



Software unterstützte Analyse von
regulatorischen Elementen in Promotoren
mittels AIModules

Dissertation zur Erlangung des naturwissenschaftlichen
Doktorgrades der Julius-Maximilians-Universität Würzburg

vorgelegt von
Muharrem Aydinli
geboren in Marktredwitz

Würzburg, 2021

Eingereicht am:

Mitglieder der Promotionskommission:

Vorsitzender: Prof. Dr. Roy Gross
Lehrstuhl für Mikrobiologie
Fakultät für Biologie
Biozentrum der Universität Würzburg – Am Hubland –
97074 Würzburg

Gutachter: Prof. Dr. Thomas Dandekar
Lehrstuhl für Bioinformatik
Fakultät für Biologie
Biozentrum der Universität Würzburg – Am Hubland –
97074 Würzburg

Gutachter: Prof. Dr. Markus Engstler
Lehrstuhl für Zell- und Entwicklungsbiologie (Zoologie I)
Fakultät für Biologie
Biozentrum der Universität Würzburg – Am Hubland –
97074 Würzburg

Tag des Promotionskolloquiums: 27. Oktober 2021

Doktorurkunde ausgehändigt am:

Eidesstattliche Erklärung

Eidesstattliche Erklärungen nach §7 Abs. 2 Satz 3, 4, 5 der Promotionsordnung der Fakultät für Biologie

Hiermit erkläre ich an Eides statt, die Dissertation: „**Software unterstützte Analyse von regulatorischen Elementen in Promotoren mittels AIModules**“, eigenständig, d. h. insbesondere selbständig und ohne Hilfe eines kommerziellen Promotionsberaters, angefertigt und keine anderen, als die von mir angegebenen Quellen und Hilfsmittel verwendet zu haben.

Ich erkläre außerdem, dass die Dissertation weder in gleicher noch in ähnlicher Form bereits in einem anderen Prüfungsverfahren vorgelegen hat.

Weiterhin erkläre ich, dass bei allen Abbildungen und Texten bei denen die Verwertungsrechte (Copyright) nicht bei mir liegen, diese von den Rechtsinhabern eingeholt wurden und die Textstellen bzw. Abbildungen entsprechend den rechtlichen Vorgaben gekennzeichnet sind sowie bei Abbildungen, die dem Internet entnommen wurden, der entsprechende Hypertextlink angegeben wurde.

Würzburg, den

Unterschrift:

Widmung

Meinen Eltern und meiner Familie, die mich auf diesen Pfad geführt und begleitet haben.

Danksagung

Besonderer Dank gilt dem Doktorvater Prof. Dr. Thomas Dandekar sowie den Betreuern Prof. Dr. Markus Engstler und Prof. Dr. Roy Gross, die mit Rat und Tat zur Verfügung standen.

Zudem danke ich Dr. Meik Kunz für die fachlichen Gespräche zu diversen Themen.

Dr. Chunguang Liang danke ich für die Durchsicht des Papers und für die Diskussionen bei technischen Schwierigkeiten der Implementierung sowie der Bereitstellung der Anwendung auf dem Server.

Summary

Regulation of gene expression is at the root of many processes in cellular biology such as cell growth and differentiation. Promoters are the starting points for the transcription of a gene. The promoter itself consists of transcription factor binding sites (TFBSs) which can be closely located or vastly apart. They are recognized and bound by transcription factors (TFs) which themselves can e.g. enhance or silence the transcribing process by the RNA polymerases. Two or more of those transcription factor binding sites within a certain range are called a *module*. Typically, those are found in cells with a nucleus and they may be conserved throughout species. The knowledge of modules may indicate a phylogenetic relationship among species but may also provide insight into the concerted actions of TFs on different genes. Currently there are a number of tools available that can enable a user to find TFBSs on DNA. But at the time of assembling this thesis, there are only two commercial software products available, that can not only detect TFs but also modules. Therefore, we present a free and open source solution that fills this gap by providing a web service that searches for TFBSs and modules on DNA as well as RNA stretches, called *AIModules*. For that, matrices from the Jaspar database or user input matrices are used for motif discovery. Additionally, we show that our tool does TF analysis in seconds, whereas tools like conTraV3 took at least an hour. Furthermore, for the module search the user can specify the degree of conservation of the TFs. We show that with our solution using the JASPAR database we find more modules than a commercially available tool. Moreover, with our application RNA stretches can also be searched for regulatory motifs if suitable matrices are provided. We illustrate this for polyadenylation sites. Thus, our

solution is free and open source, and can be deployed on servers as well as provided on-site on a notebook. We provide a tool to analyze promoters and search for conserved modules as well as common TFBSs in gene families, search for regulatory elements in mRNA such as polyadenylation sites or other regulatory elements such as enhancers or silencers in genomic DNA.

Zusammenfassung

Die Regulation der Genexpression steht am Anfang vieler zellbiologischer Prozesse wie beispielsweise dem Zellwachstum oder der Differenzierung. Gene werden an Promotoren transkribiert, wobei ein Promotor selbst aus vielen logischen Einheiten aufgebaut ist, den Transkriptionsfaktorbindestellen (TFBSs). Diese können sehr nah beieinander liegen, aber auch weit entfernt voneinander sein. Sie werden spezifisch von Transkriptionsfaktoren (TFs) gebunden, die die Transkriptionsrate z.B. verstärken (Enhancer) oder schwächen (Silencer) können. Zwei oder mehr dieser TFBSs mit bestimmtem Abstand werden als *Module* zusammengefasst, die über Spezies hinweg konserviert sein können. Typischerweise findet man Module in Zellen mit einem Zellkern. Spezies mit gemeinsamen Modulen können ein Hinweis auf die gemeinsame phylogenetische Abstammung darstellen, aber auch gemeinsame Funktionsmechanismen von TFs über Gene hinweg aufdecken. Heutzutage sind verschiedene Anwendungen verfügbar, mit denen nach TFBSs in DNA gesucht werden kann. Zum Zeitpunkt des Verfassens dieser Arbeit sind aber nur zwei kommerzielle Produkte bekannt, die nicht nur TFBSs, sondern auch Module erkennen. Deshalb stellen wir hier die freie und quelloffene Lösung *AIModules* vor, die diese Lücke füllt und einen Webservice zur Verfügung stellt, der es erlaubt nach TFBSs sowie nach Modulen auf DNA- und auf RNA-Abschnitten zu suchen. Für die Motivesuche werden entweder Matrizen aus der Jaspar Datenbank oder Matrizen vom Anwender verwendet. Darüberhinaus zeigen wir, dass unser Tool für die TF Suche nur Sekunden benötigt, wohingegen conTraV3 mindestens eine Stunde für dieselbe Analyse braucht. Zusätzlich kann der Anwender bei unserem Tool den Grad der Konserviertheit für TFs mit angeben und wir zeigen, dass wir mit

unserer Lösung, die die Jaspar Datenbank heranzieht, mehr Module finden, als ein kommerziell verfügbares Produkt. Weiterhin kann mit unserer Lösung auch auf RNA-Sequenzen nach regulatorischen Motiven gesucht werden, wenn der Anwender die dafür nötigen Matrizen liefert. Wir zeigen dies am Beispiel von Polyadenylierungsstellen. Zusammenfassend stellen wir ein Werkzeug vor, das erstens frei und quelloffen ist und zweitens entweder auf Servern veröffentlicht werden kann oder On-Site auf einem Notebook läuft. Unser Tool erlaubt es Promotoren zu analysieren und nach konservierten Modulen sowie TFBSs in Genfamilien sowie nach regulatorischen Elementen in mRNA wie z.B. Polyadenylierungsstellen oder andere regulatorische Elemente wie beispielsweise Enhancern oder Silencern in genomischer DNA zu suchen.

Inhaltsverzeichnis

Eidesstattliche Erklärung	i
Widmung	ii
Danksagung	iii
Gutachter	iv
Summary	v
Zusammenfassung	vii
Inhaltsverzeichnis	ix
Abbildungsverzeichnis	xiii
Tabellenverzeichnis	xiv
Abkürzungsverzeichnis	xv
1 Einleitung	1
1.1 Allgemeines	1
1.2 Darstellung von Motiven	6
1.3 Problemstellung	13
1.4 Suche nach Promotorsequenzen	14
1.5 Ist-Stand	15
1.6 Vergleich der Funktionen der betrachteten Tools	28

1.7	Soll-Stand	30
1.8	Adaptierte Tools	31
2	Materialien und Methoden	32
2.1	Jaspar Converter	32
2.2	Tool-Stack	32
2.2.1	Front-End	33
2.2.2	Back-End	34
2.2.3	Datenbank	34
2.3	Architektur	35
2.4	Skalierbarkeit	36
2.5	Entwicklerdokumentation	36
2.5.1	Front-End Entwicklung	36
2.5.2	Back-End Entwicklung	38
2.5.3	Datenbank Entwicklung	39
2.5.4	Serverumgebung installieren	39
2.5.5	Docker	40
2.6	Parameter AIModules und Genomatix	42
3	Ergebnisse	44
3.1	Herausforderungen und Zielsetzungen	44
3.2	AIModules	48
3.2.1	Hauptfenster	49
3.2.2	DNA zu Matrix Konverter	57
3.3	Poly(A)-Suche)	58
3.4	Modulesuche	60
3.5	Analysegeschwindigkeit	61
3.6	TFBS- und Moduleanalyse mit Genomatix	61
3.6.1	TFBSs-Analyse	63
3.6.2	Module-Analyse	66
4	Diskussion	70
4.1	Verfügbare Tools	71

<i>INHALTSVERZEICHNIS</i>	xi
4.2 Vergleich zwischen AIModules und Genomatix . . .	72
4.3 Das Scoring	74
4.4 Analysegeschwindigkeit und Suche nach Polyadenylierungsstellen	76
4.5 Interpretation der Ergebnisse aus <i>AIModules</i> . . .	76
4.6 Anwendungsmöglichkeiten	79
4.7 Zukünftige Features	80
4.8 Fazit	82
Literaturverzeichnis	I
5 Appendix	IX

Abbildungsverzeichnis

1.1	Ein Beispiel für ein Consensus Logo [17]	7
1.2	Ein Beispiel für ein Sequenz Logo [18]	8
1.3	Promotorsuche	14
1.4	Wahl eines Promotors	14
1.5	FASTA	14
1.6	FASTA Sequenz	15
1.7	Motifmap: Modellspezies	16
1.8	Motifmap: Motive	16
1.9	Motifmap: Ergebnis	17
1.10	Promo: Auswahl TFs	17
1.11	Promo: Checkboxen	18
1.12	Promo: Matrizen	18
1.13	Promo: Sequenz	19
1.14	Promo: Sequenzauswahl	19
1.15	Promo: Ergebnis	20
1.16	Prodoric: Genom und Matrizen	22
1.17	Prodoric: Ergebnis	22
1.18	TAIR	24
1.19	PlantPan	25
1.20	conTraV3	27
2.1	Architektur ohne Redundanz	35
2.2	Architektur mit Redundanz	37
3.1	Titelmenü von <i>AIModules</i>	49

3.2	AIModules: Übersicht 1	50
3.3	AIModules: Übersicht 2	51
3.4	AIModules: Matrizen	52
3.5	AIModules: TFBS Ergebnis	53
3.6	AIModules: TFBS Ergebnis (Details)	54
3.7	AIModules: Module Ergebnis	56
3.8	AIModules: Module Ergebnis (Details)	56
3.9	AIModules: Matrix Konverter	57
3.10	AIModules: Matrix Konverter Ergebnis	58
3.11	AIModules: Module Ergebnis für Interleukin	60
3.12	Legende der Interleukinsuche	61

Tabellenverzeichnis

1.1	Beispielalignment	9
1.2	Vergleich der Tools	29
3.1	Poly-A Suche	59
3.2	Parameter: TFBSs Suche für Cathepsine	64
3.3	Ergebnis: TFBSs Suche für Cathepsine	64
3.4	Parameter: TFBSs Suche für IL-10	65
3.5	Ergebnis: TFBSs Suche für IL-10	66
3.6	Parameter: Modulesuche für Cathepsine	67
3.7	Ergebnis: Modulesuche für Cathepsine	67
3.8	Parameter: Modulesuche für IL-10	68
3.9	Ergebnis: Modulesuche für IL-10	69

Abkürzungsverzeichnis

A. thaliana	Arabidopsis thaliana
bp	Basenpaar
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
IL	Interleukin
JavaEE	Java Enterprise Edition
JAX-RS	Java API for RESTful Web Services
JSON	JavaScript Object Notation
kbp	kilo Basenpaar (x 1000)
PWM	position weight matrix
PSSM	position-specific scoring matrix
PSWM	position-specific weight matrix
TF	Transkriptionsfaktor
TFBS	Transkriptionsfaktorbindestelle
XML	Extensible Markup Language

Kapitel 1

Einleitung

1.1 Allgemeines

Der erste Schritt der Prozessierung eines Gens in ein Produkt innerhalb einer Zelle ist die Transkription. Dafür gibt es logische Strukturen auf der DNA, an die RNA-Polymerasen spezifisch binden. Diese Strukturen sind die *Promotoren*. Bei Prokaryoten sind Promotoren weniger unterschiedlich und werden als *starke* bzw. *schwache* Promotoren bezeichnet, bei Eukaryoten sind diese sehr viel unterschiedlicher [37]. Die Bindung der RNA-Polymerase an die Promotoren wird durch Hilfsfaktoren, den sogenannten *Transkriptionsfaktoren* (TFs) unterstützt. TFs binden an spezifische Sequenzen, den cis-regulatorischen Elementen der Transkription wie z.B. GATA-Faktor Bindestelle oder die Tatabox.

Im Falle von Prokaryoten sind das insbesondere die σ (Sigma)-Faktoren [37]. Die mit Abstand meisten Promotoren bei *E. coli* werden über den *Sigma-70* Faktor erkannt. Bei Eukaryoten gibt es eine Vielzahl an *allgemeinen* und *spezifischen* TFs. TFs binden an Transkriptionsfaktorbindestellen (TFBSs). Diese sind DNA-Abschnitte mit einer bestimmten Konsensussequenz. Es zeigt sich, dass die Aktivität von Promotoren mit dem Grad der Übereinstimmung mit der Konsensussequenz korreliert, i.e., dass die Aktivierungsrate eines jeden Gens durch die Übereinstimmung der

– 10 und – 35 Regionen mit der Konsensussequenz gesteuert werden kann [37]. Transkriptionsfaktoren binden DNA z.B. an allgemeinen Motiven wie der TATA-Box oder an RNA-Polymerasen oder den Präinitiationskomplex (TFIIA, TFIIB, etc). Allgemeine bzw. basale Transkriptionsfaktoren treten stets als komplexe mit anderen Proteinen auf und bilden die Plattform für die RNA-Polymerase. Basale TFs sind ubiquitär. Spezifische TFs kommen nur in Zellen vor, in denen ein spezielles Gen aktiviert oder reprimiert werden soll. Aufgrund des Supercoilings eukaryotischer DNA müssen TFs nicht nahe am Transkriptionsbeginn gelegen sein. Entfernungen bis zu 100 kbp sind bekannt. Über diese Distanzen kann z.B. die Transkriptionsrate der RNA-Polymerase beeinflusst (*Enhancer*, *Silencer*) oder die Transkription gar erst ermöglicht werden. Es hat sich allerdings gezeigt, dass für die Bildung des Präinitiationskomplexes wichtigen Bereiche und damit die Positionen der TFBSs in ca. 500 bp upstream/5' zum Transkriptionsstart liegen. In Bakterien dagegen werden weniger Komponenten benötigt: Polymerasebindestelle (TATA-Box) plus Sigmafaktor, gegebenenfalls weitere Bindestelle wie Operator oder Zweikomponentensystemen. Der Teil eines Promotors, der nur diejenigen allgemeinen Promotorelemente enthält, welche absolut notwendig für die Transkription sind, ist der Minimal- oder Core-Promoter.

In Prokaryoten wird die RNA durch eine RNA-Polymerase hergestellt. Dabei ist für die Regulation wie oben erwähnt z. B. der Sigma-70 Faktor beteiligt, aber auch bei Hitzeschock Promotoren der Sigma-H und bei Ammonium-Mangel der Sigma-N Faktor. Die RNA Synthese teilt sich in vier Prozessierungsschritte ein (*Promotor Scanning*, *Initiation*, *Elongation*, *Termination*), wobei der letzten Schritt bei *E. coli* über intrinsische Terminatoren (stabiler G/C-reicher Stem/Loop) oder über den ATP-abhängigen Faktor *Rho* finalisiert wird. Eukaryoten auf der anderen Seite haben drei RNA-Polymerasen. Die Typ I Polymerase transkribiert 5,8S, 18S sowie 28S rRNA, die Typ II Polymerase mRNA sowie

snRNA und die Typ III Polymerase tRNA sowie 5S RNA. Die RNA Polymerase Typ II wird von generellen Transkriptionsfaktoren wie *TFIID* rekrutiert. Dieser TF bindet die TATA-Box über TBP (TATA-Box bindende Proteine), was zu Strukturänderungen am Promotor führt und die Bindung weiterer TFs erlaubt. Zu diesen zählen TFIIA sowie TFIIB. Hieran bindet die Polymerase II zusammen mit TFIIF und TFIIH (Helicase und Kinase) sowie TFIIIE. Mit diesem Konstrukt gelingt die *in vitro* Initiation von unverpackter DNA. Daneben gibt es noch spezielle TFs, die regulierend auf die Transkription wirken, aber nicht direkt mit der Polymerase II interagieren, sondern über Adaptoren wie dem Mediator-Complex. Die speziellen TFs sind nicht immer aktiv, sondern können über Cofaktoren (Hormone, Steroide, etc.) oder über De-/Phosphorylierung an- und abgeschaltet werden. Ähnlich wird auch mit der Tail-Domäne (CTD) der β' Untereinheit der Polymerase II verfahren. Wenn das Serin 5 der CTD phosphoryliert wird, dann wird das Capping der mRNA durchgeführt. Dabei erhält das RNA Molekül cotranskriptional am 5' Anfang ein 7-Methylguanosin Nukleosid, das die Erkennungsfunktion für die Translation im Cytosol für Ribosomen darstellt. Ein phosphoryliertes Serin 2 innerhalb der CTD wiederum rekrutiert Enzyme für die Polyadenylierung am 3' Ende. Diese Struktur dient, wie das Capping auch, der Stabilität der RNA und erhöht die Translationsrate. Bevor die prä-mRNA ins Cytosol exportiert werden kann, müssen mittels des Spleißosoms Introns und/oder Extrons entfernt werden [30]. Beim Menschen besteht dieser Komplex aus den snRNPs U1, U2, U4/U6.U5. Diese binden an den Consensuspleißstellen an den Übergängen Exon-Intron (5'-Spleißstelle), Intron-Exon (3'-Spleißstelle) und innerhalb des Introns (Verzweigungspunkt). Auf diesem Wege werden 99% aller prä-mRNAs U2-abhängig über die Consensussequenz *GU-AG* gespleißt. Daneben gibt es auch die *AT-AC* Introns, die über ein spezialisiertes Spleißosom prozessiert werden (U11, U12, U4/U6.U5). Somit ergibt sich

für den Aufbau einer eukaryotischen mRNA am 5'-Anfang das Cap, gefolgt von 5'UTR und dem Startsignal (AUG), dann die Basensequenz des codierten Proteins, danach das Stoppsignal (UAG, UGA, UAA) und 3'UTR sowie der Poly-A-Schwanz.

