

# Genomorientierte Bioinformatik

-

## Differential Analysis

Malte Weyrich

12. 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*. Die Identifizierung von *differentiellen Spleiß-Events*, also Unterschieden im *Spleißverhalten* zwischen verschiedenen Bedingungen, ist entscheidend für das Verständnis von Genregulation und Krankheitsmechanismen. 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 bereits publizierten Tool (*DEXSeq* aus Anders, Reyes, and Huber [2012](#)) verglichen.

---

# Inhalt

<b>1</b>	<b>Berechnung der Percent Spliced-In Werte</b>	<b>3</b>
1.1	Definition . . . . .	3
1.2	Programm Logik . . . . .	4
<b>2</b>	<b>Hypothesen Test</b>	<b>6</b>
<b>3</b>	<b>Komplexität und Korrektheit</b>	<b>8</b>
3.1	Komplexität . . . . .	8
3.2	Korrektheit . . . . .	10
<b>4</b>	<b>Ergebnisse</b>	<b>11</b>
4.1	<i>Read</i> Abdeckung . . . . .	11
4.2	PSI Werte . . . . .	12
4.3	Vergleich mit DEXSeq . . . . .	14
4.4	Laufzeit . . . . .	15

## 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 alignierten *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 gespleißt wurde), während ein Wert von 0 auf ein vermehrtes Ausschließen von  $se$  hindeuten würde.

