℓ and r. We can therefore join A^{w_b}, A^{w_d} and A^{w_e} into A^w by two applications of formula (1).

As a special case, we will sometimes construct multiprobs for an empty interval $w=[b,b)$. In this case, we need to consider only $\kappa=0$ and we set $A^w_{s,t,0}=1$ for $s=t$ and $A^w_{s,t,0}=0$ for $s\neq t$. By joining

such empty A^w with an adjacent multiprobs $A^{w'}$ for interval $w' = [b, e)$ or $w' = [d, b)$, we obtain again $A^{w'}$, and therefore such A^w is an identity element for the operation of joining two multiprobs.

Note that the operation of joining multiprobs has a time complexity of $\mathcal{O}((|R_1| + 1)(|R_2| + 1))$. The multiprobs for w_d can be computed in $\mathcal{O}(1)$ because it involves only a single interval of R.

Sections. To deal with overlapping input windows, we use *section* data structure. We split the whole chromosome into intervals at each position that starts or ends one of the windows and build one section for each such interval. For each section, we keep the following data.

- Interval $I = [b, e)$ that the section describes.
- Interval $J = [b_J, e_J) \subseteq I$ that is obtained from I by removing the parts covered by the reference intervals spanning the boundaries of I, if such intervals exist. If no interval spans the boundaries of I, then $J = I$. In the special case that the entire I is covered by a single interval that also spans both boundaries, J will be an empty interval $[e, e)$. We also keep a pair of booleans indicating if a particular boundary of I is spanned by an interval.
- Annotations R' and Q' obtained by applying window I on R and S, respectively.
- The value of the statistic $k = K(R', Q')$.
- Multiprobs for interval J.

Given two adjacent sections for intervals $w_1 = [b, d)$ and $w_2 = [d, e)$, we can easily build a section for interval $w = [b, e)$. We have already described the process of joining the corresponding multiprobs. The value of statistic k is added from sections w_1 and w_2 and potentially adjusted for double counting of the interval spanning boundary d. Annotations R', Q' and intervals I, J can be also easily combined from the two sections.

Algorithms. The running time of our algorithms can be expressed in terms of parameters $n = |Q|$, $m = |R|$ and t being the number of input windows.

All three algorithms first sort intervals of Q in $\mathcal{O}(n \log n)$ time and estimate the parameters of the Markov model from annotation Q in $\mathcal{O}(n)$ time.

The `naive` algorithm runs the original MCDP algorithm for each window separately, which takes $\mathcal{O}(tm^2)$ time.

The `section-based` and `tree-based` algorithms instead continue by computing initial sections. Section boundaries are found by sorting $2t$ endpoints of input windows. We then compute the subsets of annotations R and Q corresponding to each section, which can be done by a linear pass through sorted annotations. Computation of the overlap statistic in all sections also takes linear time. Computing multiprobs for a section with m_i reference intervals takes $\mathcal{O}((m_i + 1)^2)$ time. The sum $\sum_i m_i$ for all sections is at most $m + 2t$, because each section boundary can split a reference interval into two parts. Overall the running time of computing multiprobs for all sections is at most $\mathcal{O}((m + t)m)$ because $\sum_i (m_i + 1)^2 \leq (m + 1) \sum_i (m_i + 1) \leq (m + 1)(m + 4t)$.

The `section-based` algorithm then joins sections of each window sequentially. The total number of joining operations is thus $\mathcal{O}(t^2)$. However, considering that one joining operation takes $\mathcal{O}((m_i + 1)m)$ time, the running time of all joining operations for all windows is $\mathcal{O}(tm(m + t))$.

The `tree-based` algorithm instead builds a segment tree [11, 14]. This is a binary tree which has sections at its leaves. Each internal node contains a multiprobs for the union of its two children intervals. The tree has height $\mathcal{O}(\log t)$ and at each depth it covers the entire chromosome. Therefore multiprobs joining at each depth takes $\mathcal{O}(m(m + t))$ time, resulting in $\mathcal{O}(m(m + t) \log t)$ total time. Then we can express each window as a union of a logarithmic number of sections from the segment tree and join these sections together in $\mathcal{O}(tm(m + \log t))$ total time for all windows. We perform $\mathcal{O}(t)$ join operations for building the tree and $\mathcal{O}(t \log t)$ for computing results for individual windows.

