DREME: motif discovery in transcription factor ChIP-seq data

David Meyer

Department of Computer Science, Free University of Berlin, Takustr. 9, 14195 Berlin, Germany
E-mail: *david.meyer@fu-berlin.de*

ABSTRACT

Durch die große Anzahl von Sequenzen lässt sich das Problem der Motiv-Suche in ChIP-Seq Datensätzen nicht mehr exakt lösen, wenn Wildcards erlaubt sind. Daher implementiert DREME eine Heuristik die gute Ergebnisse liefert, indem unter Annahme von unabhängigen Motiven die Vorkommnisse geschätzt werden und über den errechneten p-Wert des Fisher-Exakt-Tests ein Abbruch-Kriterium besteht. Der Algorithmus wurde über einen ChIP-Seq Datensatz aus dem GalGal13-Genom getestet.

1 INTRODUCTION

ChIP-Seq dient der Sequenzierung von Transkriptionsfaktorbindestellen im Genom, sowie anderer DNA-bindender Proteine. Hierzu werden die Interaktionen mittels Formaldehyd fixiert, sodass bei der Fragmentierung durch Ultraschall die Interaktion bestehen bleibt und Segmente der Länge 100 bis 500 bp entstehen. Zur Isolation der interessanten Fragmente wird die Immunpräzipitation genutzt, indem spezifische Antikörper ihr Antigen (das DNA-bindende Protein) auf konzentrieren. Die relevanten Fragmente werden sequenziert und gegen das Genom gemappt, sodass Regionen mit überlappenden Sequenzreads als Peaks bestimmt werden können (Liu *et al.*, 2010). Zur Identifikation häufig vorkommender Motive in den Peaks wird die Heuristik DREME implementiert, die über reguläre Ausdrücke statistisch signifikante Motive erhebt (Bailey, 2011).

2 METHODS

DREME (discriminative regular expression motif elicitation) dient der Auffindung von kleinen (3 bis 8 bp) Motiven in einer Vielzahl von kleinen Sequenzen (100 bis 300 bp). Der Algorithmus startet mit der exakten Suche der vorhandenen Teilsequenzen der Länge 3 bis 8 (k-mer) der Eingabedatei. Um Probleme mit sich selbst überlappenden Motiven zu vermeiden wird für jedes k-mer die Anzahl der Sequenzen gezählt, in der dieses vorkommt und nicht die Anzahl der gesamten Vorkommnisse. Nachdem alle Teilsequenzen durchgegangen sind, wird mittels des Fisher-Exakt-Tests die Signifikanz berechnet, sodass die k-mere in einer nach p-Wert sortierten Liste Platz finden.

Der Fisher-Exakt-Test wird über die Hypergeometrische Verteilung bestimmt:

$$\frac{\binom{a+c}{a}\binom{b+d}{b}}{\binom{a+b+c+d}{a+b}}$$

a = Vordergrund-Zähler — b = Hintergrund-Zähler
$a+c$ = Sequenz-Anzahl Vordergrund — $b+d$ = Sequenz-Anzahl Hintergrund

Hierbei ist der Hintergrund eine zweite .fasta-Datei die übergeben wurde, oder die relevante Datei mit geshuffelten Daten. Für den einseitigen Test wird zusätzlich die Summe über die extremeren Fälle berechnet, indem a und d hochgezählt werden, während b und c dekrementiert werden, bis b oder $c = 0$, sodass sich der p-Wert durch die folgende Formel berechnet wird:

$$p = \sum_{i=0}^{\min(b,c)} \frac{\binom{(a+i)+(c-i)}{a+i}\binom{(b-i)+(d+i)}{b-i}}{\binom{a+b+c+d}{a+b}}$$