## 1.2. Programm Logik

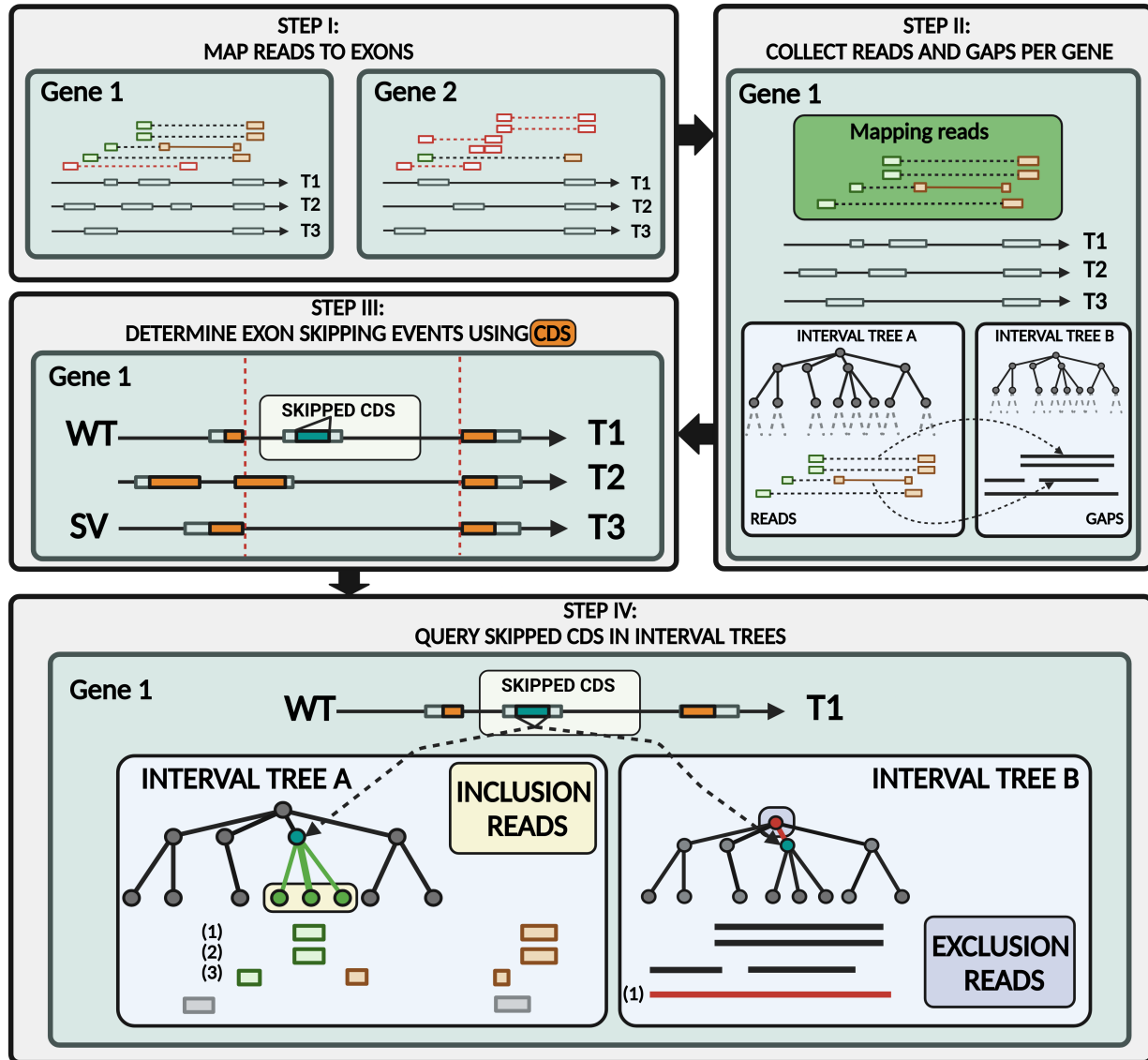


Abbildung 1 – Programmlogik zur Berechnung der *IRC* und *ERC counts* pro *Skipped Exon Event*. In dem Beispiel hätte das übersprungene Exon von G1 einen *PSI* Wert von 0.75. Die Abbildung wurde mit [BioRender 2024](#) erstellt.

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. Es werden nur *Reads* in *mappedGenes* hinzugefügt, wenn sie auf genau ein einziges *Transkript* passen.

## II. COLLECT READS AND GAPS PER GENE

Für jedes der *Gene* in *mappedGenes* werden nun zwei Intervallbäume *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.

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

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

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* ( $\Psi$ ) rein zufällig stattfindet, oder von regulatorischen Mechanismen gesteuert wird.

- $H_0$  := *Exon Skipping Events* finden zufällig statt.
- $H_1$  := *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	$N_1$	$N_2$	$N_3$	$N_4$	$N_5$	$N_6$	$N_7$	$N_8$	$N_9$	$N_{10}$
$\Psi_{\text{reduced}} \sim (p_0)$	$p_0$	$p_0$	$p_0$	$p_0$	$p_0$	$p_0$	$p_0$	$p_0$	$p_0$	$p_0$
$\Psi_{\text{full}} \sim (p_1, p_2)$	$p_1$	$p_1$	$p_1$	$p_1$	$p_1$	$p_2$	$p_2$	$p_2$	$p_2$	$p_2$

Dabei simulieren wir zwei *Modellen*:

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

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

Die *Log-Likelihood Funktionen* lassen sich wie folgt definieren:

- $\mathcal{L}_{\text{reduced}}(p_0) := \log \left( \prod_{j=1}^{10} P(i_j, N_j | p_0) \right)$
- $\mathcal{L}_{\text{full}}(p_1, p_2) := \log \left( \prod_{j=1}^{10} P(i_j, N_j | p_{g_j}) \right)$

Indem man die jeweilige Ableitung von  $\mathcal{L}$  gleich null setzt, erhält man die *MLE* Schätzer  $\hat{p}_0$  und  $\hat{p}_1, \hat{p}_2$ :

$$\begin{aligned} \mathcal{L}(\hat{p}_0) &= \log \left( \prod_{j=1}^{10} \binom{N}{i_j} p_0^{i_j} (1 - p_0)^{N_j - i_j} \right) \\ \mathcal{L}(\hat{p}_0) &= \sum_{j=1}^{10} \log \left( \binom{N_j}{i_j} \right) + \log(p_0) \sum_{j=1}^{10} i_j + \log(1 - p_0) \sum_{j=1}^{10} (N_j - i_j) \\ \mathcal{L}'(\hat{p}_0) &= \frac{\sum_{j=1}^{10} i_j}{p_0} - \frac{\sum_{j=1}^{10} (N_j - i_j)}{1 - p_0} \\ 0 &= \frac{\sum_{j=1}^{10} i_j}{p_0} - \frac{\sum_{j=1}^{10} (N_j - i_j)}{1 - p_0} \\ \hat{p}_0 &= \frac{\sum_{j=1}^{10} i_j}{\sum_{j=1}^{10} N_j} \end{aligned}$$

$$\mathcal{L}(\hat{p}_1, \hat{p}_2) = \log \left( \prod_{j=1}^{10} \binom{N}{i_j} p_{g_i}^{i_j} (1 - p_{g_i})^{N_j - i_j} \right)$$

$$\iff \mathcal{L}(\hat{p}_1, \hat{p}_2) = \mathcal{L}_{\text{Gruppe}_1}(p_1) + \mathcal{L}_{\text{Gruppe}_2}(p_2)$$