Although the `tree-based` algorithm does only $\mathcal{O}(t \log t)$ joining operations compared to $\mathcal{O}(t^2)$ for the `section-based` one, the overall running time is cubic in the term $N = m + t$ for all three algorithms, because it is dominated by the largest joining operations.

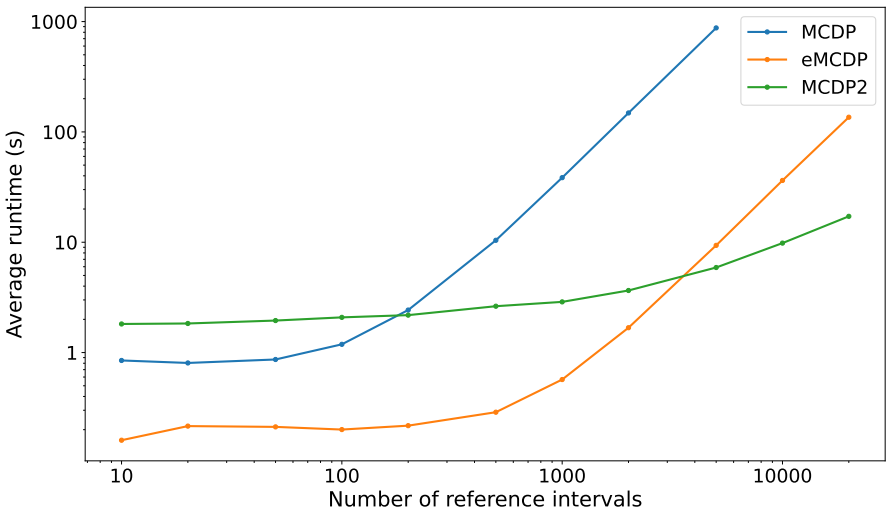

Figure 2: Running time of the eMCDP, MCDP and MCDP2 tools as a function of $|R|$. We plot the average time across 20 random inputs for each input size. Both axes are logarithmic.

4. Experiments

We implemented all three algorithms in a C++ software tool eMCDP available at https://github.com/jakubdrobny/e-mcdp. The reproducibility data is available at https://github.com/jakubdrobny/e-mcdp-reproducibility. In addition to the three variants of the algorithm, eMCDP can also be used to perform whole-genome analysis as with the previous MCDP software [9] and the MCDP2 software [10]. When we compute p-values for multiple windows of a genome, our software also outputs p-values adjusted by the Bonferroni multiple testing correction, where the number of tests is the number of windows in a single run of the tool.

In this section, we evaluate our software in terms of running time and its power to detect enrichments, and we apply it to real annotations of the human genome. All experiments have been run on a laptop with Intel© Core™ i7-9750H CPU at 2.60GHz x 12 and 32 GB DDR4 RAM.

Comparison with the original MCDP implementation. Figure 2 shows the running time comparison of the MCDP [9], MCDP2 [10] and eMCDP with different reference annotation sizes. Here, no window set it used, and all three algorithms analyse the whole genome. For this experiment, we used the `01-synthetic` data sets from the MCDP2 study [10]. These data sets contain randomly generated reference and query annotations along a single chromosome of length 100Mbp, with all intervals having length 500bp. All query annotations have 50000 intervals, while the size of the reference annotation varies from 10 to 20000. For each reference size, 20 pairs of inputs were averaged. MCDP was run only for inputs with $|R| \leq 5000$ due to its slow running time. We see that eMCDP is orders of magnitude (~100x) faster than the original MCDP algorithm [9] and for small annotations of sizes up to 5000 is faster than even the linear-time MCDP2 algorithm [10], which does not compute the exact p-values.