RNA-Polymerasen transkribieren an Promotoren. Dieser Vorgang ist aber nicht immer kontinuierlich, sondern es hat sich für unterschiedliche Spezies von Prokaryoten bis Säugen gezeigt, dass für einige Gene die Transkription fluktuiert (*bursts*), i. e., dass Faktoren wie frei verfügbare Nukleotide und andere für dieses Verhalten ursächlich sind und dies somit einen Genexpressionsmechanismus darstellen kann. Diese Fluktuationen können durch Hormone oder durch periodische Assoziationen unterschiedlicher chromosomaler Loci innerhalb der Transkriptionsmaschinerie entstanden sein.

Das Transkriptionsmuster kann kurze Abschnitte der Aktivität mit langen Phasen der Inaktivität dazwischen beschreiben, oder niedriger Aktivität in diesen langen Phasen, oder Aktivitätsschübe aufgrund von Feedback-Schleifen, oder seltene Transkriptionsergebnisse.

Die physiologische Bedeutung der diskontinuierlichen Transkription vor allem in Bakterien oder Hefen führt zu unterschiedlichen Phänotypen der Populationen, da Unterschiede in Menge und Art der transkribierten RNA auch Unterschiede in der Proteinausstattung bedeuten. Es gibt auch Hinweise, dass diese Aktivitätsänderung der Transkription in mehrzelligen Organismen zur Differenzierung von Zelltypen verantwortlich zeichnet, und auch eine Anpassung z. B. bei Antibiotikaresistenz von Bakterien zu sein scheint.

Diese ubiquitäre Fluktuation der Transkription kann auf einen neuen regulatorischen Mechanismus der Genexpression hindeuten, da Fluktuationen in der Transkription auch nachgestellte Fluktuationen z. B. bei der RNA Prozessierung bedingen können [46].

Die Transkription kann auch vor dem regulären Stoppsignal be-

endet werden. Diese vorzeitige Transkriptionsterminierung (premature transcription termination — PTT) ist sehr verbreitet und führt zu Transkripten, die entweder schnell abgebaut werden oder andere Funktionen als noncoding RNA (ncRNA) übernehmen oder über alternative Polyadenylierung Proteine mit anderen Eigenschaften als das Protein mit voller Länge ermöglichen können. Hierdurch wird aber auch das reguläre Produkt der Genexpression negativ reguliert, dies insbesondere bei Genen, die Transkriptionsregulatoren, DNA-Reparatur und Tumor Suppressor Produkte codieren.

PTT kann vor- aber auch nachteilhaft sein, weshalb Faktoren diesen Prozess kontrollieren. Es zeigte sich, dass das U1 snRNP, das beim Spleißvorgang eine Rolle spielt, PTT Stellen in Tausenden von Vertebratengenomen blockiert. Daneben wird PTT besonders in Zellen unter UV Stress vermehrt beobachtet bei zeitgleicher Abnahme der U1 snRNPs. Es bleibt aber unklar, ob PTT eine regulatorische Rolle spielt oder eher nachteilhaft ist und deshalb durch U1 snRNPs unterdrückt werden muss. Dafür spricht, dass physiologische Regulationen eher durch limitierende Faktoren erfolgen, wohingegen U1 snRNPs sehr häufig in der menschlichen Zelle anzutreffen sind (ca. 1 Mio Kopien) [25].

Transkriptionsfaktoren interagieren mit der DNA und sind spezifisch. Man unterscheidet die TF-Typen anhand der DNA-Bindedomänen.

- Helix-turn-helix
- Zink-Finger
- Homöodomäne
- Helix-loop-helix
- Leucin-Zipper

Module sind Anordnungen von TFBSs mit bestimmten Abstand und kommen typischerweise in Zellen mit Zellkern vor. Das Wissen über Module kann ein Hinweis auf Verwandtschaftsbeziehungen zwischen Spezies darstellen und auf gemeinsame Funktionsmechanismen über Gene hinweg hindeuten.

Promotorvorhersagetools können biologische Fragestellungen unterstützen, z. B. im Bereich der Differenzierung, des Zellwachstums oder in der Krebsforschung, denn die Genexpression wird gerade durch Promotoren reguliert. Die Transkription ist zentral, da das An- und Abschalten der Gene ein wichtiger Anpassungsmechanismus für den Organismus darstellt.

1.2 Darstellung von Motiven

Motive sind Aminosäure- oder Nukleotidsequenzen, die häufig vorkommen und mit einer biologischen Funktion in Verbindung stehen. Die Darstellung von Motiven kann auf unterschiedliche Weisen erfolgen. Je nach Anwendungsfall haben einige Darstellungsformen Vorteile gegenüber anderen.

1. *Consensus Sequenz*

Für die Darstellung in textueller Form kann man die Consensus Sequenz verwenden. Diese stellt die häufigsten Aminosäuren oder Basen dar, die aus einem Alignment (Sequenzvergleich mehrerer verwandter Sequenzen) hervorgegangen sind. In dieser Form gehen aber die Frequenzen und Auftretswahrscheinlichkeiten der anderen an dieser Position detektierten Aminosäuren oder Basen verloren. Die Aminosäuren-Consensus-Sequenz $N\{P\}[ST]\{P\}$ gibt z.B. an, dass an der ersten Position ein Asparagin, an der zweiten Position *kein* Prolin, an der dritten *entweder* ein Serin *oder* ein Threonin und an der vierten Position *kein* Prolin steht. Es ist aber auch eine simple Form möglich: GCTTAG

2. *Consensus Logo*

Eine weitere textuelle Form ist die des Consensus Logo (Abbildung 1.1). Diese Abbildung stellt eine vereinfachte Darstellung eines Sequenz Logos dar, und enthält an den jeweiligen Positionen die mit höchster Frequenz auftretenden Aminosäure oder Base. Allerdings werden in einem Consensus Logo nur die Information der Konserviertheit und nicht die Informationen über die Frequenzen einer jeden Aminosäure oder Base der jeweiligen Positionen ausgedrückt. Die Frequenz und damit die Konserviertheit wird durch die Höhe des Buchstaben für die jeweilige Aminosäure oder Base angedeutet. Vorteilhaft ist, dass der Informationsgehalt dieses Consensus Logo mit in den Text eingebettet werden kann [41, 42].

(2 bits) | _{TC}GAA_{CA}TATGTT_{TC}G_A [IC: 14.748 bits]

Abbildung 1.1: Ein Beispiel für ein Consensus Logo [17]

Consensus Sequenz und Consensus Logo sind gegenüber dem Sequenz Logo platzsparend und können in Manuskripten bei Platznot verwendet werden. Zudem unterbrechen diese den Textfluss im Gegensatz zu einem Sequenz Logo nicht.

3. *Sequenz Logo*

Ein Sequenz Logo (Abbildung 1.2) wird über ein Alignment erstellt und zeigt für jede Position einen Buchstabenstapel. Die Größe der Buchstaben ergibt die Frequenz an dieser Position und den Informationsgehalt in *Bit*. Konservierte Positionen haben einen Informationsgehalt von Zwei Bit. Wenn zwei Basen an einer Position jeweils zu 50 % vorkommen, hat jede der beiden Basen einen Informationsgehalt von Einem Bit. Kommen alle vier Basen an einer Position gleich häufig vor, ergibt sich für jede Base ein Informationsgehalt von Null Bit

[41]. Ein Sequenz Logo kann für Aminosäure- und Basensequenzen erstellt werden. Es kann auch die Consensus Sequenz wiedergeben.

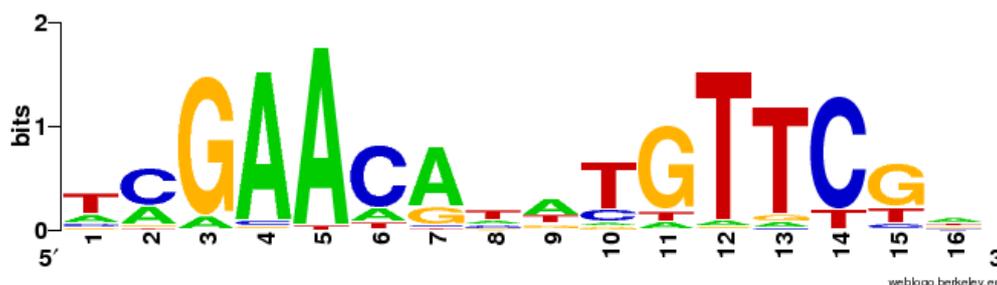


Abbildung 1.2: Ein Beispiel für ein Sequenz Logo [18]

4. Matrizen

Um neue TFBSs zu finden wurden früher Consensus Sequenzen benutzt. Diese hatten jedoch bei den Vorhersagen Limitierungen, die man durch Matrizen ausgeräumt hat [6, 48]. Matrizen werden durch Alignments von Sequenzen, die funktionell verwandt sind, oder durch Genexpressions- oder Chip-chip-Daten erstellt [28, 58]. Der Vorteil ist, dass eine Matrize alle Wahrscheinlichkeiten aller Basen an jeder Position wiedergibt. So kann die Ähnlichkeit zwischen der Matrize und einer potentiellen TFBS berechnet werden. Es lassen sich dann über einen Schwellenwert (Threshold) nur positive Befunde ausgeben.

Matrizen stellen Motive dar und sind auch unter den Namen position weight matrix (PWM), position-specific scoring matrix (PSSM) und position-specific weight matrix (PSWM) bekannt. Hierdurch wird versucht die intrinsische Variabilität eines Motivs abzubilden. Eine solche Matrix besteht aus vier Zeilen für Basen oder 20 Zeilen für Aminosäuren. Die Spalten geben die Häufigkeiten der Basen oder Aminosäuren wieder. Es gibt genau soviele Spalten wie die Länge der Sequenzen

beim Alignment. Es werden drei Matrizenarten unterschieden. Im folgenden wird nur der Fall für Basen betrachtet [20, 55], analog muss man für Aminosäuren vorgehen.

(a) *Position Frequency Matrix (PFM)*

Eine PFM stellt die absolute Häufigkeit des Auftretens von Basen an jeder Position eines Sequenzalignments dar. Mit den Sequenzen aus Tabelle 1.1, die aus Vertrebraten stammen und den Übergang exon/intron beinhalten, ergibt sich somit die Matrix 1.1.

Tabelle 1.1: Beispielalignment

Sequenz 1	GAGGTAAAC
Sequenz 2	TCCGTAAGT
Sequenz 3	CAGGTTGGA
Sequenz 4	ACAGTCAGT
Sequenz 5	TAGGTCATT
Sequenz 6	TAGGTAAGT
Sequenz 7	ATGGTAACT
Sequenz 8	CAGGTATAC
Sequenz 9	TGTGTGAGT
Sequenz 10	AAGGTAAGT

$$M = \begin{matrix} A \\ C \\ G \\ T \end{matrix} \begin{bmatrix} 3 & 6 & 1 & 0 & 0 & 6 & 7 & 2 & 1 \\ 2 & 2 & 1 & 0 & 0 & 2 & 1 & 1 & 2 \\ 1 & 1 & 7 & 10 & 0 & 1 & 1 & 5 & 1 \\ 4 & 1 & 1 & 0 & 10 & 1 & 1 & 2 & 6 \end{bmatrix} \quad (1.1)$$

Jedes dieser Matrizenelemente zeigt die Anzahl der Basen an einer bestimmten Position über alle zehn Sequenzen an. Das Adenosin kommt z. B. an Position eins dreimal vor. An den Positionen vier und fünf sind Guanodin respektive Thymin als Startpunkt des Introns konserviert und kommen damit in allen zehn Sequenzen vor.

Aus diesem PFM kann man ein PPM generieren.

(b) *Position Probability Matrix (PPM)*

Aussagekräftiger als die absoluten Häufigkeiten sind Motive, die relative Häufigkeiten (PPM) enthalten, da man hier die Elemente der Matrix als Auftrittswahrscheinlichkeit einer Base in einer funktionellen Sequenz heranziehen kann. D. h., dass man eine PPM auf eine Sequenz so anwenden kann, dass man die Wahrscheinlichkeiten der einzelnen Positionen multipliziert.

Dazu muss eine PFM normalisiert werden, indem man die einzelnen Werte durch die Anzahl der Sequenzen teilt, die für das Alignment hergenommen wurden. Formal ausgedrückt berechnen sich die Elemente eines PPM M aus der Menge S von n alignierten Sequenzen der Länge l über die Formel 1.2.

$$M_{i,j} = \frac{1}{n} \sum_{k=1}^n I_i(s_{k,j}) \quad (1.2)$$

mit $s_k = s_{k1}, \dots, s_{kl}$, $s_{k,j}$ ein Element aus $\{A, C, G, T\}$, $i = A, C, G, T$, $j = 1, \dots, l$ und der Indikatorfunktion $I_i(q) = \begin{cases} 1, & \text{wenn } i = q \\ 0, & \text{andernfalls} \end{cases}$

Es ergibt sich dann die Matrix 1.3.

$$M = \begin{matrix} A \\ C \\ G \\ T \end{matrix} \begin{bmatrix} 0,3 & 0,6 & 0,1 & 0,00 & 0,00 & 0,6 & 0,7 & 0,2 & 0,1 \\ 0,2 & 0,2 & 0,1 & 0,00 & 0,00 & 0,2 & 0,1 & 0,1 & 0,2 \\ 0,1 & 0,1 & 0,7 & 1,00 & 0,00 & 0,1 & 0,1 & 0,5 & 0,1 \\ 0,4 & 0,1 & 0,1 & 0,00 & 1,00 & 0,1 & 0,1 & 0,2 & 0,6 \end{bmatrix} \quad (1.3)$$

In einer PPM kann es bei einer kleinen Anzahl von zu alignierenden Sequenzen oder bei konservierten Positionen vorkommen, dass ein Element der Matrize Null ergibt.

Als Folge muss mit Null multipliziert werden (z. B. Adenosin in Position vier). Damit wäre jedes Ergebnis Null. Um dies zu heilen benutzt man sogenannte *pseudocounts*. Diese sind Werte, die im besten Falle unabhängig von der Anzahl der zu alignierenden Sequenzen sind und mit der Entropie der Originalmatrize korrelieren. Optimierte Pseudocounts können für jede Matrize generiert werden. Es wurde aber auch für die Praxis vorgeschlagen, dass man zu jedem Element in der PPM 0,8 dazuaddiert [33].

(c) *Position Weight Matrix (PWM)*

Bei der Auswertung fällt es leichter, wenn man anstelle der Wahrscheinlichkeiten einen logarithmierten Quotienten verwendet. Dies hat den Vorteil, dass logarithmierte Wahrscheinlichkeiten (log-likelihood) einerseits addiert anstatt multipliziert werden können. Dies bringt bei der Berechnung Beschleunigungen bei Platz- und Speicherkomplexität. Andererseits müssen bei PPMs Multiplikationen von Gleitkommazahlen (Dezimalzahlen) vorgenommen werden. Da ein Computer nur eine limitierte Gleitkommadarstellung von Brüchen unterstützt, kommt es hier sehr schnell zu Rundungen nach dem Komma. Daher haben Additionen eine höhere numerische Stabilität. Zusätzlich werden hier die Wahrscheinlichkeiten gewichtet, indem diese durch die Hintergrundwahrscheinlichkeit der jeweiligen Basen geteilt werden. Das verringert die Entropie und erhöht den Informationsgehalt. Um aus einer PPM eine PWM zu berechnen benutzt man die Formel 1.4

$$M_{i,j} = \log_2\left(\frac{M_{i,j}}{p_i}\right) \quad (1.4)$$

mit $M_{i,j}$ (nach dem Istgleichzeichen) als die Wahrscheinlichkeit der Base i an der Position j in der PPM und p_i

als Hintergrundwahrscheinlichkeit der Base i . Die Hintergrundwahrscheinlichkeit (background model) ist im einfachsten Fall die Wahrscheinlichkeit des Auftretens einer Base aus den vier Basen (A, C, G, T), also $1/4 = 0,25$, oder die Häufigkeit einer Base in allen zehn alignierten Sequenzen, oder die Häufigkeit der Base in allen zehn alignierten Sequenzen an dieser Position. Man kann es aber auch so formulieren, dass bei Species mit einem hohen G/C Anteil das background model für diese Basen erhöht und für A/T erniedrigt wird. Man erhält die PWM aus 1.5.

$$M = \begin{matrix} A \\ C \\ G \\ T \end{matrix} \begin{bmatrix} 0,26 & 1,26 & -1,32 & -\infty & -\infty & 1,26 & 1,49 & -0,32 & -1,32 \\ -0,32 & -0,32 & -1,32 & -\infty & -\infty & -0,32 & -1,32 & -1,32 & -0,32 \\ -1,32 & -1,32 & 1,49 & 2,00 & -\infty & -1,32 & -1,32 & 1,00 & -1,32 \\ 0,68 & -1,32 & -1,32 & -\infty & 2,00 & -1,32 & -1,32 & -0,32 & 1,26 \end{bmatrix} \quad (1.5)$$

Anhand dieses Beispiels kann man sehr gut den Sinn von *pseudocounts* nachvollziehen. Die Werte $-\infty$ würden dadurch verhindert.