$$\mathcal{L}_{\text{Gruppe}_1}'(\hat{p}_1) = \frac{\sum_{j \in \text{Gruppe}_1} i_j}{p_1} - \frac{\sum_{j \in \text{Gruppe}_1} (N_j - i_j)}{1 - p_1}$$

$$\iff \hat{p}_1 = \frac{\sum_{j \in \text{Gruppe}_1} i_j}{\sum_{j \in \text{Gruppe}_1} N_j}$$

$$\mathcal{L}_{\text{Gruppe}_2}'(\hat{p}_2) = \dots$$

$$\hat{p}_2 = \frac{\sum_{j \in \text{Gruppe}_2} i_j}{\sum_{j \in \text{Gruppe}_2} N_j}$$

Die *Likelihood Schätzer* können nun verwendet werden, um  $\mathcal{L}_{\text{reduced}}(\hat{p}_0)$  und  $\mathcal{L}_{\text{full}}(\hat{p}_1, \hat{p}_2)$  zu berechnen. Zudem bestimmen wir jeweils die *Likelihood Ratio Statistik (LRS)*:

$$LRS := -2 \log \left( \frac{\mathcal{L}_{\text{reduced}}}{\mathcal{L}_{\text{full}}} \right).$$

Dabei gilt:

$$LRS \sim \chi^2.$$

Somit können wir für jeden der berechneten *PSI* Werte einen *p* Wert berechnen. Zudem wurde die *Benjamini Hochburg* Methode (Benjamini and Hochberg 1995) verwendet, um die *p* Werte anzupassen, da multiple Hypothesen Tests durchgeführt wurden.

Das *R* Skript *LRS.R* implementiert die Berechnung der *LRS* und *p* Werte und schreibt die Resultate in eine Ausgabedatei:

gene	exon	p0	p1	p2	llreduced	llfull	lrs	pvalue	padj
id	start-stop	0.64	0.64	0.65	-25.86	-25.82	0.07	0.78	0.9
...	...	...	...	...	...	...	...	...	...

### 3 – Komplexität und Korrektheit

#### 3.1. Komplexität

Das entwickelte Java Programm benutzt für das Einlesen der *GTF*-Datei, dem Bestimmen der *Exon Skipping Events* und dem *Mapping* der *Reads* dieselbe Komplexität wie die Programme aus Report 1 und 3. Das einzige was an zusätzlichem Aufwand anfällt ist die Erstellung der Intervallbäume pro Gen (A) und das anschließende Zählen der *IRC* und *ERC* (B).

#### (A) Erstellen der Intervallbäume:

Sei  $\tilde{G}$  die Menge aller *Gene* aus der vorliegenden *Annotation* und  $\tilde{R}$  die Menge aller *Reads* aus dem Sequenzierexperiment. Dann ist  $G \subseteq \tilde{G}$  die Teilmenge an *Genen*, auf welche es ein *Mapping* von *Reads* gibt. Jedes *Gen*  $g_i \in G$  hat also eine Teilmenge  $R_i$  an alignierten *Reads* aus  $\tilde{R}$ . Für jedes *Gen*  $g_i$  werden alle zugehörigen *Reads*  $R_g$  in die Intervallbäume *A* und *B* eingefügt. Das Einfügen in einen Intervallbaum hat ein logarithmisches Kostenmaß:  $\mathcal{O}(\log n)$ ,



wobei  $n$  die Gesamtanzahl an Intervallen ist, welche sich bereits in dem Baum befinden. Für das konsequente Einfügen von *Reads* aus  $R_g$  in  $A$  ergibt sich folgende, von oben abgeschätzte Laufzeit:

$$\begin{aligned} 1 + \log(1) + \log(2) + \dots + \log(|R_g| + k) &< \sum_{i=1}^{|R_g|+k} \log(|R_g| + k) \\ &\in \mathcal{O}\left((|R_g| + k) \cdot \log(|R_g| + k)\right) \end{aligned}$$

Bei  $k$  handelt es sich um eine Konstante, welche die Anzahl an Fällen beschreibt, wo ein einzelner *Read* aus  $R_g$  aus mehreren *AlignmentBlocks* besteht.