Da die Überprüfung aller möglichen Wildcards als Motiv ein sehr komplexes Problem ist, werden jetzt heuristisch die besten 100 Motive genommen und generalisiert, indem jeweils eine Stelle durch eine Wildcard (über dem IUPAC-Alphabet) ersetzt wird. Für die resultierenden k-mere wird der Zähler abgeschätzt:

$$|RE_3| \approx |RE_1| + |RE_2| - \frac{|RE_1||RE_2|}{N}$$

$|RE|$ ist die Anzahl der Sequenzen die RE enthalten und N die Anzahl aller Sequenzen. Ist zum Beispiel RE_1=AGA eines der ursprünglichen Motive, wird an jeder der drei Stellen jede mögliche Wildcard (die sich durch einen Buchstaben unterscheiden) eingesetzt. Im ersten Schritt ist also z.B. RE_2=GGA, sodass RE_3=RGA, im zweiten Schritt RE_2=TGA, sodass RE_3=WGA usw. Mit den so geschätzten Zählern wird der p-Wert durch den Fisher-Exakt-Test geschätzt, sodass die Motive durch diesen sortiert werden können. Die Generalisierung inklusive Schätzung des p-Werte wird so häufig wiederholt, bis der p-Wert des besten neuen Motivs nicht mehr kleiner als die gegebene Signifikanzschwelle ist, oder keine Verbesserung festgestellt werden kann. Daraufhin werden die besten 100 generalisierten Motive genommen und exakt gesucht, sodass der korrekte p-Wert errechnet werden kann. Für das beste so gefundene Motiv wird eine Frequenzmatrix erstellt, die

angibt mit welcher Wahrscheinlichkeit welches Nukleotid an welcher Stelle des Motivs auftritt. Um weitere Motive zu finden, wird das zuletzt gefundene im Suchtext mit $'N'$ maskiert und der Algorithmus erneut gestartet.

2.1 Pseudocode

Input: ChIP-Seq-Daten im .fasta-Format
Output: Frequenzmatrix der Top-Motive

$s1 \leftarrow length(sequences)$
$s2 \leftarrow length(background)$
repeat
  ExactSearch()
  FisherExactTest()
  $i \leftarrow 0$
  while $i < 100$ **do**
    Generalize(SortedpValue[i])
    $i \leftarrow i + 1$
  end while
  ExactSearchGeneralize()
  BuildFrequencyMatrix()
  MaskMotif()
until $pValue < 0.05$

Procedure: ExactSearch()
for $len = 3 \rightarrow 8$ **do** ▷ defines length of the k-mere
  loop over all sequences
    loop over all k-mere in sequence
      count k-mer in fore- and background
end for

Procedure: FisherExactTest ()
for all k-mere **do**
  calculate $pValue$ with:

$$p = \sum_{i=0}^{\min(b,c)} \frac{\binom{(a+i)+(c-i)}{a+i}\binom{(b-i)+(d+i)}{b-i}}{\binom{a+b+c+d}{a+b}}$$

end for

Procedure: Generalize()
repeat
  for $Top100kmere$ **do**
  ▷ the 100 k-mere with the lowest $pValue$ in each round

    loop over k-mer
      replace position with each possible wildcard
      estimate counter of the new k-mer($|RE_3|$):

$$|RE_3| \approx |RE_1| + |RE_2| - \frac{|RE_1||RE_2|}{N}$$

  FisherExactTest(estimatedCounter)
  end for
until $pValue < 0.05$

Procedure: ExactSearchGeneralize()
for $Top100kmere$ **do**
  count k-mer exact in fore- and background
end for
FisherhExactTest(exactCounter)

3 RESULTS

3.1 Testing

Alle wesentlichen Funktionen wurde auf ihre Korrektheit mittels der Seqan-Test-Makros überprüft.
Die Tests laufen unabhängig voneinander und liefern für die modularisierten Funktionen die richtigen Ergebnisse.