Comparison of algorithms. Here, we compare the performance of the `naive`, `section-based` and `tree-based` variants with a sliding window size of 1Mbp and a window step of 50kbp, that is, each window is made up of 20 non-overlapping sections. For this experiment, we again used the `01-synthetic` data set from the MCDP2 study [10]. We ran the `tree-based` algorithm on reference annotations of sizes up to 5000 and the `naive` and `section-based` variants on reference annotations of sizes up to 20000. Figure 3 shows that the `tree-based` variant is substantially slower than the other two algorithms. This is likely due to the large overhead of the segment tree which is built for the whole chromosome and thus contains also costly multiprobs for sections longer than the longest window. We can also observe that the `section-based` variant slightly outperforms the `naive` variant for large

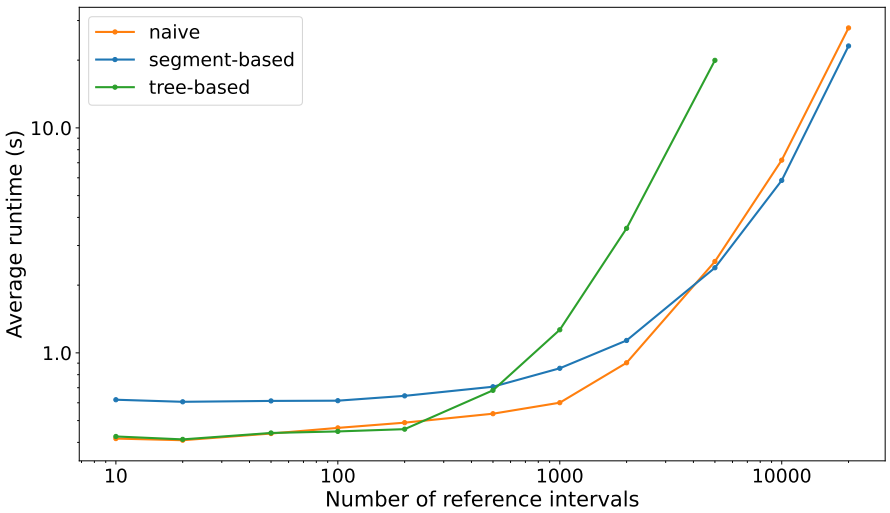

Figure 3: Running time of different algorithms as a function of $|R|$. Each sliding window consists of 20 sections. We plot the average time across 20 random inputs for each input size. Both axes are logarithmic.

inputs.

Enrichment detection. We now examine the power of eMCDP to detect local colocalizations. We generated annotations along a genome consisting of two chromosomes of length 1Mbp with genome coverage $c = 0.1$ and a fixed interval length $\ell = 100$. The annotations were generated dependently along the first chromosome and independently along the second chromosome. To generate an annotation with a coverage c and length ℓ, we simply iterated over all chromosomal positions and at each position the probability of a new interval starting is c/ℓ. If a new interval was generated, we continued the process from the last position of the newly generated interval instead of the next position. To produce independent annotations, we used this method to generate both annotations.

To generate dependent annotations with a dependency factor $d \geq 1$, we first generate R. To generate Q, we again iterate over all positions and at each position we first check if it is covered R. If it is, the probability of a new interval starting at this position is multiplied by the dependency factor d. At positions not covered by an interval from the reference annotation, the probability of a new interval starting is multiplied by $1/d$.

We ran the software with a window size of 1Mbp, therefore covering each chromosome with a single window. We generated pairs of annotations with dependency factors ranging from 1, i.e. the annotations are truly independent to a dependency factor of 2. For each dependency factor we generated 10 pairs of reference and query annotations and took the median p-value of the 10 tests. Figure 4 shows that the significance of the overlap of the annotations along the first chromosome is increasing with the increasing dependency factor. The overlap of the annotations along the second chromosome is not significant, as expected.