Das gleiche gilt für die *Gaps*, welche in den zweiten Intervallbaum eingefügt werden:

$$\begin{aligned} 1 + \log(1) + \log(2) + \dots + \log(|R_g|/2 + c) &< \sum_{i=1}^{|R_g|/2+c} \log(|R_g|/2 + c) \\ &\in \mathcal{O}\left((|R_g|/2 + c) \cdot \log(|R_g|/2 + c)\right) \\ &\in \mathcal{O}\left((|R_g| + c) \cdot \log(|R_g| + c)\right) \end{aligned}$$

Da es sich um *paired end* Daten handelt (Abbildung 1), gibt es zwischen nicht überlappenden *Read Pairs* einen *Gap* (also insgesamt  $\leq |R_g|/2$ ) und zusätzlich  $c$  viele *Gaps* zwischen den jeweiligen *AlignmentBlocks* der *Reads* (**hierbei ist  $c == k$ , da durch jeden zusätzlichen *AlignmentBlock* in einem *Read* auch ein zusätzlicher *Gap* im *Read* existiert**).

Da wir insgesamt  $|G|$  viele Gene haben also:

$$\begin{aligned} &\mathcal{O}\left(|G| \cdot \left((|R_{\tilde{g}}| + k) \cdot \log(|R_{\tilde{g}}| + k) + (|R_{\tilde{g}}| + c) \cdot \log(|R_{\tilde{g}}| + c)\right)\right) \\ &\in \mathcal{O}\left(|G| \cdot \left(2 \cdot (|R_{\tilde{g}}| + k) \cdot \log(|R_{\tilde{g}}| + k)\right)\right) \\ &\in \mathcal{O}\left(|G| \cdot \left((|R_{\tilde{g}}| + k) \cdot \log(|R_{\tilde{g}}| + k)\right)\right) \end{aligned}$$

, wobei

$$\tilde{g} := \underset{\substack{R_g \in R, \\ R_g \neq R_{\tilde{g}}}}{\forall} R_g \leq R_{\tilde{g}}$$

das Gen mit den meisten alignierten *Reads* ist.

**(B) Ermittlung der *IRC* und *ERC*:**

Eine Suchoperation in einem Intervallbaum hat ebenfalls eine Komplexität von  $\mathcal{O}(\log n)$ . Nach **Schritt (A)** befinden sich in den Bäumen  $A, B$  von einem Gen  $g \in G$  genau  $|R_i|$  und  $|R_i|/2 + c$  viele *Reads*. Jedes  $g \in G$  besitzt  $e_g$  viele *Exon Skipping Events*. So muss  $|e_g|$  oft in jeweils beiden Bäumen eine Suche getätigt werden. Pro Gen also:

$$|e_g| \cdot \log(R_{\tilde{g}}) + |e_g| \cdot \log(R_{\tilde{g}}/2 + c) = |e_g| \cdot (\log(R_{\tilde{g}}) + \log(R_{\tilde{g}} + c)) \in \mathcal{O}\left(|e_g| \cdot \log(R_{\tilde{g}} + c)\right).$$

### 3.2. Korrektheit

Der Ansatz mit den Intervallbäumen ist Korrekt. In den Intervallbäumen werden nur *Reads* abgespeichert, die genau auf ein *Transkript* passen. Somit werden schon mal alle *Reads* ohne eindeutigen Informationsgehalt ignoriert. Zusätzlich werden zu jeder hinzugefügten Region (in beiden Bäumen), die *Ursprungs ID* (in diesem Fall die *Transkript ID*) und die *Read ID* gespeichert. Da der Intervallbaum korrekt implementiert ist, liefert diese entweder alle Intervalle, die von der übersprungenen *CDS* beinhaltet werden (Baum  $A$ ), oder alle *Gaps*, welche die *CDS* überspannen (Baum  $B$ ). Die Anzahl an einzigartigen *Read IDs* beider Abfragen ergibt dann die jeweiligen *IRC* und *ERC*. Der einzige Sonderfall ergibt sich, wenn eine Region  $r$  aus dem Baum  $B$  dieselbe *Transkript ID* hat wie das der übersprungenen *CDS*. In diesem Fall impliziert  $r$  dann keinen *ERC* sonder einen *IRC*, denn  $r$  wurde schließlich eindeutig auf das *Transkript* der *CDS* aligniert.