Die Auswertung mit einer PWM erfolgt durch Addition der zu jeder Position der auszuwertenden Sequenz korrespondierenden log-Wahrscheinlichkeit. Man erhält ein Ergebnis, den *score*. Dieser gibt eine Tendenz an, mit der die Sequenz unterschiedlich von einer zufälligen Sequenz ist. Der Score ist Null, wenn die Sequenz mit gleicher Wahrscheinlichkeit eine funktionale und eine zufällige Sequenz ist, ist größer Null, wenn die Sequenz eher eine funktionale Sequenz ist als eine zufällige und kleiner als Null, wenn die Sequenz eher zufällig ist.

Für ein Scoring können alle vorgestellten Matrizen hergenommen werden. Es bietet sich aber aufgrund des höheren

Informationsgehalts die *PWM* gegenüber den anderen Formen an. Das Tool *AIModules* verwendet als Inputparameter *PFMs*. Bei der Berechnung werden diese in log-likelihoods konvertiert. Möchte man die Ergebnisse aus dem Scoring grob vorsortieren, definiert man einen Schwellenwert. Für diesen Threshold nimmt man den geringsten Score für die Ausgangssequenz, den man mit der Matrize erhält, wenn man alle Ausgangssequenzen einzeln einem Scoring unterzieht.

1.3 Problemstellung

Zum Zeitpunkt der Erstellung dieser Arbeit gibt es Anwendungen, die TFBSs auf DNA-Sequenzen identifizieren können. Diese Anwendungen sind allerdings nicht in der Lage Module auszuweisen — mit der Ausnahme von *Genomatix* [16] und *TRANSFAC* [1, 52]. *Genomatix* muss nach einer kostenfreien Periode lizenzpflichtig erworben werden. *TRANSFAC* steht nach einer Registrierung als kostenfreie Version zur Verfügung. Diese enthält allerdings veraltete Matrizen aus 2005 in einer sehr eingeschränkten Anzahl (398 Matrizen). Zudem ist diese freie Version funktionell beschränkt [51], sodass die volle Funktionsfähigkeit nur mit der lizenzierten *Professional Version* verfügbar ist. In der Biologie sind Promotoren essentiell für die Genregulation. Insofern erlaubt deren Identifikation Anwendungen in der Molekularbiologie oder Krebsforschung. Deshalb ist es bedauerlich, dass es gegenwärtig so wenige Werkzeuge zu deren Bestimmung gibt. Da eine kostenlose Alternative, die TFBSs und Module findet, fehlt, wurde *AIModules* entwickelt (<https://aimodules.de>).

1.4 Suche nach Promotorsequenzen

Promotorsequenzen sind über die Seite <https://www.ncbi.nlm.nih.gov/> abrufbar. Dafür muss dort die Datenbank auf *Nucleotide* gesetzt und als Suchparameter z. B. *h.sapiens il-10* eingegeben werden (Abbildung 1.3). Die Suche liefert eine Liste von



Abbildung 1.3: Konfiguration der Promotorsuche auf <https://www.ncbi.nlm.nih.gov/>

Ergebnissen, aus der wie in Abbildung 1.4 gezeigt, eines per Klick ausgewählt werden kann. Auf der neu geladenen Seite klickt man



Abbildung 1.4: Wahl eines Promotors

auf den Link *FASTA* (Abbildung 1.5). Man erhält die Promotor-

H.sapiens promoter region of human IL-10 gene

GenBank: X73536.1

[FASTA](#) [Graphics](#)

Abbildung 1.5: Die FASTA Sequenz des Promotors anzeigen lassen

sequenz im FASTA-Format (Abbildung 1.6). Mit diesem Format können die meisten Anwendungen und Services arbeiten.

Sollten keine Promotorsequenzen für ein Gen auffindbar sein, kann man auch ab Transkriptionsstart eines Gens *upstream* die ersten 500 bp zur Analyse heranziehen, da hier sehr viele regulatorische Elemente aufzufinden sind.

FASTA ▾

H.sapiens promoter region of human IL-10 gene

GenBank: X73536.1

[GenBank](#) [Graphics](#)

```
>X73536.1 H.sapiens promoter region of human IL-10 gene
AGCTTT CAGCAAGT GCAGACT ACT CTT ACCCACTT CCCCCAAGCACAGTT GGGGT GGGGGACAGCT GAAG
AGGT GGAAACAT GT GCCT GAGAAT CCT AAT GAAAT CGGGT AAAGGAGCCT GGAACACAT CCT GT GACCC
CGCCT GT CCT GT AGGAAGCCAGT CT CT GGAAAGT AAAAT GGAAGGGCT GCTT GGGAACTTT GAGGAT ATT
TAGCCCACCCCT CATT TTT ACTT GGGGAACT AAGGCCAGAGACCT AAGGT GACT GCCT AAGTT AGCA
AGGAGAAGT CTT GGGT ATT CAT CCCAGTT GGGGGACCCAATT ATTT CT CAAT CCCATT GT ATT CT GGA
AT GGGCAATTT GT CCACGT CACT GT GACCT AGGAACACGCGAAT GAGAACCACAGCT GAGGGCCT CT GC
GCACAGAACAGCT GTT CT CCCCAGGAAAT CAACTTTTTT AATT GAGAAGCT AAAAAATT ATT CT AAGAG
AGGT AGCCCAT CCT AAAAAAT AGCT GT AAT GCAGAAGTT CAT GTT CAACCAAT CATT TTT GCTT ACGAT GC
AAAAATT GAAACT AAGTTT ATT AGAGAGTT AGAGAAGGAGGAGCT CT AAGCAGAAAAAT CCT GT GCC
GGGAAACCTT GATT GT GGCTTTTT AAT GAAT GAAGAGGCCT CCCT GAGCTT ACAAT AT AAAAGGGGGACA
GAGAGGT GAAGGT CT ACACAT CAGGGGCTT GCT CTT GCAAAACCAACCACAAGACAGACTT GCAAAAGA
AGGCAT GCACAGCT CAGCACT GC
```

Abbildung 1.6: Die FASTA Sequenz des Promotors wird angezeigt

1.5 Ist-Stand

In diesem Abschnitt werden kurz die Anwendungen und Services vorgestellt, mit denen TFBSs vorhergesagt werden können.

1. *Motifmap*

Motifmap ist ein Tool mit dem man TFBSs für Modellspezies anzeigen lassen kann [8, 35, 59] (Abbildung 1.7, 1.8, 1.9). Dies nicht nur mittels TFs die in Datenbanken hinterlegt sind, sondern auch mittels Genomalignments und statistischen Verfahren. Allerdings kann nicht nach Modulen gesucht werden.

Select species track and click next

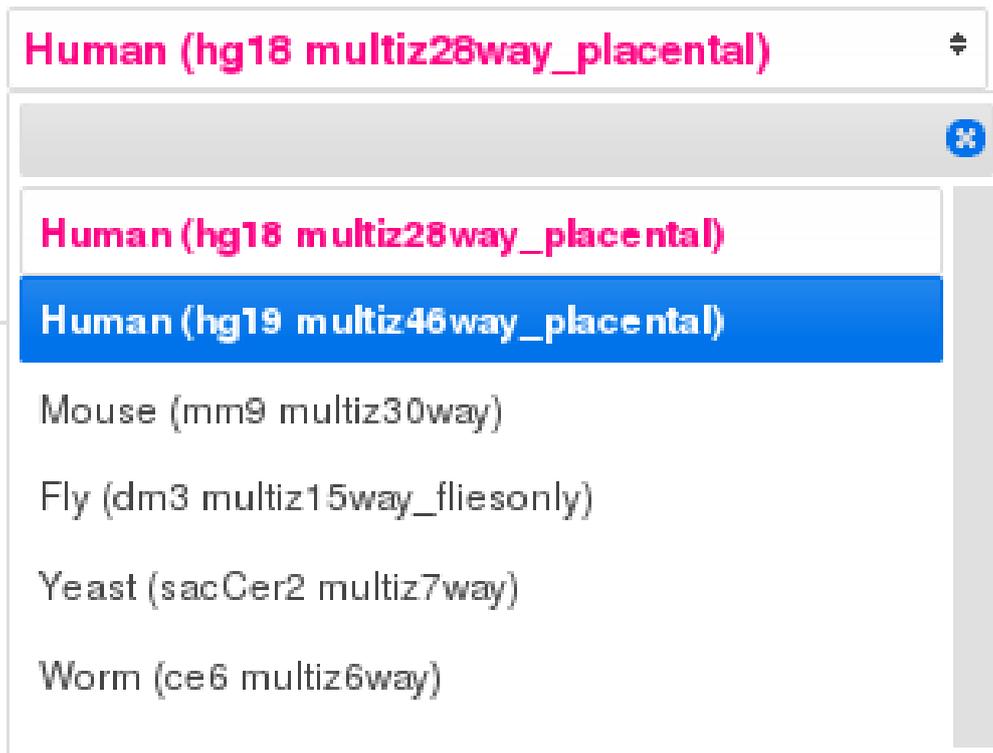


Abbildung 1.7: Modellspezies können selektiert werden. *Diese Abbildung wollen wir auch veröffentlichen (eingereicht bei BMC Bioinformatics)

Motif ID	TF Name	Consensus	Length
LM1_RFX1	RFX1	BDGTTKCCATGGMAACMV	18
LM2_CTCF	CTCF	NNDCCASHAGRKGGCASYV	19
LM4_M2	LM4_M2	DGCMTKCTGGGARWTGTAGTYY	22
LM9_NRSE	NRSE	DTCAGCACCNHGGACAGNDVM	21

Abbildung 1.8: Ein Motif kann selektiert werden. (* siehe Abbildung 1.7)

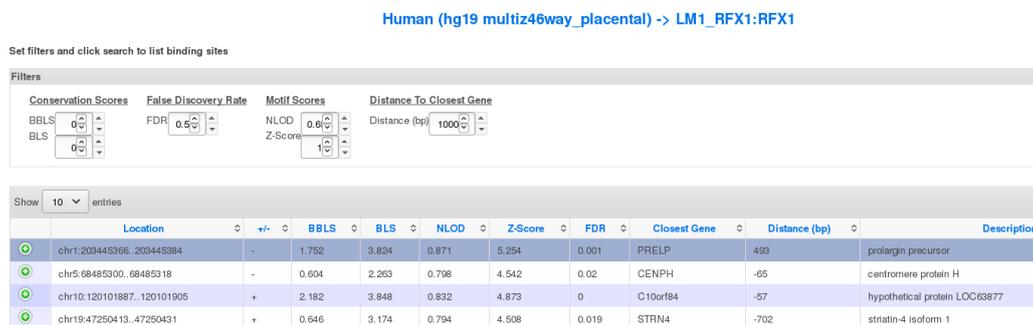


Abbildung 1.9: MotifMap sucht nach diesem Motif innerhalb des Genoms dieses Modellorganismus. (* siehe Abbildung 1.7)

2. Promo

Promo identifiziert TFBSs in DNA Sequenzen einer Spezies oder einer Gruppen von Spezies. Die Matrizen für das Scoring kommen aus der Matrizendatenbank *TRANSFAC 8.3* [13, 40, 31]. Im ersten Schritt werden TFs ausgewählt, nach denen gesucht werden soll. Als nächstes kann man DNA Sequenzen eingeben, die durchsucht werden sollen. Das Ergebnis der Suche wird auf der Ergebnisseite angezeigt (Abbildung 1.10, 1.11, 1.12, 1.13, 1.14, 1.15). Eine Suche nach Modulen ist nicht möglich.

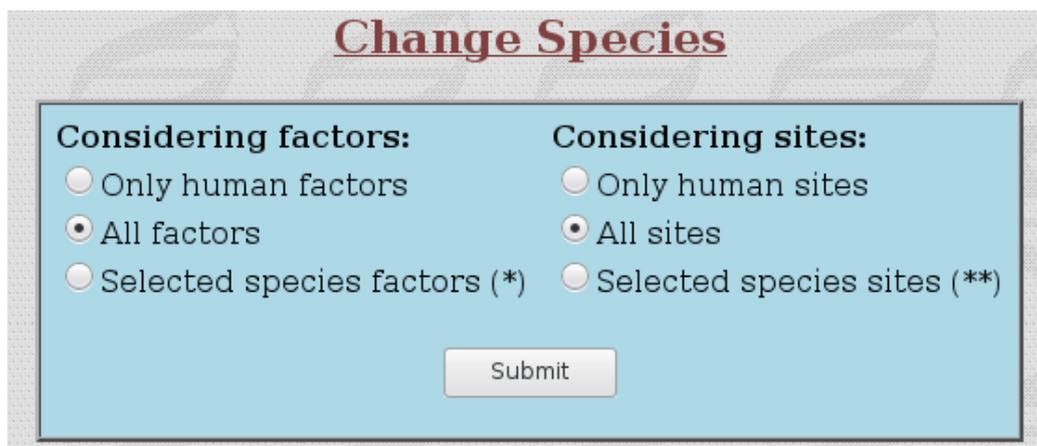


Abbildung 1.10: Erster Schritt: Welche TFs sollen für die Analyse Verwendung finden. (* siehe Abbildung 1.7)



Abbildung 1.11: TFs und Bindestellen können auch mittels Checkboxes ausgewählt werden. (* siehe Abbildung 1.7)

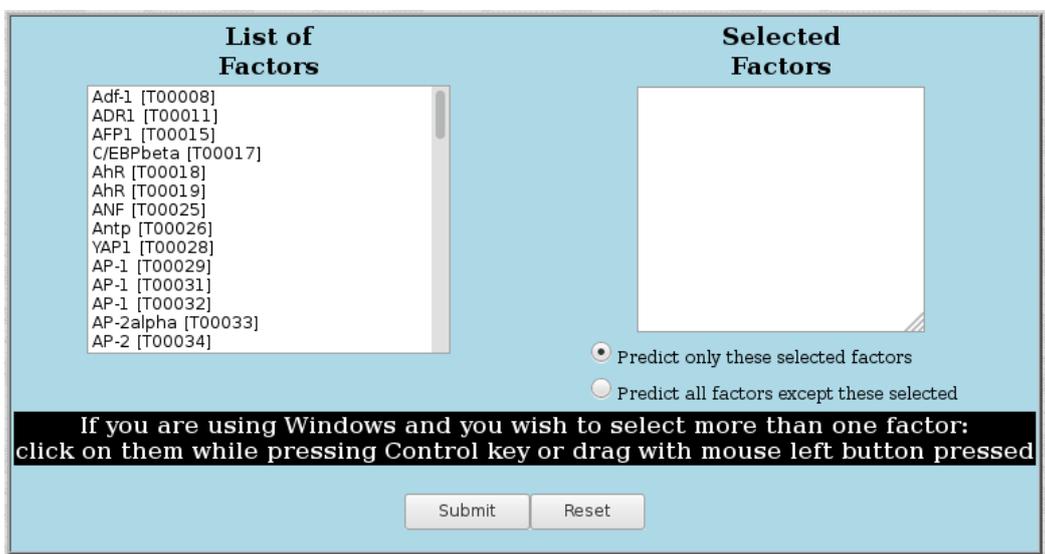


Abbildung 1.12: Matrizen können auch direkt für die Analyse selektiert werden. (* siehe Abbildung 1.7)

15 Maximum matrix dissimilarity rate

STRING TO BE ANALYZED

or

FILE TO BE ANALYZED (in FASTA format)

Browse... No file selected.

Submit

Abbildung 1.13: Eine Sequenz kann zur Analyse ausgewählt werden ... (* siehe Abbildung 1.7)

Sites found in all sequences

Sites found in 50 % of sequences 15 Maximum matrix dissimilarity rate

Sites found in 2 or more sequences

STRING TO BE ANALYZED (in FASTA format)

or

FILE TO BE ANALYZED (in FASTA format)

Browse... No file selected.

Submit

Abbildung 1.14: ... oder mehrere Sequenzen. (* siehe Abbildung 1.7)

	1	10	20	30	40
Homo_sapiens_cathepsin_V_transcript_variant-1_promoter	0 1 2 3 4 5 6 7 8 9 6 13 8	9 10 11 12 13 14 15 16 17 18 19 26 27 35 97 98 146	0 1 8 9 13 15 16 20 21 22 25 26 27 41 97 98	0 1 9 10 13 14 17 23 24 26 27 28 29 30 31 36 37 38 39 40 145	
Bos_taurus_cathepsin-Z_promoter	8 9 12 13 15 16 26 27 97 98 113 145	9 10 12 13 26 27 82 83 101 102 103 120 121 125 126 127 128 129 130 132 145	42 53 65 78 79 80 81 82 83 84 85 123 125 126 127 128 129 130 131 132 133 137	4 5 7 10 11 46 47 48 49 50 51 58 70 71 134 140	
Mus_musculus_cathepsin-F-transcript-variant-X1_promoter_576-1076	9 11 12 13 14 15 16 17 18 19 22 26 27 46 48 49 50 51 61 70 99 112 136 140	8 24 28 29 30 31 32 38 39 41 46 70 73 90 91 96 107 108 109 115 118 138 146	6 8 9 10 12 13 14 15 16 17 19 26 27 33 35 52 54 92 95	9 22 52 53 58 59 87 113	

Abbildung 1.15: Die Ergebnisseite zeigt TFBSs, die farblich kodiert sind. (* siehe Abbildung 1.7)

3. ModuleMaster

ModuleMaster ist ein Tool, das entweder als Java WebStart Anwendung oder als herunterladbare .jar-Datei, allerdings nur als Kommandozeilentool, gestartet werden kann. Es sucht in DNA-Sequenzen nach Modulen mittels der TRANSFAC[®] Matrizendatenbank oder eigener Matrizen [56, 57].