3.2 Real Data

Nachdem die Korrektheit überprüft worden war, wurde der Algorithmus auf einen reellen Datensatz angewendet. Dabei handelte es sich um 69262 Sequenzen der Länge 300 aus dem GalGal13 Genom, die in drei unabhängigen Chip-Seq-Experimenten als Bindungsstellen des Transkriptionsfaktors Pitx1 identifiziert wurden. Vorhandene Repeats waren mit N maskiert und die detektierten Bindungsstellen wurden im Vorfeld mit einer Bewertung versehen, sodass die Sequenzen absteigend nach ihrer Bewertung sortiert wurden. Beim Testen des Algorithmus auf die k-mer Längen vier bis sechs hatte interessanterweise keins der Motive einen p-Wert < 0.05 (was der Signifikanzschwelle entspricht). Dies könnte daran liegen, dass der Hintergrund dem Vordergrund zu ähnlich ist oder ein Bug besteht, der dem Testen bislang entgangen ist.

3.3 Problems

Nach anfänglichen Schwierigkeiten in Seqan reinzufinden programmierte es sich sehr gut, sodass zu Beginn vor dem Zeitplan gearbeitet wurde und das Programm nach einer Woche soweit war, wie es erst nach $2\frac{1}{2}$ Wochen geplant war. Dieser Zeitvorsprung war jedoch wichtig, da im Plan das Testen vergessen wurde, welches durch das Einarbeiten und Umstrukturierungen durch den benötigten header einiges an Zeit beanspruchte.
Weiterhin gab es ein Missverständnis beim Paper, da die Generalisierung zunächst so implementiert wurde, dass jedes Motiv maximal eine Wildcard enthält. Es jedoch so gemeint war, dass in jedem Schritt maximal eine Wildcard hinzukommt, das Motiv also nur aus Wildcards bestehen kann. Dadurch wurde der Zeitaufwand um einiges größer, weshalb die Laufzeit optimiert werden musste.
Beim Maskieren der Motive stellte sich das Problem auf, dass im nächsten Durchlauf das Programm abbrach und das bis dahin beste Motiv $N's$ enthielt, was nicht möglich sein sollte. Das Problem bestand darin, dass während des Durchlaufs des Finders die Maskierung stattfand, nach jeder Änderung am Text in dem gesucht wird jedoch der Index neu aufgebaut werden muss. Statt der Maskierung

während der Count-Funktion werden jetzt alle relevanten Positionen in einem String gespeichert und im Nachhinein ersetzt.
Durch die große Anzahl an Parametern, sowie Funktionen, wurde der Code zeitweise unübersichtlich, was zur Umstrukturierungen führte, indem ein Großteil der Parameter in einem struct ausgelagert wurde.

4 CONCLUSION

Zur Code-Optimierung und übersichtlicheren Gestaltung könnten Templates verwendet werden, da so einige Redundanzen wegfallen. Weiterhin kann die Maskierung optimiert werden, da momentan die relevanten Stellen in einem String gespeichert werden und nach der Ersetzung durch N der Index neu aufgebaut werden muss. Alternativ könnten die in Seqan enthaltenen IntervallTrees benutzt werden, sodass die aufwendige Neuberechnung der Indices nicht benötigt wird. Die initiale exakte Suche der k-mere könnte eventuell dadurch beschleunigt werden, dass anstatt der Schleife über all$k$-merer Längen nur ein Durchgang stattfindet und über ein Array, welches über den Text geshiftet wird in einem Durchlauf alle Teilsequenzen gefiltert werden können.

Zur Verbesserung des Algorithmus könnten die Eingabedaten nach der Bewertung gewichtet werden, da Peaks mit einer geringen Bewertung wahrscheinlich kein häufig vorkommendes Motiv enthalten. Zudem könnte (auf Kosten der Laufzeit) eventuell eine Verbesserung herbeigeführt werden, wenn im Generalisierungs-Schritt der Zähler teilweise exakt bestimmt wird, sodass die Verteilung unter den besten 100 Motiven anders aussehen könnte.

REFERENCES

Bailey, T. L. (2011). DREME: motif discovery in transcription factor ChIP-seq data. *Bioinformatics*, **27**(12), 1653–1659.

Liu, E. T., Pott, S., and Huss, M. (2010). QA: ChIP-seq technologies and the study of gene regulation. *BMC Biol.*, **8**, 56.