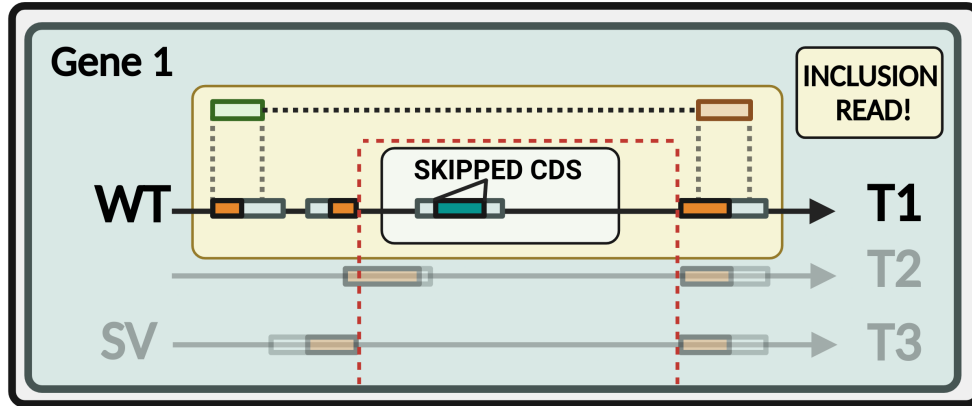


Abbildung 2 – Sonderfall bei der Berechnung von *ERC*. Der *Read* passt hier nur auf T1 und obwohl sich die übersprungene *CDS* in der Lücke zwischen *Forward* und *Reverse Read* befindet, zählt dieser *Read* als *IRC*, denn er kann nur durch T1 entstanden sein. Die Abbildung wurde mit [BioRender 2024](#) erstellt

## 4 – Ergebnisse

### 4.1. Read Abdeckung

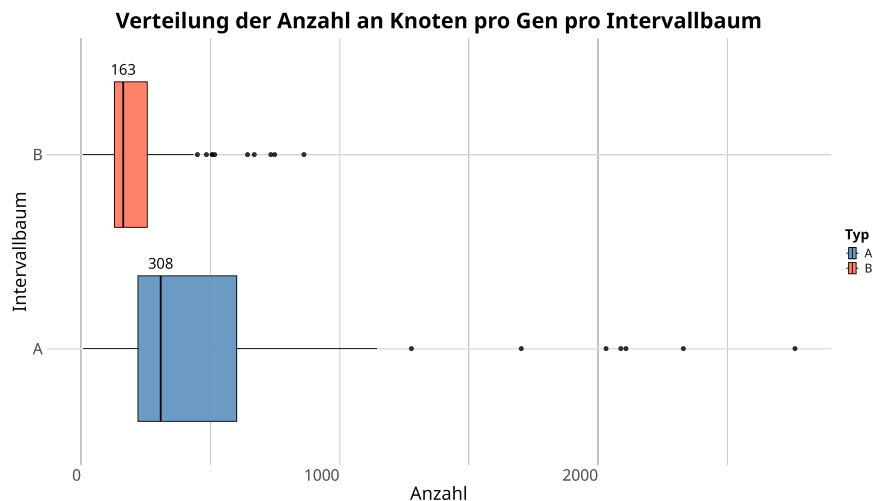
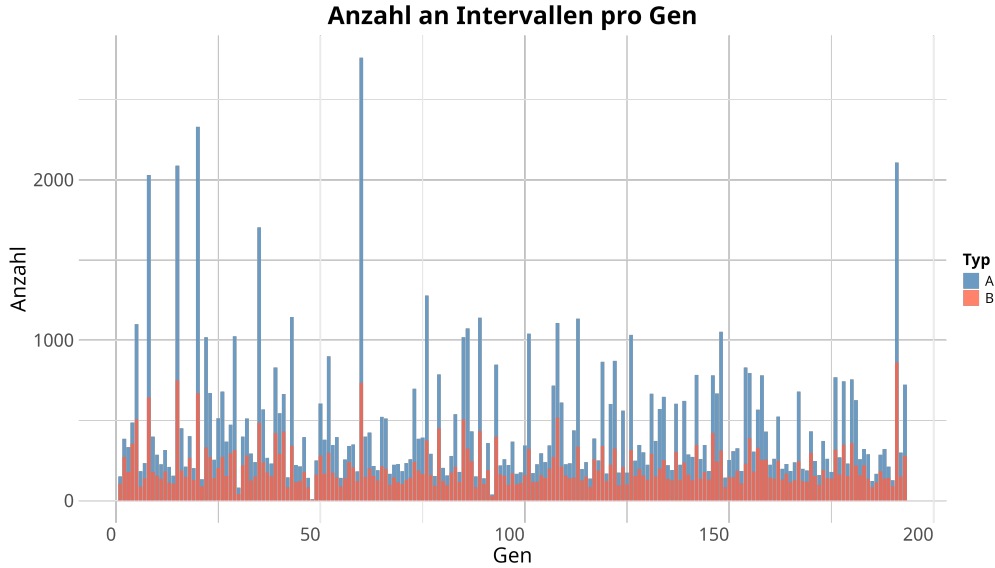


Abbildung 3 – Verteilung der Anzahl von abgespeicherten Intervalle der Bäume *A*, *B* pro Gen. Dabei beinhaltet Typ *A* jeweils alle abdeckenden *AlignmentBlocks/Reads* für ein Gen und *B* die *Gaps* zwischen *AlignmentBlocks* eines *Reads* und zwischen *Forward* und *Reverse Read*.

