Genomorientierte Bioinformatik

-

Differential Analysis

Malte Weyrich

9. Januar 2025

Alternatives Spleißing ist ein fundamentaler zellulärer Mechanismus mit großen Einfluss auf regulatorische Prozesse und Genprodukte. Dabei werden die *Exons* eines Transkripts teilweise übersprungen oder neu angeordnet. Verantwortlich für diesen Prozess ist das *Spleißosom*. Im Folgenden wird ein Programm zur Quantifizierung von *Skipped Exon Events* vorgestellt und dessen Ergebnisse mit einem Hypothesen Test evaluiert. Es wurde auf zehn *bam*-Dateien ausgeführt mit *annotation_b37.gtf* als Referenzgenom. Zusätzlich werden die Resultate mit einem bereist publizierten Tool (***DEXSeq*** aus Anders, Reyes, and Huber 2012) verglichen.

Inhalt

1 — Berechnung der Percent Spliced-In Werte

1.1. Definition

Percent Spliced-In (PSI) Werte werden in der Bioinformatik genutzt, um die Evidenz von *Skipped Exon Events* anhand von beobachteten Daten darzustellen. Die *Skipped Exon Events* entstehen durch *Alternatives Spleißen* und beeinflussen maßgeblich das Proteinendprodukt eines *Gens*. Die Expression von Transkripten kann mittels *RNAseq* untersucht werden, bei dem die Sequenzen der vorliegenden Transkripte sequenziert werden. Somit entstehen Milliarden von *Reads*, welche mit der Hilfe von *Software* zurück auf das *Referenz Genom* projiziert werden können. Anschließend werden die *Reads* annotiert, das heißt es wird geschaut, von welchen Transkripten die *Reads* stammen. Der *PSI* Wert wird dann mittels *Inclusion Read Counts (**IRC**)* und *Exclusion Read Counts (**ERC**)* berechnet werden:

$$PSI := \frac{IRC}{IRC + ERC}.$$

Sei G ein *Gen* mit Transkripten WT und SV. Zudem sei se eine, durch die Annotation implizierte, übersprungene Region in WT, für die der *PSI* Wert berechnet werden soll. Wenn R die Menge an aligniernten *Read Pairs* auf G ist, dann ist $I \subseteq R$ die Menge an alignierten *Read Pairs* auf se und somit

$$IRC_{se} := |I| .$$

Die *Exclusion Reads* $E \subseteq R$ sind genau die *Read Pairs*, welche nicht auf se *gemapped* werden können.

$$ERC_{se} := |E| .$$

Ein *PSI* Wert von 1 würde beispielsweise bedeuten, dass die Region se in den beobachteten Transkripten eines *Gens* überwiegend inkludiert wurde (also nicht durch das *Spleißosom* heraus gespließt wurde), während ein Wert von 0 auf ein vermehrtes Ausschließen von se hindeuten würde.