Die TRANSFAC[®] Professional Datenbank ist aus Lizenzgründen nur in der WebStart Version enthalten, die Kommandozeilenversion beinhaltet die öffentlich zugängliche, aber veraltete, TRANSFAC[®] Datenbank. Das Tool konnte allerdings weder als WebStart- noch als Kommandozeilenversion auf den Betriebssystemen Linux Mint 19.1 Tessa 64bit und Microsoft Windows 10 Professional 64 bit gestartet werden. Nach Aussagen von Prof. Dr. Andreas Zell von der Universität Tübingen, gäbe es seit Jahren keine Bioinformatik-Forschung mehr in Tübingen und deshalb werde dieses Tool auch nicht mehr unterstützt. Aus diesem Grund wird dieses Tool nur der Vollständigkeit halber hier erwähnt. Zusätzlich gab es beim Versuch des Anwendungsstarts eine Zertifikatswarnung. Die Anwendung wurde versucht folgendermaßen zu starten:

```
$ javaws mm.jnlp
```

und

```
$ javaws http://www.ra.cs.uni-tuebingen.de/software/  
ModuleMaster/downloads/start.php?fn=mm.jnlp
```

In beiden Fällen gab es diesen Fehler:

```
Caused by: java.lang.NoClassDefFoundError: javax/xml/  
soap/SOAPException
```

Dieser Fehler konnte nicht behoben werden.

4. *Prodoric*[®]

Prodoric[®]¹ [32, 39] enthält TFBSs für Prokaryoten. Mit den enthaltenen Tools lassen sich u. a. Genregulationsnetzwerke visualisieren. Im Tool *Virtual Footprint* und im Submodul *Promotor Analysis* kann nach TFBSs gesucht werden. Zuerst muss der Anwender ein Genom auswählen, um daraufhin durch eine Auswahl an vorhandenen Matrizen Bindestellen zu finden (Abbildung 1.16, 1.17). Eine Suche nach gemeinsamen Modulen ist nicht möglich.

5. *Softberry*

Softberry [47] ist ein umfangreicher Service, mit dem u. a. Promotoren vorhergesagt werden können.

- PROMH(G): Promotorvorhersage mittels verwandter Sequenzen [53]
- Nsite: Auffinden von regulatorischen Elementen [23, 54]
- NsiteH: Suche nach regulatorischen Elementen in orthologen Sequenzen [23]
- NsiteM: Suche nach konservierten regulatorischen Elementen [23, 54]

¹*PROcaryotIC Database Of Gene Regulation*

➔ Step 1: Input DNA Sequence...

➔ Step 2: Select Pattern and Start Searching...

Abbildung 1.16: ProDoric: Auswahl von Genom und Matrizen. (* siehe Abbildung 1.7)

Promoter Sequence:

```

1 CTAGGTTGCT CAGGTGCTCG GTCATATCTG ACTCTTTTTC AAGCCTGGG ACTGTAGCTT GCCAAGTCC TTTTCAATG 80
81 GAACTTTTCA GCGGAKRATA CTGCACTGG TTGCCATGCC GTCCTCATC GPRAGAGTGG ATAGGOTGG ADTTRTGAC 160
181 GAGACCCCT GCGGALAYC CTGTGAGTGT GAGCCAGAG GAGATGGGG GATCCGGGG ATCTTCTCG AGCAGAGGT 240
241 CCCAAGACCC CAAAGCTGGG GGTGGGGGG CCTAAGATG AATGAGTGT GATAGATCAG AGGCAAGAG CAAGTGGGG 320
321 GGTGGGGGG GGTTCGCGCC AGGCTTGGGC CTGACCCGA GTCCGGACGG GCCGGCCAGC GGGGGGGGG GGGGCGGGT 400
401 CAGCGGGGCTCCCGCCG CCGGCCCCGG GGGGTCTCC GCGGGGGC GTCGAGTGA AGGGCTTCCG GGGCGGGG 480
481 GACCCCTCCG GAGGGTGCAC
    
```

PWM (species)	Start Position	End Position	Stand	Score	Sequence
AlgU (-10) Pseudomonas aeruginosa (strain ATCC 15692 / PAO1)	77	85	-	6.00	AATCCCAT
AlgU (-10) Pseudomonas aeruginosa (strain ATCC 15692 / PAO1)	286	294	+	5.95	GGTCTGATA
AlgU (-10) Pseudomonas aeruginosa (strain ATCC 15692 / PAO1)	271	279	-	5.93	ATTCTTAGG

Abbildung 1.17: ProDoric: Tabellarische Darstellung der Ergebnisse. (* siehe Abbildung 1.7)

Es ist möglich eine Sequenz einzugeben, die dann analysiert wird — zusätzlich eine weitere Sequenz, wenn auf Konservertheit der Bindestellen geprüft werden soll. Die Ergebnisse werden textuell angezeigt. Module können nicht angezeigt werden.

6. *TAIR*

TAIR² [49] ist ein Promotoranalysetool, bei dem der Anwender Gene angeben kann. Diese werden auf TFBSs, die mindestens in drei Sequenzen vorkommen müssen, ausgewertet. Die Auswertung ist beschränkt auf Motive der Länge sechs. Der Service ist für den Modellorganismus *Arabidopsis thaliana* vorgesehen (Abbildung 1.18). Eine Modulesuche wird nicht unterstützt.

7. *PlantPan 3.0*

PlantPan 3.0³ [4, 38] ist ein Vorhersagetool für TFs und TFBSs vornehmlich in Pflanzenspecies. TFs können als IUPAC Code eingegeben oder aus einer Liste ausgewählt werden. Mit dem Tool *Gene Group Analysis* können eigene Sequenzen nach Bindestellen durchsucht werden (Abbildung 1.19). Die Suche nach gemeinsamen Modulen wird nicht unterstützt.

8. *TESS — Transcription Element Search System*

TESS⁴ [43, 44, 45] war ein Web-Vorhersagetool für TFBSs. Laut Website ist das Webtool aber nicht mehr erreichbar, da der Autor die Gruppe verlassen hat. Der Quellcode des Backends ist verfügbar. Auf diesen Rechenkern, der TFBSs bestimmt, wird im Teil 1.8 weiter eingegangen.

²The Arabidopsis Information Resource

³The Plant Promoter Analysis Navigator

⁴Laut Softwarelizenz muss der folgende Satz in diesem Dokument enthalten sein: „This product includes software developed by CBIL at the Center for Bioinformatics at the University of Pennsylvania.“

Statistical Motif Analysis in Promoter or Upstream Gene Sequences

The program compares the frequencies of 6-mer "words" in your query set of sequences (on both strands) with the frequencies of the words in the current genomic sequence set of 33518 sequences, each containing 500 (or 1000) bp upstream of the start codon of each gene. You can type in sets of AGI locus identifiers (e.g. At1g01030) or sets of fasta sequences. Make sure each fasta header is formatted as such, fasta symbol (>), immediately followed by a unique ID, a space, then all other descriptions (e.g. >ABCD1.1 my gene). Ensure that there are no sequences appearing more than once in your query set.

Upload file: No file selected.

Dataset:

500 bp upstream 1000 bp upstream 3000 bp upstream

Output type:

HTML Text

Abbildung 1.18: TAIR: Suche nach Bindestellen der Länge sechs in *A. thaliana* (* siehe Abbildung 1.7)

Step 1 Please input promoter sequences (FASTA format)

```

>Mycn1_test_005
CGCAATAGCGGACGTCATCTTCTAGCTCGGGATCTCCGGAAGTCTCGGTGATCGCTCCAGGATGCGCAGCAGCTCGGTCTCGGT
CGTCCACTGATCCGAAACCGGTTTTGATTTCTTTCAGACTAAAAAAGCTGTAAAAAGCCGCGCCCGGACACGCGGTAAAGCTCAAGAGG
CGCGGATCA

```

Step 2 Please specify TF binding motifs

PlantPAN 3.0 database
Please select transcription factors from the following species:

All species

Choose species

<input checked="" type="checkbox"/> Arabidopsis thaliana	<input type="checkbox"/> Brachypodium distachyon	<input type="checkbox"/> Chlamydomonas reinhardtii	<input type="checkbox"/> Glycine max
<input type="checkbox"/> Malus domestica	<input type="checkbox"/> Oryza sativa	<input type="checkbox"/> Populus trichocarpa	<input type="checkbox"/> Sorghum bicolor
<input type="checkbox"/> Volvox carteri	<input type="checkbox"/> Zea mays		

User-customized motifs
Input custom motif sequence with IUPAC code and group with "[]" (EX: [GC]SS[GC]GC):

```

RDWMTB
ACGT
A[CA][CGT]G[AT][TCA]T
TATATAT
RDWSSB
RYSNKBDHVN

```

Step 3 Set parameters

► How many co-occurrence TFBSs you want to analyze? 1 2

► Input the threshold for analysis
Support >= % (the frequency of promoters containing the TF/TFBS)

Abbildung 1.19: PlantPan: In diesem Tool können eigene Sequenzen nach TFBSs durchsucht werden. (* siehe Abbildung 1.7)

9. *Genomatix*

Genomatix [16] ist mit einem Testaccount 7 Tage kostenfrei, aber mit Einschränkungen in der Funktionalität, nutzbar. Dieses Tool kann TFBSs vorhersagen und dazu die putativen Module darstellen — auch für eigene Sequenzen. Es können individuelle Matrizen und auch Module hinterlegt werden. Folgende Tools sind enthalten:

- MatInspector: Suche nach TFBSs, Restriktionsstellen, etc.
- Common TFs: Sucht in mehreren Sequenzen nach gemeinsamen TFBSs
- MatDefine: Erstellung von Matrizen mittels eigener kurzer Sequenzen
- FastM: Erstellung von eigenen Models (Modulen) mit den jeweiligen Abständen dazwischen
- ModelInspector: Suche nach zusammen auftretenden oder selbst definierten Modulen

Das Programm ModelInspector sucht in jeder vom Anwender eingegebenen Sequenz einzeln nach Modulen, aber zeigt nicht gemeinsame Module aller Sequenzen an.

10. *TRANSFAC*

TRANSFAC [1, 52] ist in der Lage TFBSs und Module vorherzusagen. Dazu können in der Datenbank vorhandene eukaryotische Sequenzen, Matrizen und Module herangezogen werden oder eigene Sequenzen, Matrizen und Module Verwendung finden. Laut Webseite kann eine Testperiode⁵ für TRANSFAC Professional ermöglicht werden, aber auch die Benutzung einer öffentlich zugänglichen, jedoch veralteten

⁵<http://gene-regulation.com/pub/databases.html>, aufgerufen am 13. Juni 2021

Transfac Version⁶ mit Funktionseinschränkungen⁷ aus dem Jahr 2005 ist nach einer obligatorischen Registrierung vorgesehen (wurde beides im Rahmen dieser Dissertation nicht getestet). Das Tool kann jedoch keine gemeinsamen Module aller Sequenzen anzeigen.

11. *conTraV3*

ConTraV3 [60, 61] ist eine Webapplikation, mit der nach TFBSs gesucht werden kann. Dafür können Modellorganismen ausgewählt oder es kann eine Datei mit Fasta-Sequenzen hochgeladen werden. Matrizen werden entweder aus der darunterliegenden Datenbank ausgewählt oder durch den Anwender hochgeladen (Abbildung 1.20). Die Suche nach gemeinsamen Modulen wird allerdings nicht unterstützt.

Abbildung 1.20: *conTraV3*: In diesem Tool kann der Anwender nach TFBSs suchen — nach der Auswahl von Sequenzen (auch eigenen) und Matrizen (auch eigenen). Die Suche nach gemeinsamen Modulen wird aber nicht unterstützt. (* siehe Abbildung 1.7)

⁶<http://gene-regulation.com/pub/databases.html>, aufgerufen am 13. Juni 2021

⁷https://portal.genexplain.com/archive/documents/transfac_comparison.pdf, aufgerufen am 13. Juni 2021

1.6 Vergleich der Funktionen der betrachteten Tools

Die im letzten Abschnitt 1.5 beschriebenen Tools werden hinsichtlich der Funktionalitäten übersichtlich in der Tabelle 1.2 dargestellt. Module konnten im Rahmen dieser Dissertation bisher nur durch das Tool Genomatix gesucht und gefunden werden (Modulsuche ist auch mit TRANSFAC möglich, wurde aber hier nicht näher betrachtet). Allerdings sind dies nicht gemeinsame Module aller eingegebenen Sequenzen, sondern beziehen sich immer nur auf die jeweilige Sequenz (dies gilt auch für TRANSFAC). AI-Modules kann nach gemeinsamen Modulen aller eingegebenen Sequenzen suchen.

Tabelle 1.2: Ein Vergleich der Funktionen der Tools (conTraV3 [60], TRANSFAC [1, 52], Genomatix [16], Motifmap [8, 35, 59], Promo [13, 31, 40], ModuleMaster [56], Prodoric [32, 39], Softberry [23, 47, 53, 54], TAIR [49], PlantPan 3.0 [4, 38], TESS [43, 44, 45]) *Diese Tabelle wollen wir auch veröffentlichen (eingereicht bei BMC Bioinformatics)

Name	Unterstützt Suche nach TFBS	Hinterlegt in Datenbanken	Vorberechnete Ergebnisse	Eingabe eigener Matrizen möglich	Eingabe eigener Sequenzen möglich	Anzeige von gemeinsamen Modulen eigener Sequenzen
AIModules	ja	Matrizen	nein	ja	ja	ja
conTraV3	ja	Genome, Matrizen	nein	ja	ja	nein
TRANSFAC	ja	Genome, Promotoren, Matrizen und Module	nein	ja	ja	nein, Betrachtung der Module pro Sequenz
Genomatix	ja	Genome, Promotoren, Matrizen und Module (Vertebraten und Pflanzen)	nein	ja	ja	nein, es werden für jede Sequenz einzeln die Module angezeigt
Motifmap	ja	Genome und Matrizen	nein	nein	nein	nein
Promo	ja	Matrizen	nein	nein	ja	nein, nur gemeinsame TFBSs
ModuleMaster	Nicht lauffähig unter Linux Mint 64 bit und Windows 10 64 bit. Kein Support, da originäres Institut keine Bioinformatik mehr betreibt.					
Prodoric	ja	Genome und Matrizen	ja	nein	ja, aber nur eine Sequenz	nein
Softberry	ja	Matrizen	nein	nein	ja	nein, nur eine Sequenz kann angegeben werden
TAIR	ja, wenn TFBS in mind. 3 Sequenzen vorkommt	Matrizen	nein	nein	ja	nein, Anzeige von TFBSs, die in mind. 3 Sequenzen vorkommen; nur für einige Pflanzen; beschränkt auf TFBSs bestehend aus 6-mer
PlantPan 3.0	ja	Genome	nein	nein, aber Consensus Sequenz als IUPAC code (z.B. A[CA][CGT]G)	ja	nein, nur gemeinsame TFBSs
TESS	Nicht mehr verfügbar als Webservice. Backendcode ist verfügbar.					

1.7 Soll-Stand

Es soll ein offenes, einfach zu bedienendes Web-Tool erstellt werden, das mithilfe der JASPAR-2020-Matrizenbank [24, 34] auf DNA-Sequenzen Module vorhersagen kann. Durch eine Uracil (U) zu Thymin (T) Textersetzung haben wir dieses Tool auch für RNA-Sequenzen erweitert. Somit kann auch nach RNA-Motiven wie beispielsweise Polyadenylierungsstellen gesucht werden. Sequenzen gibt der Anwender vor. Matrizen können aus einer Liste von Taxa ausgewählt werden oder ebenso vom Anwender stammen.

Der Einfachheit halber wurde sich gegen eine Desktopapplikation entschieden und für eine Webapplikation. Eine Desktopapplikation bietet zwar die Möglichkeit unabhängig von einer Internetverbindung verwendbar zu sein, allerdings muss in diesem Falle ein Konzept entwickelt werden, wie Updates eingespielt werden sollen. Weiterhin müssen auf der Umgebung des Anwenders die Voraussetzungen dafür vorhanden sein, damit die Desktopanwendung läuft (Java-VM, .net-Framework, etc.). Die beim Anwender liegenden Ressourcen müssen den Anforderungen der Applikation genügen (Prozessor, Arbeitsspeicher, etc.). Schließlich ist die Komplexität einer Anwendung, die auf allen Umgebungen laufen soll (MacOS, Linux, Windows, etc.) nicht zu unterschätzen.

Bei all diesen Punkten ist der Betrieb der Anwendung als Webanwendung von Vorteil. Man hat eine vordefinierte Umgebung zur Verfügung und kann bei virtuellen Servern, wenn Geschwindigkeitsprobleme bestehen sollten, eine höhere Prozessor- oder Hauptspeicherstufe dazu buchen. Bei neu entwickelten Anforderungen liegt die Verantwortlichkeit der Installation der Neuerung nicht beim Endanwender, sondern kann zentral erfolgen. Allerdings benötigt der Anwender für die Nutzung eines solchen Tools eine Internetverbindung und einen Browser. Die Webapplikation wurde für den Chromium Browser optimiert.