Wie bereits in Abschnitt 3.1 beschrieben, sollte die Anzahl an Intervallen pro Gen in Baum *B* nicht größer als in *A* sein. Die Verteilungen in Abbildung 3 lassen dies bereits vermuten, da die Verteilung der Werte für *B* ungefähr um den Faktor 0.5 nach links verschoben ist. Ebenfalls



**Abbildung 4 – Anzahl an Intervallen pro Gen pro Intervallbaum.**

zusehen ist, dass die Hälfte aller Gene mindesten 308 alignierte *AlignmentBlocks*.

In Abbildung 4 ist es deutlicher zu erkennen, dass die Mengen  $|R_g|$  in Bäumen *A* immer größer ist als die Anzahl an *Gaps* in *B*. Ebenfalls zu sehen ist, dass es einige Gene mit vielen überlappenden *Forward* und *Reverse Reads* gibt. Dies ist der Fall wenn Baum *A* mehr als doppelt so viele Intervalle beinhaltet als Baum *B*.

#### 4.2. PSI Werte

In Abbildung 5 wurden die korrigierten, logarithmierten P-Werte nach chromosomaler Position aufgetragen. Der Großteil an signifikanten Werten liegt in den Chromosomen 1-9, was darauf hindeuten könnte, dass die *Exon Inklusion* innerhalb dieser Regionen durch regulatorische Elemente wie z.B. *Promotoren*, *Enhancern* oder *Silencer* gesteuert wird. Der Trend könnte ebenfalls implizieren, dass die Gene auf den ersten neun Chromosomen anfälliger für Alternatives Spleißen sind, da sie beispielsweise für wichtige Spleißvarianten codieren.

Abbildung 6 zeigt die Verteilung der Wahrscheinlichkeiten  $p_0$ ,  $p_1$  und  $p_2$ , welche jeweils mit den *Likelihood Schätzern* pro Event errechnet worden sind. Hier ist bereits ein Unterschied zwischen dem  $\Psi_{\text{reduced}} \sim (p_0)$  und dem  $\Psi_{\text{full}} \sim (p_1, p_2)$  zu sehen, da die  $p_0$ -Verteilung eine einzelne, konzentrierte Wahrscheinlichkeitsverteilung zeigt, während das vollständige Modell zwei separate Verteilungen ( $p_1$  und  $p_2$ ) mit unterschiedlichen Maxima aufweist.

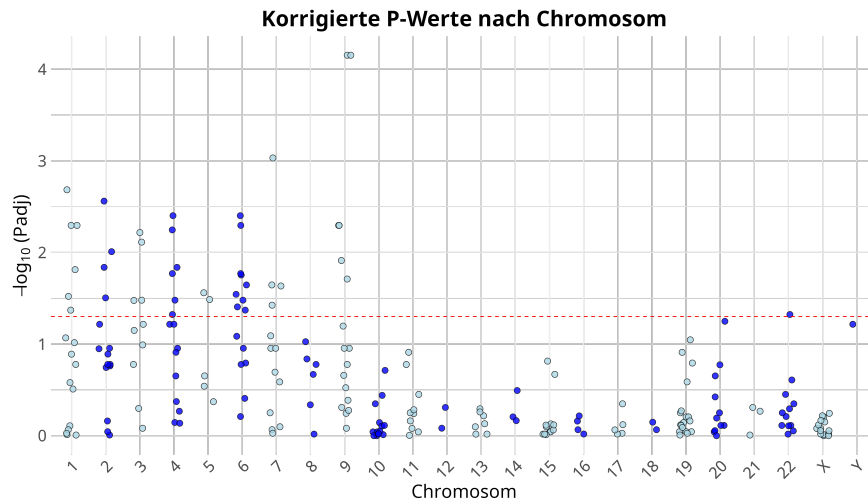


Abbildung 5 – Angepasste P-Werte nach Chromosomen. Die Rote Linie kennzeichnet das transformierte Signifikanzniveau von  $\alpha = 0.05$ . Alle Punkte über der Indikatorlinie sind als signifikant zu interpretieren. Insgesamt wurden 232 *Exon Skipping Events* entdeckt.