Effect of the window size. To examine the effect of the window size on the ability of our software to detect a significant overlap, we generated annotations along a genome consisting of a single chromosome of length 1Mbp with a coverage of 0.1. The annotations were generated independently along most of the chromosome with the exception of the range of positions 400kbp to 600kbp, where the dependency factor d was set to 10 to make the overlap of the reference and query annotations really significant. We generated 10 pairs of reference and query annotation and ran our software with different window sizes in the range of 10kbp to 800kbp and a window step of 5kbp or 10% of the window size, whichever was smaller. For each window in each test, we plot the median adjusted p-value across all 10 tests.

The results are shown in Figure 5, separately for windows shorter and longer than the target region of length 200kbp. For long windows, our detection ability increases as we decrease the window size

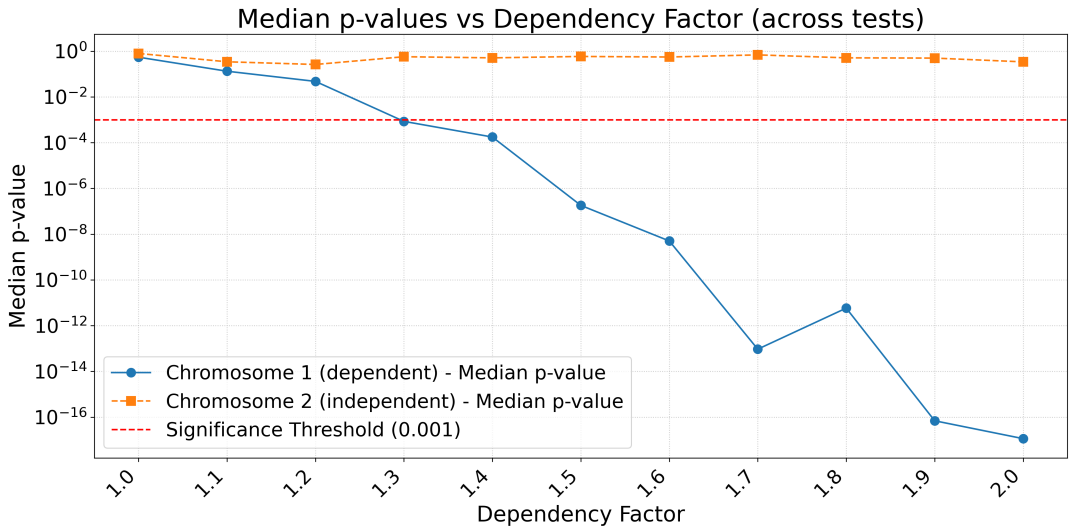

Figure 4: The power of eMCDP to detect overlap as a function of the dependency factor. We show the median p-value of the significance of the overlap of annotations in two chromosomes across 10 tests for each dependency factor. Annotations along chromosome 1 are generated dependent with increasing dependency factor and independent along chromosome 2.

to match the size of the region. This is due to the fact that longer windows cover more positions of the genome where the annotations are independent, and thus they have a higher chance of overlap occurring due to pure chance.

If we consider windows shorter than the target region, our ability to detect the significance is the best with window sizes slightly smaller than the size of the region with the significant overlap. Very small windows contain only a small number of intervals, and thus the probability of overlap occurring due to pure chance increases.

A real data set. To evaluate eMCDP on real annotations, we used the Zarrei et al. (2015) [15] data set from the original MCDP study [9]. This data set compares copy number losses (23438 intervals) and exons of the Refseq-annotated genes in the human genome. We consider exons of all non-coding genes (9975 intervals after merging overlaps), exons of all protein coding genes (208558 intervals after merging) and exons of all genes (216813 intervals after merging). The annotation of copy number losses was used as R and one of the three gene annotations as Q. We tested for both enrichment and depletion with sliding windows of size 1Mbp and step 100kbp. We used p-value threshold of 0.01 after Bonferroni multiple testing correction.