Natürlich kann die Applikation auch auf einem Notebook zum Laufen gebracht werden, entweder durch die Installation der entsprechenden Server oder durch das Starten der Docker Container (siehe Abschnitt 2.5).

1.8 Adaptierte Tools

Zum Zeitpunkt der Implementierung waren alle benutzen Tools und Frameworks mit einer offenen Lizenz verfügbar. Diese werden in Kapitel 2 näher besprochen. Vorab sei erwähnt, dass ein Rechenkern, der nicht Module, sondern TFBSs vorhersagt aus der TESS Publikation [43, 44, 45] für dieses neue Web-Tool als Teil des Backends Verwendung findet (s. Punkt 8).

Kapitel 2

Materialien und Methoden

In diesem Teil wird Bezug genommen auf die verwendeten Technologien. Es wird gezeigt, wie die Architektur aufgebaut ist und warum sich gerade dafür entschieden wurde. Allgemein ist das Tool in drei Schichten untergliedert: Front-End, Back-End und Datenbank.

2.1 Jaspar Converter

Es wird ein in Java geschriebenes Tool bereit gestellt, das die Jaspar-Matrizen, die in einem Textformat vorliegen in ein *.dat Format konvertiert, um in die Postgres-Datenbank kopiert zu werden. Das bietet sich z. B. an, wenn eine neue Jaspar Version veröffentlicht wird. Das Tool trägt den Namen *JasparConverter* und ist im Quellcode dokumentiert.

2.2 Tool-Stack

In diesem Abschnitt werden die Technologien vorgestellt, die zu Beginn der Implementierung alle quelloffen waren.

2.2.1 Front-End

Das Front-End ist in aller Kürze das, was der Anwender zu sehen bekommt — also der Inhalt des Browsers.

Ein Browser bedient sich seiner Engines (interne Rechenkern), um aus dem, was von einem Webserver (die Quelle der Internetseite) heruntergeladen wird, eine darstellbare Form zu generieren. Vom einem Webserver können unterschiedliche Dateien gesendet werden (z. B. HTML-, CSS-, JavaScript-Dateien, etc.). Diese werden vom Browser interpretiert und zu einem Ergebnis zusammengesetzt.

Mit der zunehmenden Verbreitung des Internets haben sich Technologien entwickelt, die es ermöglichen, nicht nur statische HTML-Seiten anzuzeigen. Dies gelang unter anderem mithilfe einer Programmiersprache, die im Browser ausgeführt wird — JavaScript. Dadurch konnten dynamische Webseiten erstellt werden, die die Anzeige anpassen oder Benutzereingaben validieren.

In den letzten Jahren hat der Funktionsumfang von Webtools weiter zugenommen und es hat sich gezeigt, dass die Kommunikation zwischen Client (z. B. Browser) und dem Webserver dazu führt, dass durch Up- und Downloads zusätzliche Wartezeiten beim Anwender entstehen und der Arbeitsfluss behindert wird. Dies führte zur Entwicklung von sogenannten Single-Page-Applikationen (SPA). Hinter diesem Ausdruck steckt der Gedanke, dass der Client nur das Allernötigste mit dem Server austauschen sollte, um eine möglichst unterbrechungsfreie Nutzung des Services zu bieten. Dafür sendet der Server beim Aufruf der Seite die gesamte Applikation an den Client. Diese ist dementsprechend größer als eine reine HTML-Seite und enthält alle statischen Elemente, d. h. auch Unterseiten.

Für eine solche SPA gibt es Frameworks. Das sind allgemein gesagt Werkzeuge, die einem bei der Arbeit in einem bestimmten Bereich unterstützen. Für die Büroarbeit wäre ein solches Frame-

work z. B. OpenOffice, LibreOffice, Microsoft Word, etc. Für die Entwicklung des Front-Ends gibt es in einem SPA Umfeld mehrere Möglichkeiten. Zu Beginn der Implementierung war AngularJS [19] ein breit unterstütztes und oft verwendetes Framework, weshalb sich für dieses entschieden wurde.

2.2.2 Back-End

Das Front-End des Client sendet Anfragen an ein dahinter liegendes System (Back-End), das diese abarbeitet und beantwortet. In dieser Schicht steckt meist die Logik, die CPU- und hauptspeicherintensive Arbeiten durchführt, Verbindungen zu anderen Systemen (z. B. Datenbank) aufbaut oder Authentifizierungen prüft. Weit verbreitet sind Back-End-Systeme auf Java-Basis [36]. Diese können als sogenannte Servlets im Webcontainer eines Applikationsservers betrieben werden. Als solcher wird der Apache Tomcat [14] verwendet. Dadurch können in Java geschriebene Webapplikationen als Servlets ausgeführt werden. Die Applikation wird mittels JavaEE¹ und JAX-RS² realisiert. Das Back-End nutzt für das Auffinden von TFBSs die ausführbare Datei *tessWms* (siehe Punkt 8), die so modifiziert wurde, dass es über das JSON-Format³ kommunizieren kann.

2.2.3 Datenbank

Das Datenbanksystem soll performant sein und unter Linux betrieben werden können. Zusätzlich soll die Datenbank nativ mit dem Back-End verbunden werden können. Deshalb fiel die Entscheidung auf PostgreSQL⁴. Die Datenbank enthält Motive aus JASPAR 2020 [34].

¹Java Enterprise Edition

²Java API for RESTful Web Services

³https://de.wikipedia.org/wiki/JavaScript_Object_Notation, aufgerufen am 13. Juni 2021

⁴<https://www.postgresql.org/>, aufgerufen am 13. Juni 2021

2.3 Architektur

In diesem Abschnitt werden die verwendeten Technologien zum großen Ganzen zusammengefügt.

Im Allgemeinen wird das Front-End in einem Apache Webserver betrieben. Dieses kommuniziert mit dem Back-End (analog zum Client-Server-Modell) über das JSON⁵ Format. Wenn der Anwender Motive selbst übergeben hat, dann werden diese an tessWms weitergereicht. Wenn der Anwender ein Taxon ausgewählt hat, liest das Back-End die entsprechenden Motive aus der Datenbank und übergibt diese tessWms. Das Ergebnis wird an das Front-End zurückgeliefert, das dieses weiter aufbereitet (s. Abb. 2.1). TessWms wurde um die JSON-Kommunikation erweitert.

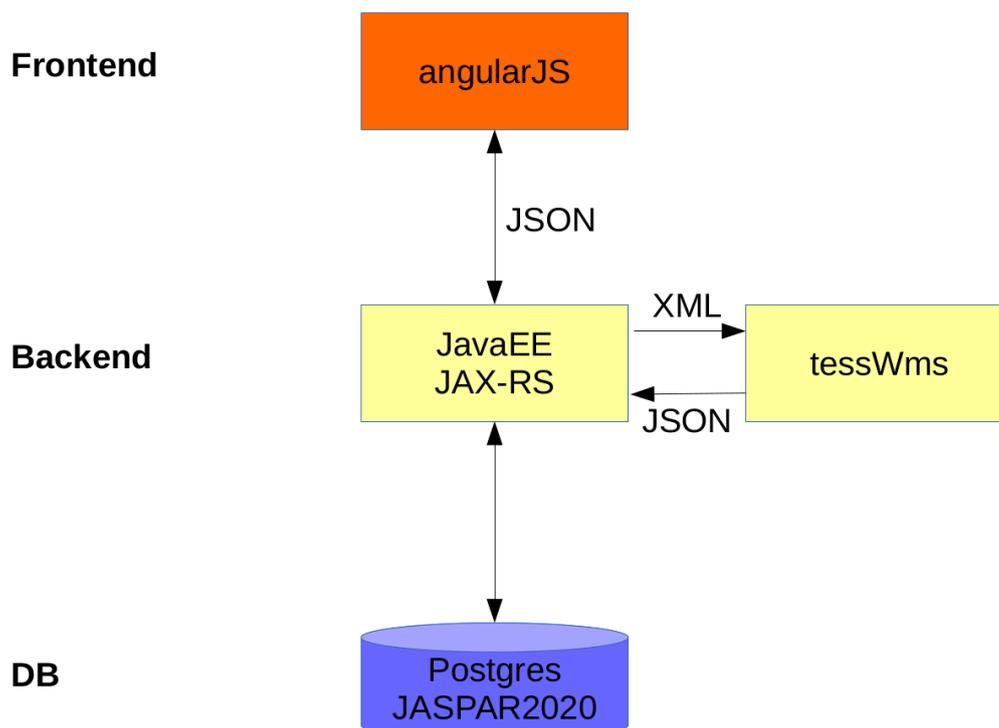


Abbildung 2.1: Die Architektur ohne Redundanz (* siehe Abbildung 1.7)

⁵JavaScript Object Notation

2.4 Skalierbarkeit

Die gesamte Applikation ist vertikal skalierbar (scale up), d. h. durch Hinzufügen von Hardwareressourcen kann die Ausführung beschleunigt werden. Dies ist aber nur begrenzt möglich, da irgendwann die beste Hardware verbaut ist. Da die Applikation kein verteiltes Rechnen unterstützt, ist sie nicht horizontal skalierbar (scale out).

Für den Fall, dass die Webanwendung in Zukunft doch hohe Lasten zu bewältigen hat, wurden alle drei Schichten der Applikation in Docker-Containern [9] mittels *docker swarm* ausgeliefert. Dies hat den Vorteil, dass mehrere Instanzen der drei Applikationsschichten auf mehrere Server verteilt werden können (siehe Abbildung 2.2). Intern übernimmt Docker die Kommunikation der Schichten und verteilt die Lasten auf die jeweiligen Instanzen. Damit kann sehr schnell auf eine hohe Last reagiert werden, indem z. B. für das rechenintensive Back-End mehr Instanzen generiert werden. Die Containervirtualisierung hat auch den Vorteil, dass die Applikation bei einem Cloud-Anbieter mittels Kubernetes betrieben werden kann, um so höheren Durchsatz und besseren Failover zu garantieren.

2.5 Entwicklerdokumentation

Hier wird eine Dokumentation erstellt, die es erlaubt die Applikation zu erweitern und zu warten. Der Quellcode wird, sobald das Manuskript akzeptiert wurde, auf der Seite <https://zenodo.org/badge/latestdoi/363702392> veröffentlicht.

2.5.1 Front-End Entwicklung

Um das Front-End zu entwickeln wurde das Node.js Projekt [7] benutzt. Diese Umgebung erlaubt es JavaScript Code außerhalb

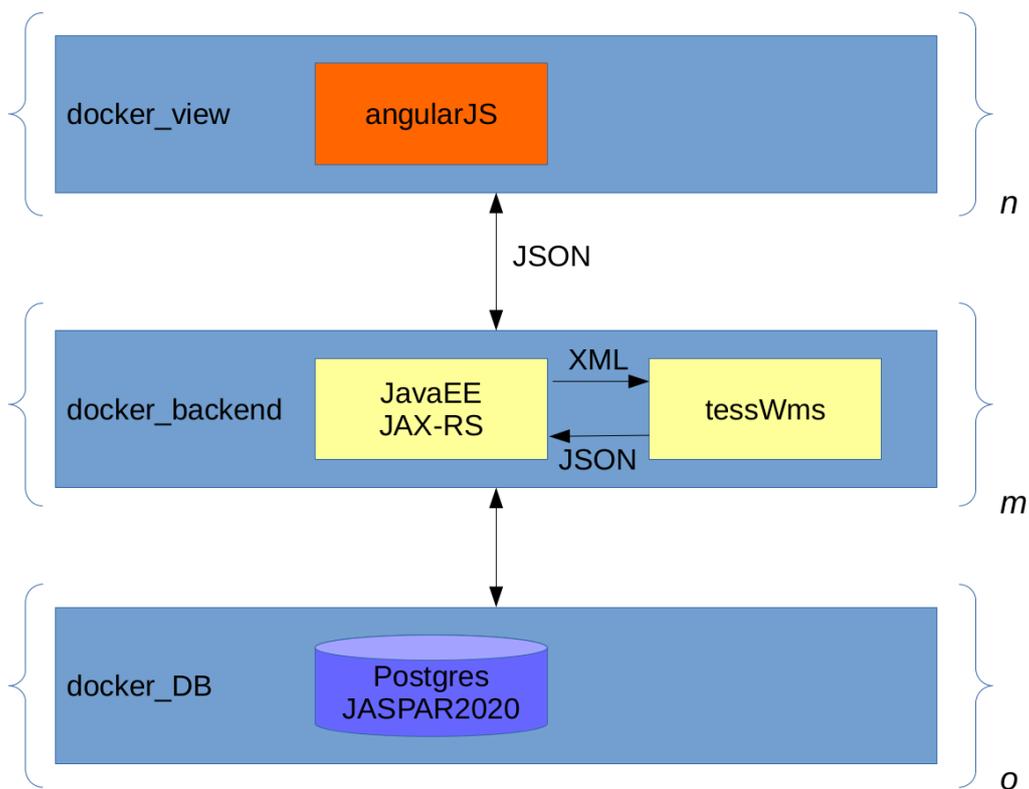


Abbildung 2.2: Die Architektur mit Redundanz: Jede Schicht kann in einer definierte Anzahl von Docker-Containern zur Verfügung gestellt werden. Die Lastverteilung geschieht automatisiert. (* siehe Abbildung 1.7)

eines Browsers auszuführen.

Der Quellcode des Frontends befindet sich im veröffentlichten Quellcode im Ordner *Sources/frontend* und bedarf einer Anpassung. Im Quellcode muss die IP des Backends hinterlegt werden. Dies erfolgt in der Datei *tools.js* in dieser Zeile:

```
$scope.callURL = 'add URL of backend';
```

Danach kann das Frontend mittels *yeoman*⁶ über den Befehl `grunt build` oder `grunt build -force` gebaut werden. Das Ergebnis wird automatisch minified (Dateien werden kleiner) und uglified (Quellcode wird maskiert) im Ordner *dist/* abgelegt. Der Inhalt dieses Ordners kann auf einen Webserver kopiert werden.

2.5.2 Back-End Entwicklung

Das Backend wurde mit Eclipse⁷ Mars und `jdk_1.8.0_74` entwickelt. Für das Abhängigkeits- und Buildmanagement wurde Maven⁸ verwendet. Das Projekt für das Backend heißt *pwm-rest*, das für den Konverter *JasparConverter*. Beide Projekte befinden sich im veröffentlichten Quellcode im Ordner *Sources/backend*. Tomcat⁹ kann über den *Eclipse Marketplace* installiert werden. Das hat den Vorteil, dass innerhalb von Eclipse ein Applikationsserver mit dem Programmcode gestartet werden kann. Dadurch werden alle gespeicherten Änderungen direkt übernommen. Weiterhin kann der Tomcat auch im Debug-Modus gestartet werden. Wenn das Backend innerhalb von Eclipse auf dem Tomcat Anfragen entgegennimmt, dann wird eine *execute-Berechtigung* für den User benötigt. Dies wird in der Konsole von Eclipse als Fehler ausgewiesen. Um das zu heilen muss auf die Datei `tessWms` der Befehl

```
$ chmod u+x tessWms
```

⁶<https://github.com/yeoman/generator-angular>, aufgerufen am 13. Juni 2021

⁷<https://www.eclipse.org/downloads/>, aufgerufen am 13. Juni 2021

⁸<https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html>, aufgerufen am 13. Juni 2021

⁹<http://tomcat.apache.org/>, aufgerufen am 13. Juni 2021

ausgeführt werden.

Das Projekt *pwm_rest* muss in Eclipse importiert werden. Danach muss die IP und die Anmeldeinformationen für die Datenbank in der Datei *persistence.xml* hinterlegt werden:

```
<property name="javax.persistence.jdbc.url"
  value="jdbc:postgresql://localhost:5432/docker" />
<property name="javax.persistence.jdbc.user"
  value="user" />
<property name="javax.persistence.jdbc.password"
  value="password" />
```

In Eclipse kann mittels des Maven-Befehls `mvn clean install` eine *.war-Datei* generiert werden. Diese wird dann auf dem Tomcat Application Server deployed.

2.5.3 Datenbank Entwicklung

Die postgres-DB¹⁰ kann mittels *pg-admin* administriert werden. In diese Datenbank können die mit dem JasperConverter generierten *.dat-Dateien* per *copy-Befehl* kopiert werden. Diese Dateien befinden sich auch im veröffentlichten Quellcode im Ordner *Sources/postgres*. Hier ist auch ein Backup der Datenbank vorhanden, das über *pg-admin* wiederhergestellt werden kann. Das Datenbank-Backup sowie die *dat-Dateien* enthalten die Matrizen aus Jasper 2020.

2.5.4 Serverumgebung installieren

Die *Builds* aus *grunt* für das Frontend und aus *Maven* für das Backend können wie gewohnt auf Web- sowie Applikationsservern deployed werden. Die postgres-DB kann auf dem gleichen Server

¹⁰<https://www.postgresql.org/>, aufgerufen am 13. Juni 2021

installiert werden, um die Applikation lauffähig zu bekommen. Dies kann natürlich auch auf dem eigenen Notebook oder PC erfolgen, sodass die Applikation auch ohne Internetverbindung und lokal zur Verfügung steht.

2.5.5 Docker

Die Anwendung ist auch in einer Docker¹¹ und einer *docker swarm*¹² Umgebung lauffähig. Zuallererst müssen die Images erstellt werden. Diese stellen eine Blaupause dar und müssen zuerst gebaut werden. Wenn die Images gestartet sind, spricht man von einem *docker-Container*.