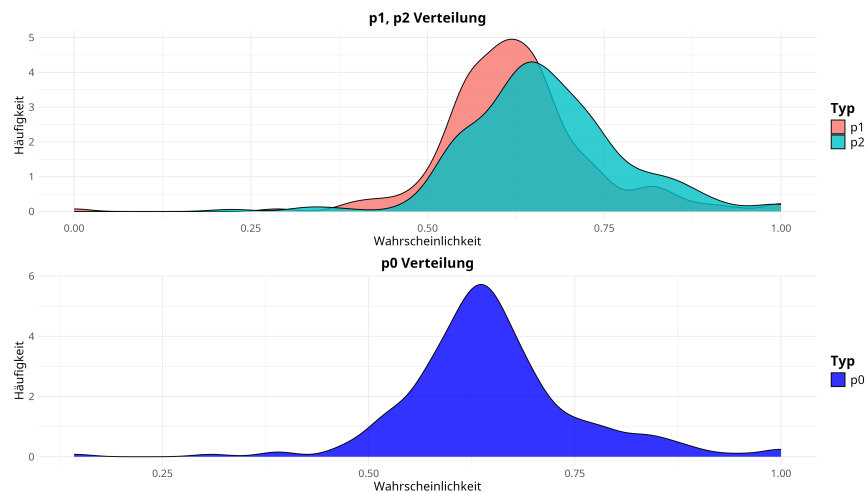
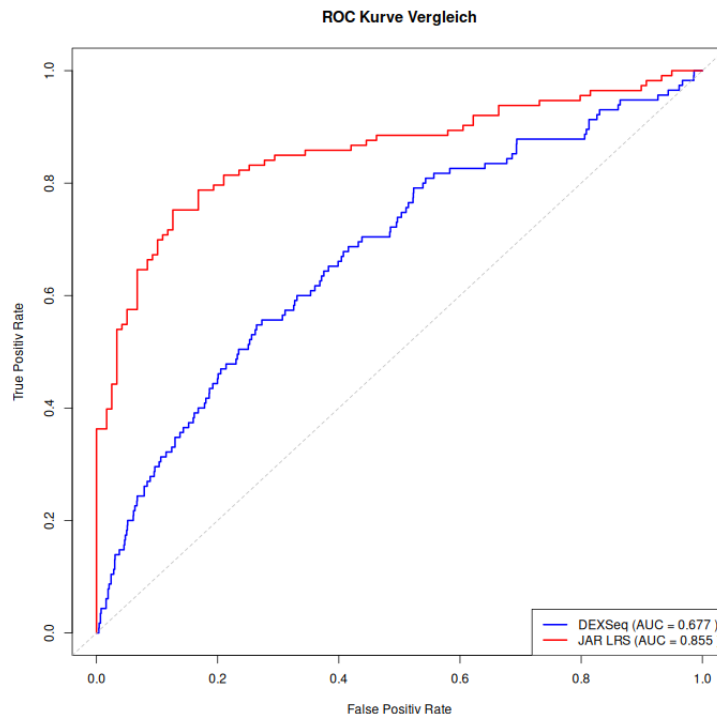


Abbildung 6 – Verteilung der  $p_0$ ,  $p_1$  und  $p_2$  Werte für das  $\Psi_{\text{reduced}}$  und  $\Psi_{\text{full}}$  Modell.

### 4.3. Vergleich mit DEXSeq

Um unsere Ergebnisse mit *DEXSeq* zu vergleichen, haben wir die selben zehn *bam*-Dateien für das Tool aufbereitet und als Eingabe verwendet. Die resultierenden Exons wurden zurück auf die von uns verwendete Annotationsdatei projiziert, um sie mit den Ergebnissen unseres Modells vergleichen zu können. Für beide Tools wurde in Abbildung 7 die "Receiver Operating Characteristics" (ROC) Kurve aufgetragen.