Table 1 shows that significant enrichment was only detected when comparing the annotation of copy number losses with the annotation of exons of all non-coding genes. This enrichment was found in two overlapping windows located on chromosome 15, which contain a dense cluster of 76 CD Box snoRNA non-coding genes which overlap 5 loss intervals (see Figure 6). The expected number of overlaps in these windows is only 0.2 with standard deviation 0.4. Both the Zarrei [15] and MCDP [9] studies reported enrichment when comparing the annotations of copy number losses and exons of all non-coding genes. However, the counts in this region are relatively low to be a significant contribution the overall enrichment. We tried increasing the window size to 10Mbp, since there were at most reference 13 intervals present in any of the 1Mbp windows, but that resulted in no windows with significant enrichment.

When analyzing depletion, we detected significant results for almost all chromosomes when comparing the annotations of copy number losses and exons of all protein coding genes. Both the Zarrei [15] and MCDP [9] studies detected significant depletion when comparing these annotations, a finding which our results also support. An example of a depleted region on chromosome 21 is shown in Figure 7. The annotation of exons of all protein coding genes is depleted around position 18Mbp from chromosome

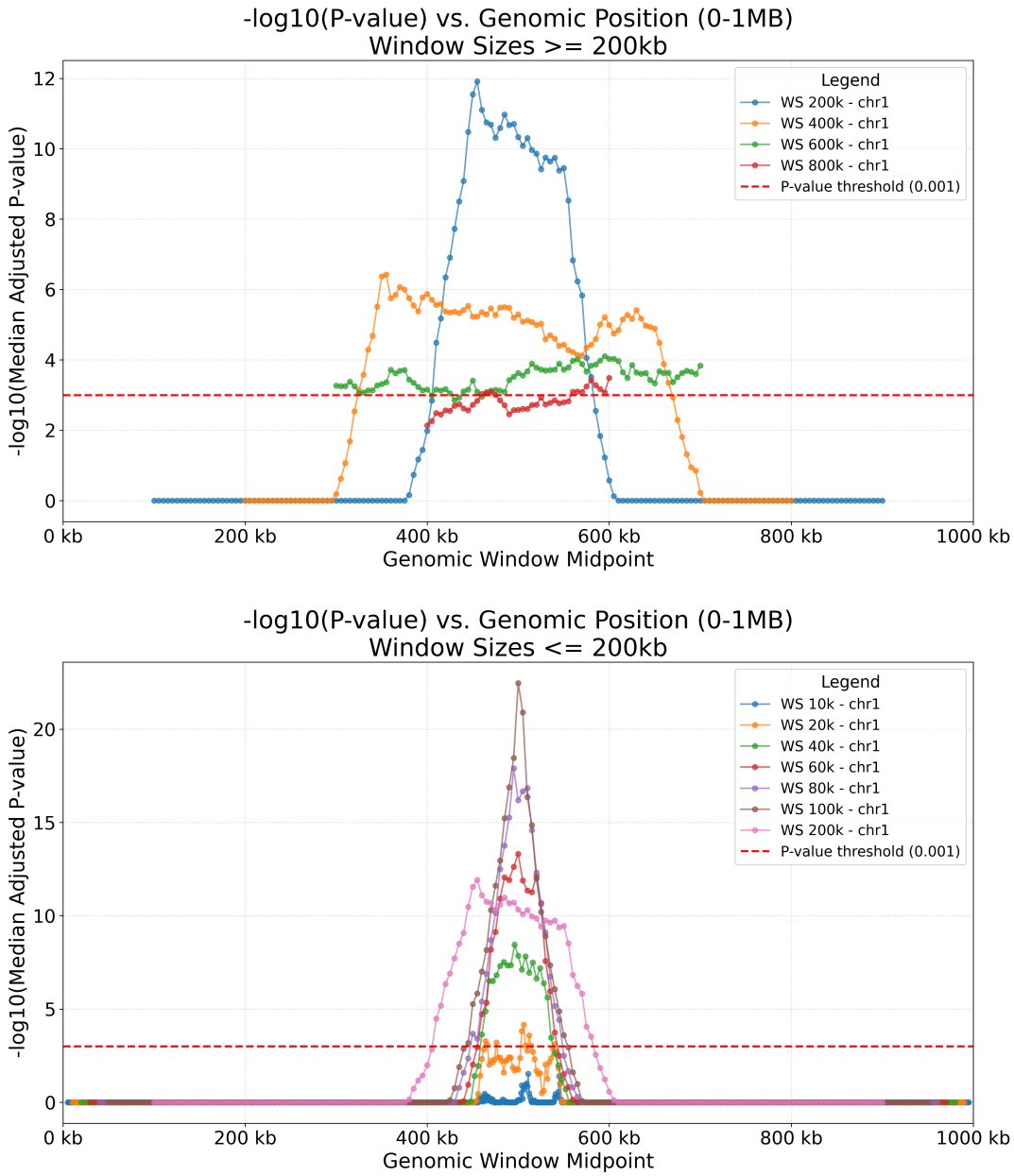

Figure 5: Effect of the windows size on colocalization detection. We plot the median of negative logarithm of the p-value across 10 tests for different window sizes. The true enrichment is located at positions 400kbp to 600kbp. Top: windows longer that the enriched region, bottom: windows shorter than the enriched region.