Dazu muss im Ordner *Sources/docker/container/backend* des veröffentlichten Quellcodes

```
# docker build -t scorepwm_backend .
```

, im Ordner *Sources/docker/container/frontend*

```
# docker build -t scorepwm_frontend .
```

und im Ordner *Sources/docker/container/postgres-db*

```
# docker build -t eg_postgresql .
```

ausgeführt werden. Die jeweiligen Vorgehensweisen stehen auch in den Ordnern in den Dateien *use.txt*. Durch die *docker build* Befehle werden die Konfigurationsinformationen in der jeweiligen *Dockerfile* interpretiert und ausgeführt.

Nachdem die *docker-Images* erstellt wurden, können die *docker-Container* gestartet werden.

1. Entweder einzeln:

Dazu muss im Ordner *Sources/docker/container/backend*

¹¹<https://www.docker.com/>, aufgerufen am 13. Juni 2021

¹²<https://docs.docker.com/engine/swarm/>, aufgerufen am 13. Juni 2021

```
# docker run -v /tmp:/tmp \  
-v /tmp/tomcat-logs/:/usr/share/tomcat8/logs/ \  
-v /tmp/tomcat-conf/:/usr/share/tomcat8/conf/ \  
-d -p 8080:8080 scorepwm_backend
```

, im Ordner *Sources/docker/container/frontend*

```
# docker run -v /tmp:/tmp -d -p 80:80 \  
scorepwm_frontend
```

und im Ordner *Sources/docker/container/postgres-db*

```
# docker run -p 5432:5432 -P -d eg_postgresql
```

ausgeführt werden. Dies führt dazu, dass es von allen Images einen Container gibt, die miteinander kommunizieren. Die Befehle sind auch in den Ordnern in der Datei *use.txt* dokumentiert.

2. Oder als *docker-swarm*:

Die generierten Images können im Ordner *Sources/docker/* konzertiert gestartet werden. Dazu verwendet man *docker-swarm* und eine *yml-Konfigurationsdatei*. Die Konfigurationsdatei im Ordner *Sources/docker/* heißt *docker-compose.yml*. Hierin sind die Imagenamen vermerkt, nebst *Loadbalancing-Regeln*, die Anzahl der Container pro Image, etc.

Gestartet wird der *docker-swarm* über

```
# docker stack deploy -c docker-compose.yml modules
```

, wobei *modules* der Name des Stacks ist, der in dem *docker-swarm* deployed wurde. Um weitere *nodes* (Rechenknoten) in den Schwarm aufzunehmen, muss zuerst ein Hauptknoten definiert werden.

```
# docker swarm init
```

Danach kann ein weiterer Knoten (Rechner, Server, etc.), der eine Netzwerkverbindung zum Hauptknoten aufbauen kann, in den Schwarm aufgenommen werden:

```
# docker swarm init --advertise-addr <myvm1 ip>
```

Die Verteilung der Container auf die Knoten im Schwarm geschieht automatisiert im Hintergrund anhand der Regeln in der *docker-compose.yml*-Datei.

Unabhängig davon, ob nur je ein Container pro Image gestartet oder ein Schwarm initialisiert wird, kann geprüft werden, ob die einzelnen Schichten der Architektur regelrecht laufen. Um die Funktion des Frontends zu prüfen navigiert der Browser auf die URL `http://localhost`. Das Backend kann geprüft werden über die URL `http://localhost:8080/api/areyoualive`. Eine aktive Instanz der Datenbank kann über die Verbindung mit dem Tool *pg-admin*¹³ geprüft werden.

2.6 Parameter AIModules und Genomatix

Über die Parameter *La* und *Ld* kann man im Frontend von AIModules die Qualität der zu findenden TFBSs einstellen. Dabei definiert *La* die minimale Log-Wahrscheinlichkeit und *Ld* das maximale Wahrscheinlichkeitsdefizit. Dies errechnet sich aus der Differenz zwischen der maximalen und der minimalen Log-Wahrscheinlichkeit ($Lm - La$). *Lm* stimmt somit mit der Consensus Sequenz überein, wobei jede Base bis zu einem Wert von zwei zum Ergebnis beisteuern kann. Somit entspricht der höchste Wert für *Lm* der Consensus Sequenz einer TFBS und das heißt, dass *Ld* angibt, wie weit *La* einer TFBSs zu dessen *Lm* abweicht [43, 45].

¹³<https://www.pgadmin.org/>, aufgerufen am 13. Juni 2021

Für MatInspector des Produkts Genomatix gibt es dagegen die Parameter `Optimized` und `0,75`. Eine TFBSs, die der Consensus Sequenz entspricht, bekommt den Score von `1,00`. `Optimized` in diesem Zusammenhang bedeutet, dass die Bindestelle valide ist, wenn der Score größer gleich `0,75` entspricht [15].

Kapitel 3

Ergebnisse

3.1 Herausforderungen und Zielsetzungen

Promotoren sind zentral für die biologische und pathophysiologische Regulation z. B. bei bei Anpassung an Stress-Situationen [12, 22], bei normalem Zellwachstum, Zelldifferenzierung [27, 29] und Krebs [21]. Um die involvierten TFBSs und Module zu identifizieren wird ein Werkzeug benötigt, das dies einfach und zuverlässig bewerkstelligt. Zudem können über ein solches Tool Stammbäume und phylogenetische Analysen unterstützt werden. Außerdem wird der Analyseumfang der Motive auf regulatorische RNA-Motive erweitert. Dieser Funktionsumfang wird in diesem Kapitel definiert und in Kapitel 4 näher betrachtet.

Zu Beginn der Promotion galt es zu definieren wie der Umfang der Applikation auszusehen hat und grob zu skizzieren wie diese Anforderungen umgesetzt werden können. Es galt eine Anwendung zu erstellen, die einen für Anwender attraktiven Funktionsumfang bietet und auch zukunftsfähig ist. Dies deshalb, da wir das Ziel verfolgten nicht nur eine Applikation zu entwickeln die für sich selbst steht und am Ende der Promotion ihren Soll erfüllt haben sollte, sondern es war ins Auge gefasst worden ein sogenanntes **Framework**, i. e. ein Grundgerüst, zu schaffen, dass die Weiterentwicklung und auch Wartung möglich macht – einerseits durch

moderne Technologien und andererseits durch eine klare Struktur der Architektur und auch des Programmierstils. Ein positiver Nebeneffekt dieser Idee war auch, dass nicht Unmengen an Zeit aufgewandt werden muss, um zu verstehen, was der Programmcode macht und deshalb lag es nahe, dass zukünftig im Rahmen von Praktika oder Bachelor-/Master-Arbeiten sowie Dissertationen eine Weiterentwicklung möglich ist.

Dafür war es auch wichtig Ideen zu entwickeln, die in Zukunft attraktive Funktionen ermöglichen, um die Applikation über längere Zeit aktiv zu halten. Diese sollen nicht nur durch spannende neue Funktionen die Anwendung aktuell halten, sondern sie zielen auch darauf ab Veröffentlichungen zu produzieren und dadurch die Motivation an dem Tool hoch zu halten. Dazu zählen beispielsweise, dass wir einen `JasparConverter` (siehe Abschnitt 2.1) entwickelt haben, der es automatisiert ermöglicht neue Versionen der Jaspar Matrizen [34] in ein Format zu konvertieren, dass in die AIModules Datenbank importiert werden kann. Weiterhin ist das `Frontend`, also das, was im Browser dargestellt wird, in der Programmiersprache `Javascript` geschrieben. Dies macht es mittels des Tools `nodejs`¹ möglich, diesen Programmcode recht einfach auf den Server zu verlagern und mit entsprechender Hardware ausgestattet, die Geschwindigkeit der Berechnungen zu erhöhen.

Zudem ist für den Anfang die Berechnungsgeschwindigkeit dadurch eingeschränkt, dass alle Komponenten der Anwendung AIModules auf einem virtuellen Server mit einer mäßigen (virtuellen) Hardwareausstattung laufen. Im Rahmen der Entwicklungen haben wir dieses Szenario allerdings schon betrachtet, sodass alle Komponenten mittels Lastverteilung einzeln auf mehrere Instanzen verteilt Berechnungen übernehmen können. Dies haben wir mittels `docker`² und `docker swarm`³ realisiert (siehe Abschnitt

¹<https://nodejs.org/en/>, Zugriff am 26. Juni 2021

²<https://www.docker.com/>, Zugriff am 26. Juni 2021

³<https://docs.docker.com/engine/swarm/>, Zugriff am 26. Juni 2021

2.4). Die Konfigurationsdateien dafür können mit mäßigem Aufwand sogar für **Kubernetes**⁴ angepasst werden, sodass die Applikation auch auf großen Cloud Providern zur Verfügung gestellt werden kann, um die Verfügbarkeit und auch die Berechnungsgeschwindigkeit zu erhöhen. Weitere Erweiterungsmöglichkeiten sind im Abschnitt 4.7 ausgeführt.

Diese Betrachtungen über die Zukunft der Applikation bauten auf dem Funktionsumfang der als fachliche Anforderungen definiert wurde. Diese waren einerseits Transkriptionsfaktorbindstellen (TFBS) und Module auf DNA vorherzusagen. Die Motivation für ein Tool, das die Modulesuche beherrscht, nährte sich auch dadurch, dass nur kommerzielle Services aktuelle Modulesuchen anboten (siehe auch Abschnitt 1.3) und wir durch unsere Anwendung eine Nachfrage hiernach zu befriedigen erhofften, umso mehr, da die Anwendung frei verfügbar und kostenlos sein sollte. Für die TFBS-Suche bot sich **tessWms** [43] an, das in der Programmiersprache **C** entwickelt wurde und somit mit nativer Geschwindigkeit Berechnungen ausführte.

Als nächstes haben wir uns damit auseinandergesetzt hochwertige Matrizen für die Analyse zu beschaffen. Da die öffentlich verfügbare Version von **TRANSFAC** [1, 52] online nur veraltete Matrizen aus 2005 anbietet und ein Download nicht ohne Weiteres möglich ist, haben wir uns für die Matrizen aus der **Jaspar** [34] Datenbank entschieden. Diese hatten den Vorteil, dass sie Aktualisierungen erfahren, sodass im Rahmen der Entwicklung von **AIModules** die **Jaspar** Datenbank in der Version 2018 [10] eingearbeitet wurde, um nach der Veröffentlichung der **Jaspar DB 2020** [34] die Matrizen in **AIModules** zu aktualisieren. Es ist zu erwarten, dass in Zukunft **Jaspar** weiterhin Aktualisierungen erfährt, sodass **AIModules** immer die neuesten Matrizen zur Verfügung stellen kann.

Nach der Betrachtung der Matrizen haben wir uns der Archi-

⁴<https://kubernetes.io/de/>, Zugriff am 26. Juni 2021

tektur und des Designs angenommen. Entschieden haben wir uns für eine Drei-Schichten-Architektur (siehe Abschnitt 2.2), bei der die drei Schichten nur lose miteinander gekoppelt sind, sodass alle drei Schichten ohne viel Aufwand durch andere oder neuere Technologien ausgetauscht werden können. Wie erwähnt haben wir uns durchweg für Open-Source Komponenten entschieden und AIModules auch unter einer Open-Source Lizenz (*GPLv2*⁵) veröffentlicht. Das Design der Komponentenschnittstellen orientiert sich an standardisierten Schnittstellen, um die lose Kopplung der Schichten zu unterstützen.

Unsere Entscheidung, die Anwendung möglichst anwenderfreundlich zu machen, führte zu einer Anwendungsoberfläche, die aufgeräumt ist und nur die nötigsten Informationen bietet (siehe Abbildungen 3.2 und 3.3). Erweiterte Informationen und zusätzliche Tipps zur Anwendung bekommt der Anwender dadurch, dass er den Mauszeiger über die einzelnen Elemente platziert (*mouseover*). Die Matrizen, die in der Datenbank gespeichert sind, können bequem über die Oberfläche ausgewählt werden (siehe Abbildung 3.4). Um das Ergebnis, das die TFBSs und Module enthält, ansprechend anzuzeigen, werden diese farblich als Bilddatei auf der Webseite eingebettet (siehe Abbildungen 3.5 und 3.7). Dieses Bild und die dazugehörige Legende über die Transkriptionsfaktoren (TFs) lassen sich bequem herunterladen. Außerdem haben wir dem Anwender eine Möglichkeit gegeben, nur die TFs anzuzeigen, die er auch z. B. in einer Veröffentlichung hervorheben möchte. Weiterhin haben wir eine Übersicht implementiert, die das Ergebnis als Tabelle und als Excel Datei enthält (siehe Abbildungen 3.6 und 3.8). Diese Datei kann der Anwender herunterladen, um erweiterte Details zu den einzelnen Funden zu erhalten. Möchte der Anwender eigene Matrizen aus alignierten Sequenzen erzeugen und verwenden, so bietet unsere Anwendung auch dafür eine

⁵<https://www.gnu.org/licenses/old-licenses/gpl-2.0.html>, Zugriff am 26. Juni 2021

Lösung. Wir haben auf der Webseite einen Converter hinterlegt, der genau das macht. Der Anwender gibt dazu die Sequenzen in den Converter ein (siehe Abbildung 3.9) und erhält eine Matrize (siehe Abbildung 3.10), die in AIModules verwendet werden kann. Darüberhinaus haben wir Buttons auf der Website hinterlegt, die die Funktionalität der Applikation demonstrieren sollen. Dadurch kann der Anwender mit einem Mausklick die TFBS-, Module- und RNA-Motifsuche durchführen (siehe Abbildung 3.2 1) und 3)). Um unsere Applikation auch für die RNA-Motivesuche zu erweitern, haben wir eine Textersetzung implementiert, die Uracil nach Thymin konvertiert.

Die Funktionalität unseres Tools und vor allem die Ergebnisse daraus werden in den folgenden Abschnitten näher betrachtet. Dabei erfolgen auch Vergleiche mit anderen Anwendungen, die TFBSs und Module vorhersagen können. Alle verwendeten Sequenzen sind im Anhang 5 zu finden.

3.2 AIModules

Wir haben versucht die Benutzeroberfläche des Tools nicht zu überladen, damit sich jeder neue Anwender sofort zurecht findet. Deshalb werden auf der Hauptseite nur die allerwichtigsten Punkte angezeigt. Beim Anklicken von Checkboxen oder Radiobuttons wird die Sicht erweitert und der Anwender erhält Zugriff auf weitere Funktionalitäten. Zusätzlich wurden die meisten Controls mit einem Mouse-over Hinweis ausgestattet.

Da die Berechnungen auf Serverseite sehr CPU- und arbeitsspeicherintensiv sind, und das Rendering und die Modulesuche auf Clientseite ähnlichen Lasten unterliegen, bietet es sich für die Modulesuche an, iterativ vorzugehen:

1. Mit wenigen Sequenzen nach TFBSs suchen.
2. Die interessantesten Matrizen aus dem Tool herauskopieren.

3. Sollte eine Matrize nicht im Ergebnis vorhanden sein, evtl. über L_a und L_d feinjustieren, indem L_a erniedrigt und L_d erhöht wird.
4. Erst danach für die gegebenen Sequenzen und Matrizen die Modulesuche durchführen.

3.2.1 Hauptfenster

Auf der Benutzeroberfläche von *AIModules* unter *Tool* (Abbildung 3.1) erhält der Anwender Zugriff auf die TFBS- und Modulesuche (Abbildungen 3.2, 3.3, 3.4).



Abbildung 3.1: Titelmanü von *AIModules*

Wenn alle Sequenzen und Matrizen die eingebaute Validitätsprüfung bestehen, wird der *Score-Button* sichtbar. Betätigt der Anwender diesen Button, wird der Request an den Server geschickt und die Response auf den Client gerendert. Im Falle der TFBS Suche wird die Sicht um Abbildung 3.5 erweitert.

Search TFBSs and Modules in DNA-Sequences

AIModules enables you to find transcription factor binding sites (TFBSs) and modules on DNA using own matrices or the JASPAR 2020 database.

Select Data Source for DNA Sequence.

Own Sequences

Input here DNA-Sequence in Fasta-Format:
 > sequence_name
 aagtccc

1) Demo 1 (TFBS search) Demo 2 (TFBS search) Demo 3 (Module search)
 Demo 4.1 (Poly-A module search) Demo 4.2 (Poly-A TFBS search) Interleukin Module Search
 Cathepsin TFBS search Interleukin-10 TFBS search Interleukin-10 module search
 Homo sapiens IGH, IGKC, PIGR module search

2) Input DNA-Sequence in Fasta-Format here.

Please type DNA sequence.
 Input DNA-Sequence!

More demanding search to build cathepsin modules (result from server ca. 0.5 sec; module search ca. 150 sec; rendering ca. 0,1 sec on Intel-i7):

3) Cathepsin module search

Minimum Score Threshold (La: log likelihood ratio score)

4) 7

5) Maximum log likelihood Deficit (Ld)

8

Abbildung 3.2: TFBS- und Modulesuche von AIModules: Die Roten Zahlen sind nur zur Orientierung eingefügt. Die meisten Controls bieten eine mouse-over Funktionalität. 1) Demo-Suchen die die Funktionalität des Tools zeigen. 2) Der Anwender kann unter diesem Punkt eigene Sequenzen im *Fasta-Format* eingeben. Uracil (U) wird durch Thymin (T) ersetzt. 3) Eine komplexere Suche, die ca. 2,5 min dauert. 4) Hier wird der *log likelihood ratio score* *La* eingegeben. Wenn der korrespondierende Score einer Matrize (einer Bindestelle) über diesem Wert liegt, wird die Bindestelle im Ergebnis festgehalten. Der Suchraum kann erweitert werden, indem der Button *Set Low Score* betätigt wird. Dadurch werden zwar mehr Treffer im Ergebnis berücksichtigt, aber die Suche wird langsamer und unspezifischer. 5) Dieser Wert gibt den *maximum log likelihood deficit* *Ld* an, einen Wert, mit dem die obere Grenze für valide Ergebnisse definiert werden kann. Diesen Wert kann man als Vergleichswert zur Consensussequenz sehen, *i.e.* je kleiner dieser Wert, desto ähnlicher ist die Bindestelle einer Consensussequenz einer funktionalen Bindestelle. In der nächsten Abbildung werden die weiteren Felder der Applikation beschrieben.