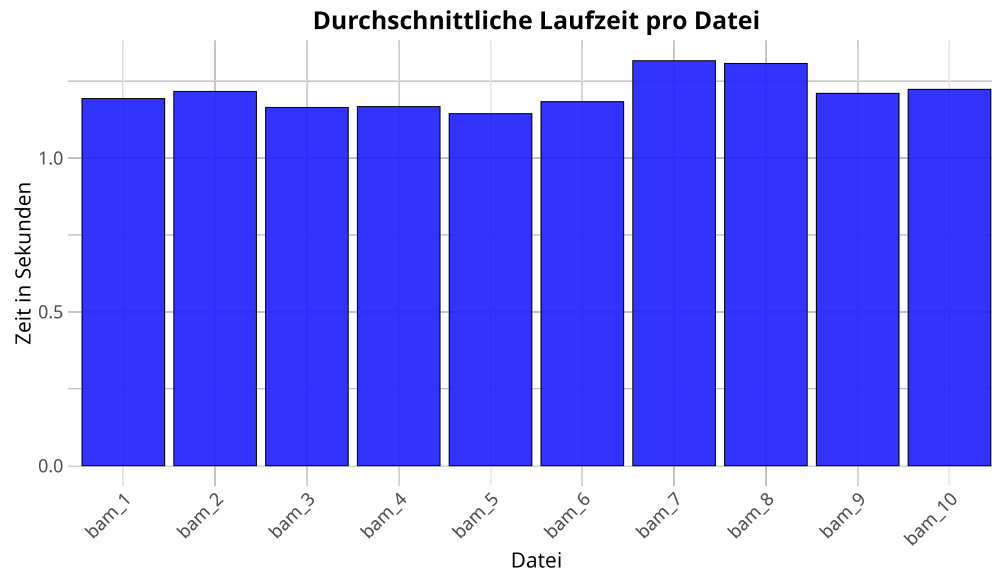


**Abbildung 7 – ROC Kurven Vergleich mit *DEXseq*.**

Die *ROC-Kurven* zeigen einen deutlichen Leistungsunterschied zwischen unserem *JAR LRS* Modell und *DEXSeq*. Unser Modell erreicht einen *AUC*-Wert von 0.855, während *DEXSeq* einen niedrigeren *AUC*-Wert von 0.677 aufweist (kaum besser als ein Zufälliges Modell mit  $AUC = 0.5$ ). Dies deutet auf eine überlegene Diskriminierungsfähigkeit des *JAR LRS* Modells hin. Der steilere Anstieg der roten Kurve im linken Bereich des Diagramm bedeutet, dass unser Modell eine höhere *True Positive Rate* bei einer niedrigeren *False Positive Rate* erreicht - eine wichtige Eigenschaft für die praktische Anwendung, da sie auf eine bessere Fähigkeit hinweist, tatsächliche differentielle Spleiß-Events zu erkennen, während gleichzeitig weniger falsch-positive Ergebnisse produziert

werden.

#### 4.4. Laufzeit



**Abbildung 8 – Durchschnittliche Laufzeit der JAR auf allen Dateien nach zehnfacher Ausführung auf AMD Ryzen 7 PRO 4750U with Radeon Graphics (16) @ 1.700GHz.**

Die Laufzeiten in Abbildung 8 sind alle knapp über einer Sekunde. Das liegt aber vor allem daran, dass die *GTF*-Datei für diese Aufgabe manuell überarbeitet wurde, sodass jedes Gen höchstens zwei Transkripte besitzt. Dadurch ist die gekürzte Version nur *24MB* groß. Das Einlesen der *GTF*- und das Prozessieren der *bam*-Dateien nimmt am meisten Zeit in Anspruch: *323ms* für das Einlesen der *Annotation* und *800ms* für das *Mapping* einer *bam*-Datei auf die Gene in der *GTF*. Die Methode ***getPctSplicedCounts*** ist zuständig für die:

- Identifizierung von *Exon Skipping Events*
- Berechnung der *PSI* Werte

Sie benötigt lediglich *68ms*, da es insgesamt nur 199 Gene in *mappedGenes* gab und nur 232 *Exon Skipping Events* durch die Annotationsdatei impliziert werden.

## Referenzen

**Anders, Simon, Alejandro Reyes, and Wolfgang Huber.** 2012. “Detecting differential usage of exons from RNA-seq data.” *Genome Research* 22:4025. <https://doi.org/10.1101/gr.133744.111>. (Cited on page 1).

**Benjamini, Yoav, and Yosef Hochberg.** 1995. “Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing.” *Journal of the Royal Statistical Society: Series B (Methodological)* 57 (1): 289–300. <https://doi.org/https://doi.org/10.1111/j.2517-6161.1995.tb02031.x>. eprint: <https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.2517-6161.1995.tb02031.x>. <https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1995.tb02031.x>. (Cited on page 8).

**BioRender.** 2024. *BioRender - Biological Figure Creation Tool*. <https://BioRender.com>. Accessed: 2024-12-09. (Cited on pages 4, 11).