start. This regions contains a long loss interval (1.3Mbp). As a result, the expected number of overlaps is 1 with the standard deviation 10^{-10}, but the actual number of overlaps is 0.

5. Conclusion

In this work, we developed three algorithms for colocalization analysis of two genomic annotations in genomic windows. This allows the user to determine if any enrichment or depletion found on the whole-genome level is concentrated in specific genomic regions.

All three algorithms compute the full probability mass function for each window. Although two of the algorithms try to reduce redundant computations in overlapping windows, the running time is dominated by the quadratic term needed to combine the largest sections, and thus these changes do not reduce the running time. Indeed, the algorithm using segment trees was by far the slowest. This can be

Table 1
Summary of eMCDP results on a real data set by Zarrei et al. (2015) [15] using 1Mbp windows. Copy number losses were compared with exons of all genes, non-coding genes and protein-coding genes for enrichment (E) and depletion (D). The third column shows the number of windows with significant enrichment or depletion (adjusted p-value < 0.01), the fourth column shows the number of chromosomes with at least one significantly enriched or depleted window, the fifth column shows the most significant p-value, and the last column shows the running time of each test.

Query annotation	Test type	Significant windows	Significant chromosomes	Max $-\log_{10}(P_{adj})$	Time (s)
All genes (exon)	E	0	0	-	5.58
Non-coding (exon)	E	2	1	2.04	4.16
Prot-coding (exon)	E	0	0	-	5.37
All genes (exon)	D	470	22	87.53	5.51
Non-coding (exon)	D	0	0	-	4.15
Prot-coding (exon)	D	597	22	84.79	5.66