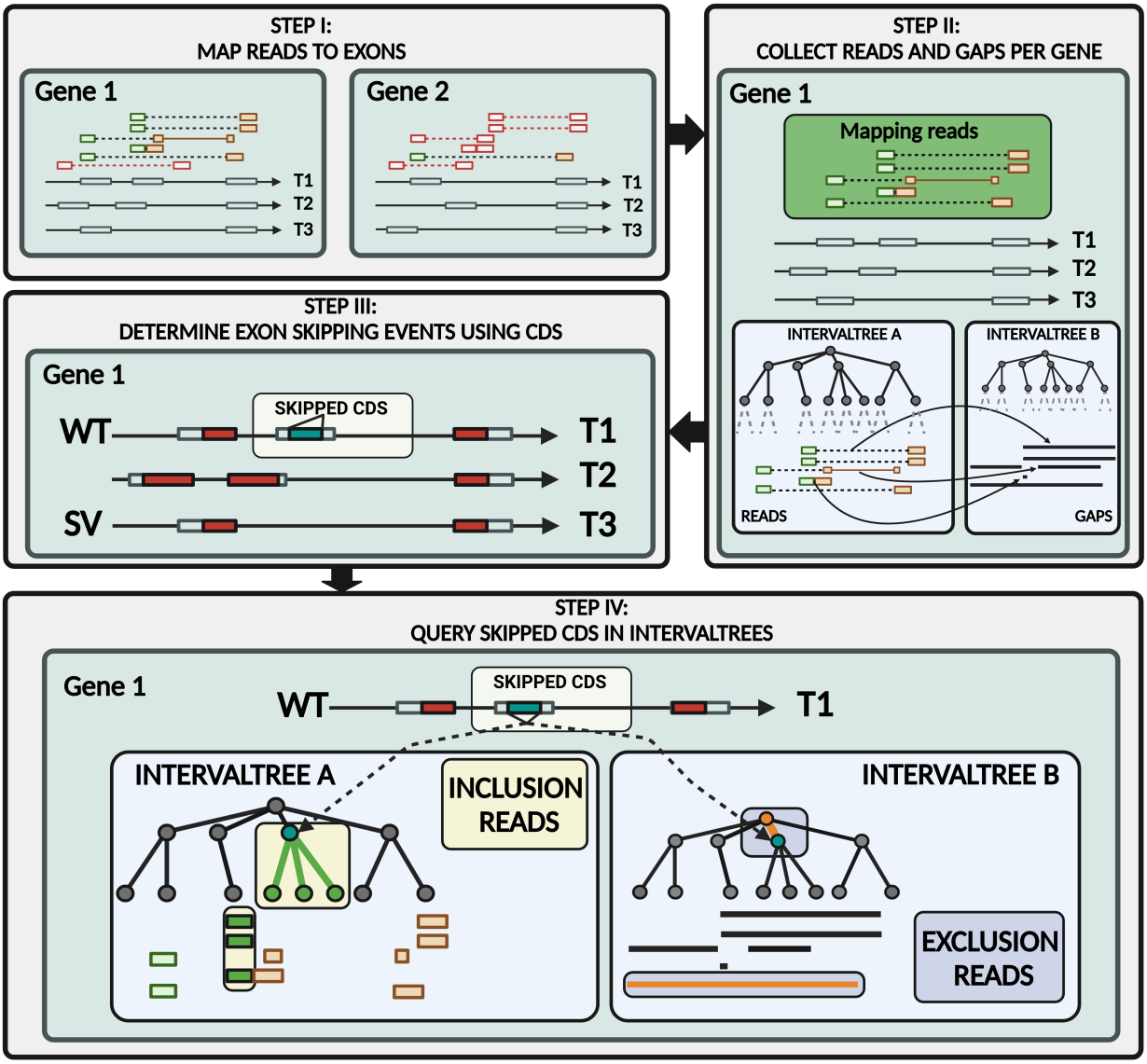

Abbildung 1 — Programmlogik zur Berechnung der *IRC* und *ERC counts* pro *Skipped Exon Event*. In dem Beispiel hätte das übersprungene Exon von G1 einen *PSI* Wert von 0.75. Die Abbildung wurde mit BioRender 2024 erstellt.

Der *JAR* werden eine *bam*- und ein *GTF*-Datei übergeben. Zudem muss eine Ausgabedatei Spezifiziert werden:

```
java -jar psi.jar -bam <bam>
                  -gtf <gtf>
                  -o <ausgabe.psi>
```

Die *JAR* liest zunächst die *GTF*-Datei ein und speichert alle *Gene* und deren annotierte *Transkripte* ab. Dabei enthält jedes *Transkript* jeweils dessen *Exons* **und** *Coding DNA Sequences (CDS)*. Gleichzeitig wird die *bam*-Datei mittels der *samtools library* decodiert und abgespeichert. Nach der Einleseroutine kann das Programm in vier Schritte eingeteilt werden (siehe Abbildung 1):

I. MAP READS TO EXONS

Die *Read Pair* Koordinaten werden mit den ***Exon*** Koordinaten der Transkripte abgeglichen. Sobald mindestens ein *Read Pair* auf ein *Gen mapped*, wird das *Gen* in einer *ArrayList<Gene>* ***mappedGenes*** abgelegt. Die Kriterien für einen validen *Read* wurden bereits im letzten Report geklärt.

II. COLLECT READS AND GAPS PER GENE

Für jedes der Gene in ***mappedGenes*** werden nun zwei Intervalbäume A, B erstellt. Baum A beinhaltet alle *AlignmentBlocks* der zum Gen zugeteilten *Read Pairs*, während Baum B die Lücken zwischen einzelnen *AlignmentBlocks* und zwischen den *Forward* und *Reverse Reads* abspeichert.