6) Yes

Hint:
First search for TFBSs without the module filtering functionality but with a low Score and Matrices from Database.
Then copy needed Matrices and do a modules search with these matrices.

7) Threshold for a TF to be considered for module filtering (see above under 2.):

0 (the lower the threshold the longer the calculation will take)

*If scoring/redering should take too long, please restart and try the following:
Leave threshold for module filtering unchanged and/or
increase La and/or
decrease Ld and/or
try with fewer sequences*

Select Data Source for Scoring.

8) Jaspar 2020 Database
 Own Matrices

Input Matrix formatted like (A C G T). Please replace Uracil (U) with Thymine (T):

```
> matrix_name
0 1 2 3
4 5 6 7
```

9) Human Poly(A)*

*Bioinformatics. 2013 Jul 1; 29(13): i316–i325. Published online 2013 Jun 19. doi: 10.1093/bioinformatics/btt218

10)

Please insert matrices.

**Please input matrix.
Input Matrix!**

Abbildung 3.3: Weiter mit AIModules: 6) Mit dieser Checkbox wird die Modulsuchefunktion aktiviert. Wenn nur nach TFBSs gesucht werden soll, darf diese Checkbox nicht aktiviert werden. 7) Für den Fall einer Modulsuche wird dieser Stepper sichtbar. Der Stepper nimmt Werte zwischen Null und der unter Abbildung 3.2 2) eingegebene Anzahl an Fasta-Sequenzen an. Mit diesem Parameter kann der Anwender bestimmen, in wie vielen Sequenzen ein TF vorkommen muss, damit dieser in der Folge in die Modulsuche mit eingehen darf. Standardmäßig ist dieser Wert auf die maximale Anzahl an Sequenzen gestellt. 8) Hinter dem ersten Radiobutton verbirgt sich eine Auswahl an gespeicherten Matrizen. 9) Eine Beispielmatrix für die Suche nach RNA-Motiven, hier humane Poly-(A)-Sequenz, kann selektiert werden. 10) Wählt der Anwender den zweiten Radiobutton, so kann er selbst definierte Matrizen im Fasta-Format eingeben.

Select Data Source for Scoring.

Jaspar 2020 Database
 Own Matrices

Please select Matrices from Dropdown:

▼

ALL

FUNGI

INSECTS

NEMATODES

PLANTS

UROCHORDATES

VERTEBRATES

Abbildung 3.4: Matrizen können auch aus der hinterlegten Datenbank selektiert werden. Die Matrizen stammen aus der Jaspar 2020 Datenbank und sind nach Taxa sortiert.

Bei Betätigen des *Score-Buttons* wird dieser deaktiviert, und es startet ein Timer und eine Animation. Der Timer stoppt, wenn die Response vom Server angekommen ist und die Animation stoppt, wenn das Rendering erledigt ist. Wenn entweder der Timer oder die Animation zu lange laufen, dann dauert die Berechnung zu lange. In diesem Fall muss die Applikation zurückgesetzt und ein abgewandelter Request über den *Score-Button* ein weiteres Mal abgesetzt werden. Diesen abgewandelten Request bekommt man dadurch zustande, dass entweder weniger Sequenzen eingegeben werden oder La erhöht oder Ld erniedrigt wird oder weniger Matrizen ausgewählt werden oder eine Kombination aus diesen Punkten. Das Ziel ist die Abfrage schlanker zu gestalten, damit die Berechnungen der Ergebnisse weniger lange dauern. Wir haben uns für das Rendern auf der Anwenderseite im Browser entschieden und gegen ein serverseitiges Rendern, um die Lasten auf dem Server zu verringern. Klickt man auf den Button *TFBS* in Abbildung 3.5, erhält man die Tabelle in Abbildung 3.6. Die Tabelle lässt sich

default: pseudocounts=1; probabilitiesOfEachBase=uniform

1)

Result from server was retrieved in 0.401 sec
The next calculations are done on the client: module search and rendering
Rendering done in 0.3709999918937683 sec

2) demo2.1
1 ————— 279
demo2.2
1 ————— 280

3) MA1100.2 ASCL1 MA1468.1 ATOH7
 MA1472.1 BHLHA15(var.2) MA0471.2 E2F6
 MA0598.3 EHF MA0473.3 ELF1
 MA0640.2 ELF3 MA0136.2 ELF5
 MA0076.2 ELK4 MA0474.2 ERG
 MA0098.3 ETS1 MA0761.2 ETV1
 MA0764.2 ETV4 MA0765.2 ETV5

4)

5)

6)

Optimized for Chromium

Abbildung 3.5: TFBS Suchergebnis: Die Roten Zahlen dienen nur der Orientierung. 1) Nach dem Absenden des Requests wird der *Score-Button* deaktiviert. Darunter wird ein Timer gestartet, der erst stoppt, wenn die Response angekommen ist. Neben dem Timer läuft parallel eine Animation. Diese stoppt erst, nachdem nach Empfang der Response das Redering beendet wurde. Die vergangene Zeit in den Einzelschritten wird unter dem *Score Button* ausgegeben 2) Das Ergebnis wird in einen SVG-Container gerendert und zeigt die Sequenznamen an. Unter jedem Sequenznamen befindet sich eine schwarze Linie, die die Sequenz in voller Länge angibt. Oberhalb der Linie werden die Bindestellen für den (+)-Strang und unterhalb die des (-)-Strangs visualisiert. 3) In diesem Bereich werden die Matrizen angezeigt, die zu einem Ergebnis geführt haben. Der Font der jeweiligen Matrizen korreliert in der Farbe [2] mit den jeweiligen Bindestellen in 2). Neben jedem Matrizenname befindet sich ein *C-Button*, mittels welchem die Matrize in die Zwischenablage kopiert werden kann. 4) Die Container in 2) und 3) können als SVG-Datei über den Button *Download SVG* heruntergeladen werden. Mit den Buttons *Select All* und *Deselect All* können alle Checkboxes in 3) an- oder abgewählt werden, was sich direkt auf die Bindestellen in 2) auswirkt. Die Matrizen können auch einzeln an- und abgewählt werden. 5) Über den *TFBS-Button* wird eine weitere Sicht geöffnet, die tabellarisch die gefundenen Bindestellen darstellt. Diese Tabelle kann in dieser Sicht als *Excel-Datei* heruntergeladen werden. 6) Zum Start einer neuen Analyse muss die Applikation über den *RESTART-Button* zurückgesetzt werden.

als Excel-Datei über den Button *Export Result* herunterladen.

Export Result											
Sequence Name	Sequence Length	Matrix Title	Matrix Length	Hit No.	Hit Sense	Hit Start	Hit Stop	Hit Score (La)	Hit Max log likelihood ratio score (Lm) or matrix possible	Difference (Ld) (maxscore(Lm) - score(La))	Hit Oligo
demo2.1	279	MA0081.1 SPIB	7	1	N	130	136	10.432293	11.208253	0.775960	TGAGGAA
		MA0087.1 Sox5	7	1	N	232	238	7.710760	11.864565	4.153805	AGTGTTT
		MA0152.1 NFATC2	7	1	N	235	241	6.682159	12.341468	5.659310	GTTTCCC
		MA0597.1 THAP1	9	1	N	138	146	6.077459	11.314089	5.236631	GTGCCCCAG
		MA0624.1 NFATC1	10	1	N	234	243	6.284671	12.519140	6.234469	TGTTTCCCCG
		MA0625.1 NFATC3	10	1	N	234	243	6.070070	13.675168	7.605098	TGTTTCCCCG
		MA0642.1 EN2	10	1	R	113	122	6.841934	12.854892	6.012957	GCTGATCAGG
		MA0658.1 LHX6	10	1	N	113	122	7.646390	14.813952	7.167562	GCTGATCAGG
				2	R	113	122	7.820948	14.813952	6.993004	GCTGATCAGG
		MA0671.1 NFIX	9	1	N	137	145	6.932735	10.843204	3.910469	GGTGCCAC
		MA0090.2 TEAD1	10	1	R	131	140	9.188079	13.718155	4.528077	GAGGAAGGTG

Abbildung 3.6: Die Details zu den gefundenen TFBSs werden tabellarisch angezeigt (hier nur ein Auszug). Diese Informationen können als Excel Datei über den Button *Export Result* heruntergeladen werden. (* siehe Abbildung 1.7)

Sind aus der TFBS-Suche Matrizen hervorgegangen, die einer Modulesuche unterzogen werden sollen, so ist die Checkbox aus Abbildung 3.3 6) zu aktivieren. Der Anwender kann entscheiden, in wie vielen der eingegebenen Sequenzen ein TF vorkommen muss, damit dieser für die Modulesuche berücksichtigt wird (Abbildung 3.3 7)). Ein Modul ist in *AIModules* eine Anordnung aus zwei TFBSs mit einem Abstand, wobei die Modulesuche sich an einem festen Algorithmus orientiert:

1. Die Bindestelle muss auf *n-Sequenzen* vorkommen. Dies definiert der Anwender unter Abbildung 3.3 7). In diesem Schritt wird auch berücksichtigt, auf welchem Strang die Bindestelle liegt.
2. Module dürfen sich von Anfang bis Ende nur um +/- 200 bp unterscheiden.
3. Ein solches Modul muss auf mindestens zwei der eingegebenen Sequenzen vorkommen. Auch hierbei wird die Strangzugehörigkeit berücksichtigt.

Die Antwort des Servers aus der Modulesuche mit denselben Sequenzen wie aus Abbildung 3.5 wird gerendert und als Ergebnis erhält der Anwender Abbildung 3.7. Diese ist der Abbildung 3.5 nicht unähnlich. Der Unterschied ist, dass aufgrund des Parameters für die Modulesuche (s. Abbildung 3.3 7)) viel weniger Bindestellen gefunden werden. Zusätzlich ist ein weiterer Button sichtbar — *Modules*. Wird auf diesen geklickt erhält man eine tabellarische Darstellung der putativen Module (Abbildung 3.8).

Die Details der Modulesuche zeigen die Module in der Form *TF1::TF2* für jede Sequenz an. Jede Zeile enthält als Zusatzinformation den Start und das Ende des Moduls, *i.e.* den *5'-Beginn* von *TF1* und das *3'-Ende* von *TF2* — zusätzlich auch die Orientierung des Strangs auf dem sich das Modul befindet.

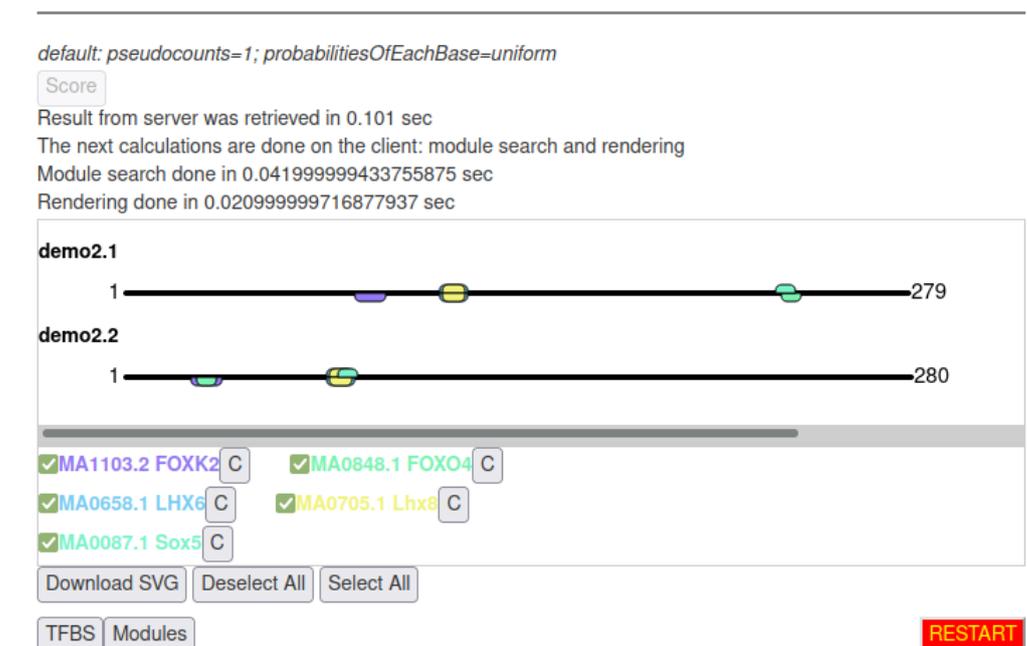


Abbildung 3.7: Gerendertes Ergebnis aus der Modulesuche. Diese Sicht ist mit Abbildung 3.5 vergleichbar, allerdings ist die Menge an gefundenen Bindestellen geringer. Zusätzlich ist ein *Modules-Button* hinzugekommen, über den eine tabellarische Sicht der putativen Module geöffnet werden kann.

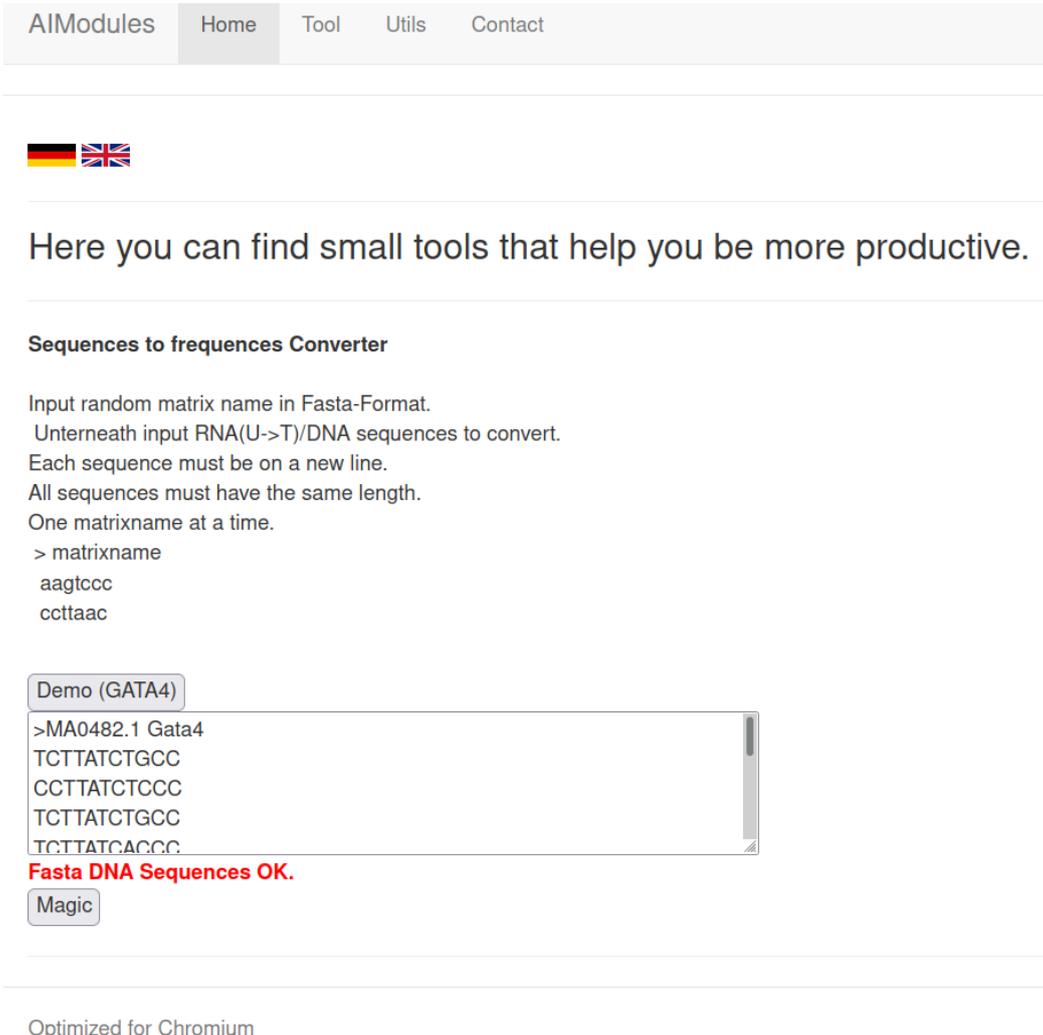
Export Result

Sequence Name	Module	Module Start	Module End	Sense
demo2.1	MA0658.1 LHX6::MA0705.1 Lhx8	113	121	N
	MA0658.1 LHX6::MA0087.1 Sox5	113	238	N
	MA0705.1 Lhx8::MA0087.1 Sox5	114	238	N
	MA1103.2 FOXK2::MA0658.1 LHX6	83	122	R
	MA1103.2 FOXK2::MA0705.1 Lhx8	83	121	R
	MA0658.1 LHX6::MA0705.1 Lhx8	113	121	R
	MA1103.2 FOXK2::MA0848.1 FOXO4	83	240	R
demo2.2	MA0658.1 LHX6::MA0705.1 Lhx8	73	81	N
	MA0658.1 LHX6::MA0087.1 Sox5	73	83	N
	MA0705.1 Lhx8::MA0087.1 Sox5	74	83	N
	MA1103.2 FOXK2::MA0848.1 FOXO4	25	33	R
	MA1103.2 FOXK2::MA0658.1 LHX6	25	82	R
	MA1103.2 FOXK2::MA0705.1 Lhx8	25	81	R
	MA0658.1 LHX6::MA0705.1 Lhx8	73	81	R

Abbildung 3.8: Die Details zu den gefundenen Modulen werden tabellarisch angezeigt. Diese Informationen können als Excel Datei über den Button *Export Result* heruntergeladen werden.