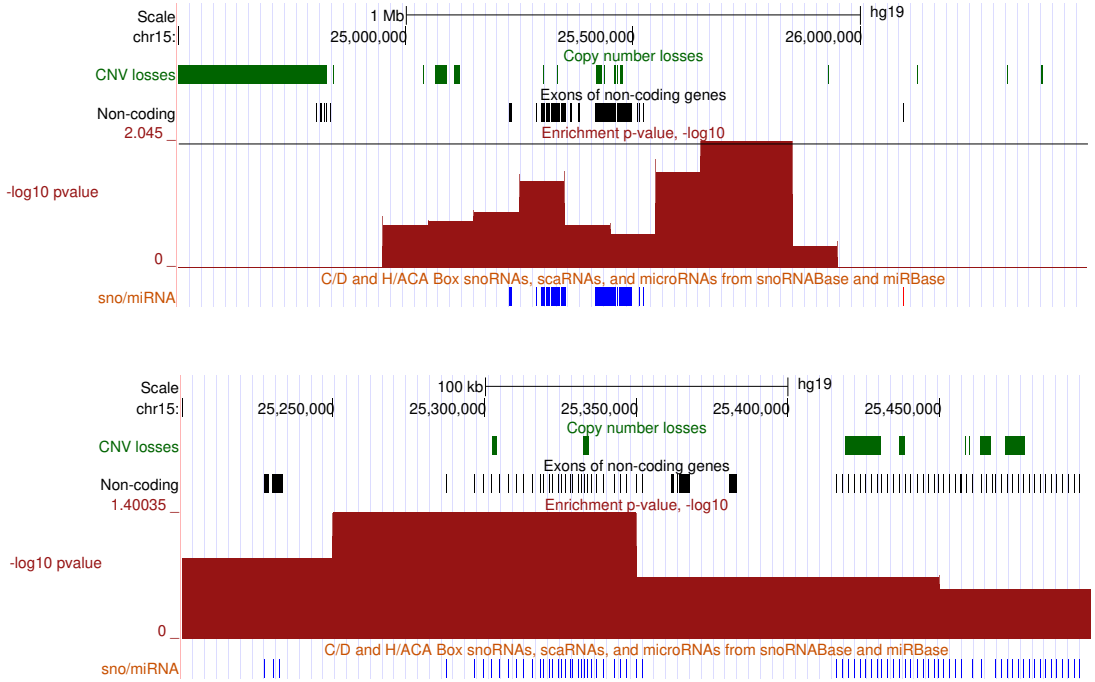

Figure 6: An example of enrichment analysis of the copy number losses and exons of all non-coding genes visualized in the UCSC Genome Browser [16] for a part of the human chromosome 15. The figures show both annotations as well as a plot of the negative logarithm of adjusted p-values. The p-value is plotted in the middle 100kbp of each 1Mbp window. Top: region chr15:24,500,000-26,500,000 (2Mbp); bottom: region chr15:25,200,000-25,500,000 (300kbp).

explained by the algorithm computing the full tree, even long intervals that will never be used and are costly to compute. An improvement would be to compute multiprobs in the tree on demand so that these redundant multiprobs are never computed.

On the other hand, the tree-based approach would be very efficient if we replaced computation of the multiprobs with the plumbuses from MCDP2 [10], which can be joined in constant time. Thus the reduction of the number of join operations from $\mathcal{O}(t^2)$ to $\mathcal{O}(t \log t)$ would bring time savings. However, normal approximation of p-values computed using plumbuses is appropriate only for data sets with a large number of intervals, so that this approach would be applicable only to very long windows.

Our algorithms can work for any set of input windows. Nonetheless, typical windows required by an user are likely to be sliding windows of constant size and step. In this special case, one could replace the segment tree with sliding window algorithms appropriate for an arbitrary associative operation,

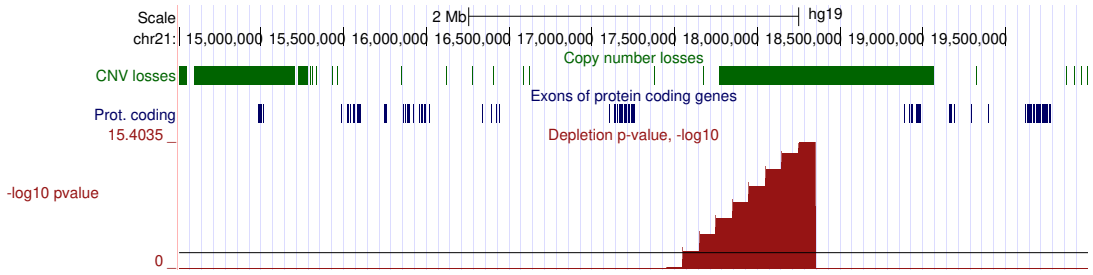

Figure 7: An example of depletion analysis of the copy number losses and exons of all protein-coding genes visualized in the UCSC Genome Browser [16] for a part of the human chromosome 21. The figure shows both annotations as well as a plot of negative logarithm of adjusted p-values. The p-value is plotted in the middle 100kbp of each 1Mbp window.

such as the De-Amortized Banker's Aggregator [17], which would require only $\mathcal{O}(t)$ joining operations for all windows combined.