III. DETERMINE EXON SKIPPING EVENTS USING CDS

Im nächsten Schritt wird über alle Gene aus ***mappedGenes*** iteriert. Für jedes einzelne Gen werden alle *Skipped Exon Events* mittels der *CDS* des Gens berechnet. Dabei wird dieselbe Logik wie in Report 1 verwendet.

IV. QUERY SKIPPED CDS IN INTERVALTREE

Für jede *CDS* werden nun dessen Intervalle in den Bäumen A, B abgefragt und die *Read IDs* gezählt.

$$IRC := |\{A.\textbf{\textit{getIntervalsSpannedBy}}(CDS.getStart(), CDS.getStop())\}|$$

$$ERC := |\{B.\textbf{\textit{getIntervalsSpanning}}(CDS.getStart(), CDS.getStop())\}|$$

Dabei werden für IRC **Intervalle** in A betrachtet, **die von der *CDS* überspannt werden** ($\equiv$ "*Reads* die innerhalb der *CDS* liegen") und für ERC werden **Intervalle** in B gesucht, **die die *CDS* überspannen** ($\equiv$ "Intervalle, die die *CDS* beinhalten").

Die *PSI* Werte werden in die Ausgabedatei mit folgendem Format geschrieben:

```
Gene ID            cdsStart-cdsStop    IRC   ERC  Total  PSI
ENSG00000165623.5  13275735-13275801   25    13   38     0.65
...                ...                 ...   ...  ...    ...
```

2 — Hypothesen Test

Wir wollen untersuchen, ob der Prozess des *Alternativen Spleißens* anhand von *Exon Skipping Events* ($\mathbf{\Psi}$) rein zufällig stattfindet, oder von regulatorischen Mechanismen gesteuert wird.

- $\mathbf{H_0}$:= *Exon Skipping Events* finden zufällig statt.
- $\mathbf{H_1}$:= *Exon Skipping Events* sind nicht reiner Zufall.

Dafür nehmen wir an, dass die Anzahl an *IRC* Binomial Verteilt ist:

$$P(i, N|p) = \binom{N}{i} p^i (1-p)^{N-i}.$$

Hierbei beschreibt p die Wahrscheinlichkeit, dass ein einzelnes *Exon* als *Inclusion Read* klassifiziert wird, während N die Anzahl an *Reads* darstellt, die für das momentane *Exon* einen Informationsgehalt haben. Zudem gehen wir davon aus, dass jedes Transkript mindestens einen informativen *Read* pro Exon hat.

Für den Test wurden die zehn Ausgabedateien der *JAR* in zwei Gruppen unterteilt (siehe Tabelle 1):

Tabelle 1 — Einteilung der Samples in zwei Gruppen

Ausgabedatei i	1	2	3	4	5	6	7	8	9	10
Gruppe g_i	1	1	1	1	1	2	2	2	2	2
IRC	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}
Gesamt Anzahl	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9	N_{10}
$\mathbf{\Psi_{reduced}} \sim (p_0)$	p_0	p_0	p_0	p_0	p_0	p_0	p_0	p_0	p_0	p_0
$\mathbf{\Psi_{full}} \sim (p_1, p_2)$	p_1	p_1	p_1	p_1	p_1	p_2	p_2	p_2	p_2	p_2

Dabei simulieren wir zwei *Modellen*:

- $\mathbf{\Psi_{reduced}} \sim (p_0)$: Hat einen einzigen Parameter p_0 für alle Gruppen.
- $\mathbf{\Psi_{full}} \sim (p_1, p_2)$: Besitzt zwei verschiedenen Parameter.

Die *Likelihood* kann nun mit der *Maximum Likelihood Estimation* Methode abgeschätzt werden.

Die *Likelihood Funktionen* lasses sich wie folgt definieren:

- $\mathbf{L_{reduced}}(\mathbf{p_0}) := \prod_{j=1}^{10} P(i_j, N_j | p_0)$
- $\mathbf{L_{full}}(\mathbf{p_1}, \mathbf{p_2}) := \prod_{j=1}^{10} P(i_j, N_j | p_{g_j})$

3 — Vergleich mit DEXSeq

A — Appendix Section

hm

Text goes here