Another desired extension of our current implementation would be the addition of context-dependent models from MCDP2. These models can avoid artifacts stemming from assembly gaps, and they can also help to study the influence of confounding factors, such as the GC content.

Acknowledgments

This research was supported in part by the Slovak Grant Agency VEGA grants 1/0538/22 and 1/0140/25 and by the Slovak Research and Development Agency grant APVV-22-0144. The research was also supported by the EU Horizon 2020 programme under the Marie Skłodowska-Curie grant agreement No. 872539 (PANGAIA).

Declaration on Generative AI

During the preparation of this work, the authors used Generative AI tool Gemini 2.5 Pro in Google AI Studio in order to: Grammar and spelling check. After using these tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] M. G. Dozmorov, Epigenomic annotation-based interpretation of genomic data: From enrichment analysis to machine learning, Bioinformatics 33 (2017) 3323–3330.

[2] C. Kanduri, C. Bock, S. Gundersen, E. Hovig, G. K. Sandve, Colocalization analyses of genomic elements: Approaches, recommendations and challenges, Bioinformatics 35 (2019) 1615–1624.

[3] M. G. Dozmorov, L. R. Cara, C. B. Giles, J. D. Wren, GenomeRunner web server: regulatory similarity and differences define the functional impact of SNP sets, Bioinformatics 32 (2016) 2256–2263.

[4] N. C. Sheffield, C. Bock, LOLA: enrichment analysis for genomic region sets and regulatory elements in R and Bioconductor, Bioinformatics 32 (2016) 587–589.

[5] B. Gel, A. Diez-Villanueva, E. Serra, M. Buschbeck, M. A. Peinado, R. Malinverni, regioneR: an R/Bioconductor package for the association analysis of genomic regions based on permutation tests, Bioinformatics 32 (2016) 289–291.

[6] A. Heger, C. Webber, M. Goodson, C. P. Ponting, G. Lunter, GAT: a simulation framework for testing the association of genomic intervals, Bioinformatics 29 (2013) 2046–2048.

[7] G. K. Sandve, S. Gundersen, H. Rydbeck, et al., The genomic HyperBrowser: Inferential genomics at the sequence level, Genome Biology 11 (2010).

[8] S. Sarmashghi, V. Bafna, Computing the statistical significance of overlap between genome annotations with iStat, Cell Systems 8 (2019) 523–529.e4.
[9] A. Gafurov, B. Brejová, P. Medvedev, Markov chains improve the significance computation of overlapping genome annotations, Bioinformatics 38 (2022) i203–i211.
[10] A. Gafurov, T. Vinař, P. Medvedev, B. Brejová, Efficient analysis of annotation colocalization accounting for genomic contexts, in: Research in Computational Molecular Biology, Springer Nature Switzerland, Cham, 2024, pp. 38–53.
[11] J. L. Bentley, Solutions to Klee's rectangle problems, Unpublished manuscript (1977) 282–300.
[12] J. R. Dixon, S. Selvaraj, F. Yue, A. Kim, Y. Li, Y. Shen, M. Hu, J. S. Liu, B. Ren, Topological domains in mammalian genomes identified by analysis of chromatin interactions, Nature 485 (2012) 376–380.
[13] S. Myers, L. Bottolo, C. Freeman, G. McVean, P. Donnelly, A fine-scale map of recombination rates and hotspots across the human genome, Science 310 (2005) 321–324.
[14] M. De Berg, O. Cheong, M. van Kreveld, M. Overmars, Computational geometry: algorithms and applications, Springer, 2008.
[15] M. Zarrei, J. R. MacDonald, D. Merico, S. W. Scherer, A copy number variation map of the human genome, Nature Reviews Genetics 16 (2015) 172–183.
[16] G. Perez, G. P. Barber, A. Benet-Pages, et al., The UCSC Genome Browser database: 2025 update, Nucleic Acids Research 53 (2024) D1243–D1249.
[17] K. Tangwongsan, M. Hirzel, S. Schneider, In-order sliding-window aggregation in worst-case constant time, The VLDB Journal 30 (2021) 933–957.