3.2.2 DNA zu Matrix Konverter

Im Titelmanü unter dem Punkt *Utils* (siehe Abbildung 3.1) befindet sich ein Matrix Konverter, der es erlaubt, homologe Sequenzen der gleichen Länge in eine Matrize zu konvertieren, die direkt in die Anwendung kopiert werden kann (siehe Abbildung 3.9, 3.10).



The screenshot shows a web application interface for a 'Sequences to frequencies Converter'. At the top, there is a navigation bar with links for 'AIModules', 'Home', 'Tool', 'Utils', and 'Contact'. Below the navigation bar, there are two language selection icons: the German flag and the UK flag. A main heading reads 'Here you can find small tools that help you be more productive.' The tool's title is 'Sequences to frequencies Converter'. The instructions state: 'Input random matrix name in Fasta-Format. Unterneath input RNA(U->T)/DNA sequences to convert. Each sequence must be on a new line. All sequences must have the same length. One matrixname at a time.' An example input is shown: '> matrixname', 'aagtccc', and 'cctaac'. A 'Demo (GATA4)' button is present. Below it, a text area contains the following Fasta format input: '>MA0482.1 Gata4', 'TCTTATCTGCC', 'CCTTATCTCCC', 'TCTTATCTGCC', and 'TCTTATCAGCC'. A red message below the text area says 'Fasta DNA Sequences OK.'. A 'Magic' button is located at the bottom of the text area. At the very bottom of the page, it says 'Optimized for Chromium'.

Abbildung 3.9: Der Matrix Konverter erlaubt es homologe Sequenzen derselben Länge in eine Matrize zu konvertieren, die in der Anwendung verwendet werden kann.

```
Result:
[A, C, G, T]
>MA0482.1 Gata4
0 2 0 8
0 1 0 0 0
0 0 0 1 0
0 0 0 1 0
1 0 0 0 0
0 0 0 1 0
0 1 0 0 0
2 0 0 8
0 4 6 0
0 1 0 0 0
0 1 0 0 0
```

Abbildung 3.10: Das Ergebnis des Matrix Konverters: Die Matrize ist im Fasta Format und kann direkt in die Anwendung kopiert und verwendet werden. (* siehe Abbildung 1.7)

3.3 Poly(A)-Suche

Bei der Eingabe der Matrizen kann sich der Anwender zwischen den JASPAR 2020-Matrizen (siehe Abbildung 3.4) oder eigenen Matrizen (siehe Abbildung 3.3 8)–10)) entscheiden. Dabei bietet *AIModules* auch die Funktionalität mittels Matrizen des Anwenders nach RNA-Motiven zu suchen. Als Beispiel dafür bietet die Webapplikation die Möglichkeit über den Button *Human Poly(A)** direkt nach Polyadenylierungsstellen zu suchen.

Als Beispiel für die Poly(A) Motivesuche wurde humane mRNA untersucht (La wurde auf neun gesetzt, Ld auf acht, Modulusuche ausgestellt, eigene Matrizen wurden gewählt). Die Matrize für die Poly(A)-Suche wurde in *AIModules* selbst erzeugt mit den Sequenzen aus [3]. Das Ergebnis der Suche ist in Tabelle 3.1 abgedruckt.

Tabelle 3.1: Das Ergebnis der Polyadenylierungsstellensuche (* siehe Tabelle 1.2)

Sequence Name	Sequence Length	Matrix Title	Matrix Length	Hit No.	Hit Sense	Hit Start	Hit Stop	Hit Score (La)	Hit Max log likelihood ratio score (Lm) or matrix possible	Difference (Ld) (maxscore(Lm) - score(La))	Hit Oligo
NM_000600.5 Homo sapiens interleukin 6 (IL6), transcript variant 1, mRNA	1127	human-Poly-A	6	1	N	1025	1030	9.889837	9.889837	0.000000	AATAAA
				2	N	1103	1108	9.889837	9.889837	0.000000	AATAAA
				3	R	842	847	9.889837	9.889837	0.000000	TTTATT
NM_000594.4 Homo sapiens tumor necrosis factor (TNF), mRNA	1678	human-Poly-A	6	1	N	1655	1660	9.889837	9.889837	0.000000	AATAAA
				2	R	928	933	9.889837	9.889837	0.000000	TTTATT
				3	R	1345	1350	9.889837	9.889837	0.000000	TTTATT
				4	R	1352	1357	9.889837	9.889837	0.000000	TTTATT
				5	R	1356	1361	9.889837	9.889837	0.000000	TTTATT
				6	R	1363	1368	9.889837	9.889837	0.000000	TTTATT
				7	R	1367	1372	9.889837	9.889837	0.000000	TTTATT
				8	R	1387	1392	9.889837	9.889837	0.000000	TTTATT
NM_020525.5 Homo sapiens interleukin 22 (IL22), mRNA	1165	human-Poly-A	6	1	N	1144	1149	9.889837	9.889837	0.000000	AATAAA
				2	R	824	829	9.889837	9.889837	0.000000	TTTATT
				3	R	873	878	9.889837	9.889837	0.000000	TTTATT
				4	R	991	996	9.889837	9.889837	0.000000	TTTATT
				5	R	1015	1020	9.889837	9.889837	0.000000	TTTATT
				6	R	1029	1034	9.889837	9.889837	0.000000	TTTATT
				7	R	1043	1048	9.889837	9.889837	0.000000	TTTATT
				8	R	1131	1136	9.889837	9.889837	0.000000	TTTATT

3.4 Modulesuche

Um die Modulesuche des Tools AIModules zu testen, wurden Promotoren von Interleukin Genen auf Module hin untersucht (La wurde auf acht gesetzt, Ld auf acht, Modulesuche wurde aktiviert, Schwellwert für Module wurde auf drei gesetzt — also maximale Konserviertheit von TFBSs —, die Vertebratenmatrices von JASPAR wurden gewählt). Das Ergebnis ist, dass für alle Sequenzen wie erwartet NFAT gefunden wurde, wie in [5] beschrieben. Die Bilddateien wurden als Vektorgrafik (*SVG*-Datei)⁶ heruntergeladen und sind in den Abbildungen 3.11 und 3.12 dargestellt.

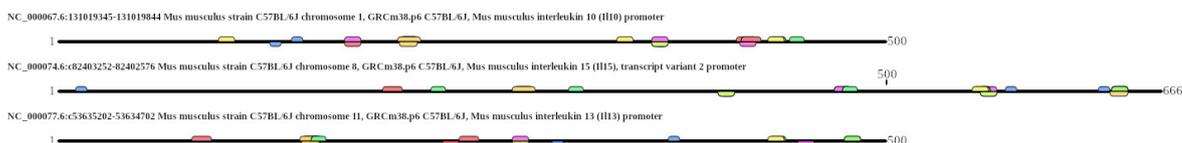


Abbildung 3.11: Modulesuche für Interleukingene (*oben*: NC_000067.6:131019345-131019844 Mus musculus strain C57BL/6J chromosome 1, GRCm38.p6 C57BL/6J, Mus musculus interleukin 10 (Il10) promoter, *mitte*: NC_000074.6:c82403252-82402576 Mus musculus strain C57BL/6J chromosome 8, GRCm38.p6 C57BL/6J, Mus musculus interleukin 15 (Il15), transcript variant 2 promoter, *unten*: NC_000077.6:c53635202-53634702 Mus musculus strain C57BL/6J chromosome 11, GRCm38.p6 C57BL/6J, Mus musculus interleukin 13 (Il13) promoter)

Die verwendeten Promotorsequenzen und die gefundenen Module sowie TFBSs sind im Anhang 5 angefügt.

⁶Scalable Vector Graphics



Abbildung 3.12: Legende der Interleukinsuche

3.5 Analysegeschwindigkeit

Um einen Vergleich der Geschwindigkeit von AIModules zu haben, wurde es mit conTraV3 verglichen [60, 61]. Obwohl conTraV3 im produktiven Zustand getestet wurde und AIModules in einer Virtualen Maschine in *VirtualBox* (Debian, 4,5 GB RAM und zwei Prozessorkerne Intel i7-3520m 2x2,9 GHz) lokal getestet wurde, war die Analysegeschwindigkeit von AIModules besser. Es wurde eine Sequenz verglichen (AJ223836.1 *Chionodraco hamatus* mRNA Für Cathepsin; Länge 1332 bp; die Sequenz ist im Anhang 5). Im Tool conTraV3 wurde eine Matrize (PWM1id:TF1name:HOME) und Standardwerte gewählt (Stringency: core = 0.95, similarity matrix = 0.85) und im Falle von AIModules alle 1317 Matrizen mit den Standardwerten $La=6$ sowie $Ld=8$. AIModules benötigte für die Analyse im Backend nur ca. eine Sekunde und ca. fünf Sekunden, um das Ergebnis der gefundenen TFBSs anzuzeigen. Das Vergleichstool conTraV3 lief eine Stunde und die Analyse wurde danach vom Anwender abgebrochen.

3.6 TFBS- und Moduleanalyse mit Genomatix

In diesem Abschnitt werden Ergebnisse der Analyse von *Cathepsin* und *IL-10* dargestellt. Die betrachteten Tools sind unter den

Punkten 1.5 und 1.6 beschrieben. Daraus ergibt sich, dass zum Zeitpunkt der Erstellung dieser Arbeit nur zwei Produkte bekannt waren, die auch Module erkennen konnte: *Genomatix* und *TRANSFAC*. Um die Resultate zu vergleichen wurde *Genomatix* als Tool gewählt. Dieses Tool muss nach einer einwöchigen Testperiode kostenpflichtig lizenziert werden. Die Analysen werden mit *AIModules* und vergleichshalber mit *Genomatix* durchgeführt. Die homologen Gene sind aus *GenBank*. Die Kandidaten für *Cathepsine* sind

- Homo_sapiens_cathepsin_V_transcript_variant-1_promoter
- Bos_taurus_cathepsin-Z_promoter
- Mus_musculus_cathepsin-F-transcript-variant-X1_promoter_576-1076

und für *IL-10*

- X73536.1_H.sapiens_promoter_region_of_human_IL-10_gene
- AY486432.1_Macaca-mulatta_interleukin-10-(IL-10)_gene_promoter_region
- AF121965.1_Mus-musculus_interleukin-10-(IL10)_gene_promoter_partial_sequence

Die verwendeten Sequenzen sind im Appendix in Kapitel 5 hinterlegt. Für die Beschreibung der unterschiedlichen Gewichtungungsverfahren der beiden Anwendungen siehe auch Abschnitt 2.6.

Wir haben für jede Sequenz die TFBS- und Moduleanalyse auf beiden Systemen durchgeführt. Da sich die Matrizenamen in beiden Systemen unterscheiden, mussten die Ergebnisse aufgearbeitet werden. Für die Modulesuche genügte ein manueller Vergleich beider Welten, da *Genomatix* wenige Module auswies. Im Falle

der TFBS-Suche waren die Ergebnisse für eine manuelle Suche erschöpfend, sodass die Vorgehensweise darin bestand die Ergebnisse aus *AIModules* so anzupassen, dass nur der Familienname der Matrize übrig blieb. Diese Werte wurden für jede Sequenz in ein Array kopiert. Daneben wurden die *Genomatix* Ergebnisse unverändert in weitere Arrays verpackt, sodass durch ein *python-Script* (siehe Anhang 5) ein semi-automatisches Suchen möglich wurde. Dazu wurden die Arrays der *AIModules* Ergebnisse mit den korrelierenden Arrays der *Genomatix* Ergebnisse auf Stringgleichheit überprüft. Die Resultate aus diesem Vergleich wurden in separate Arrays kopiert und per *standard output* ausgedruckt. Das Format war dabei *AIModules-TF-Familien-Name::Genomatix-TF-Name*, wobei *Genomatix-TF-Name* eine kommaseparierte Liste sein konnte. Dieser Ausdruck wurde manuell in *LibreOffice Calc* weiterbearbeitet, sodass die Ergebnisse aus beiden Softwaresystemen einem unvollständigen Vergleich unterzogen werden konnten.

3.6.1 TFBSs-Analyse

1. Cathepsine

Cathepsine sind Proteasen mit katalytischer Triade zur Verdauung anderer Proteine. Das aktive Zentrum enthält einen Cysteinrest, aber auch Serin ist bekannt. Dies kann für weitere Funktionen in der Physiologie genutzt werden, z.B. Antikörperproduktion oder Proteinabbau. In der Pathophysiologie spielen Cathepsine eine Rolle in der Metastasierung und der Differenzierung in der Knochenbildung, sowie bei rheumatischen Erkrankungen. Es ist eine ganze Cathepsinfamilie im menschlichen Körper bekannt. Bei der Untersuchung der Cathepsine stellte sich die Frage, ob die grundsätzlichen Promoten alle gleich oder je nach Zelltyp an und abschaltbar sind. Interessanterweise zeigte es sich, dass diese im Prinzip ähnlich funktionieren [26]. Dieses Ergebnis, das ursprünglich

Jahre an Bearbeitungszeit benötigte, kann mit dem neuen Tool in minutenschnelle dargestellt werden.

Die Parameter für die Cathepsin–TFBS–Suche sind in Tabelle 3.2 aufgeführt.

Tabelle 3.2: Parameter für die Suche nach TFBSs in AIModules und Genomatix für Cathepsine (* siehe Tabelle 1.2)

AIModules	Genomatix (Matinspector)
La wurde auf sechs und Ld auf acht gesetzt. Vertebraten-Matrizen wurden aus der JASPAR Datenbank selektiert.	Die Matrix Family Library Version 11.1 wurde verwendet und Vertebratenmatrizes (0.75/Optimized) sowie General Core Promoter Elements (0.75/Optimized) ausgewählt.

Eine Matrix kann N Bindestellen pro Sequenz und Orientierung als Ergebnis produzieren.

Tabelle 3.3: Ergebnisse für Cathepsine in der TFBS-Suche mittels AIModules und Genomatix. Es wurden drei Cathepsinpromotoren analysiert. Mit AIModules wurden viel mehr TFBSs gefunden als mit Genomatix. Beide Anwendungen fanden gemeinsame TFBSs. (* siehe Tabelle 1.2)

Sequenzname	Homo sapiens cathepsin V transcript variant-1 promoter	Bos taurus cathepsin-Z promoter	Mus musculus cathepsin-F-transcript-variant-X1 promoter 576-1076
Anzahl TFBS AIModules	253	322	492
Anzahl TFBS Genomatix	155	224	166
Gemeinsame TFBS oder Cathepsin-Famile	19	24	28

Mit dem Tool AIModules wurde in allen drei Fällen mehr TFBSs gefunden als mit Genomatix, wobei gemeinsame Bindestellen beider Systeme vorkamen (Tabelle 3.3).

2. IL-10

Interleukine sind Zytokine, die unterschiedlich auf Zellen des Immunsystems wirken, wobei einige Immunreaktionen bestärken und andere antagonistisch wirken. So wirkt Interleukin-10 (IL-10) als ein wichtiges Immunsignal u. a. dämpfend auf das Immunsystem, indem es die Bildung anderer Zytokine hemmt und somit eine überschießende Immunreaktionen verhindert. Zudem fördert es die Antikörperproduktion durch B-Lymphozyten. In der Pathophysiologie spielt das Zytokin eine wichtige Rolle bei entzündlichen Darmerkrankungen. Interleukine sind über Spezies hinweg bekannt. Somit kann das Tool *AIModules* dieses Zytokin über Organismen hinweg detektieren helfen, um z. B. über die Konserviertheit einen phylogenetischen Stammbaum zu erstellen.

Die Parameter für die IL-10–TFBS–Suche sind in Tabelle 3.4 aufgeführt.

Tabelle 3.4: Parameter für die Suche nach TFBSs in AIModules und Genomatix für IL-10 (* siehe Tabelle 1.2)

AIModules	Genomatix (Matinspector)
La wurde auf sechs und Ld auf acht gesetzt. Vertebraten-Matrizen wurden aus der JASPAR Datenbank selektiert.	Die Matrix Family Library Version 11.1 wurde verwendet und Vertebratenmatrizes (0.75/Optimized) sowie General Core Promoter Elements (0.75/Optimized) ausgewählt.

Eine Matrix kann N Bindestellen pro Sequenz und Orientierung als Ergebnis produzieren.

In allen drei Analysen hat AIModules mehr Bindestellen als Genomatix gefunden (Tabelle 3.5) und in zwei Fällen doppelt

Tabelle 3.5: Ergebnisse für IL-10 in der TFBS-Suche mit AIModules und Genomatix. AIModules fand viel mehr Transkriptionsbindestellen als Genomatix, wobei gemeinsame TFBSs gefunden wurden. (* siehe Tabelle 1.2)

Sequenzname	X73536.1 H.sapiens promoter region of human IL-10 gene	AY486432.1 Macaca-mulatta interleukin-10-(IL-10) gene promoter region	AF121965.1 Mus-musculus interleukin-10-(IL10) gene promoter partial sequence
Anzahl TFBS AIModules	918	963	212
Anzahl TFBS Genomatix	289	350	93
Gemeinsame TFBS oder IL-Famile	42	56	18

so viele, wobei gemeinsame Bindestellen vorkamen. Mit den Einstellungen in Tabelle 3.4 wurden Bindestellen in den Sequenzen X73536.1_H.sapiens_promoter_region_of_human_IL-10_gene und AY486432.1_Macaca-mulatta_interleukin-10-(IL-10)_gene_promoter_region gefunden (MA0519.1 Stat5a:: Stat5b, MA0518.1 Stat4, MA0137.3 STAT1 und MA0144.2 STAT3), aber nicht für AF121965.1_Mus-musculus_interleukin-10-(IL10)_gene_promoter_partial_sequence. Genomatix hat dasselbe Resultat für den TF Stat gezeigt und es wurde für die Sequenz AF121965.1_Mus-musculus_interleukin-10-(IL10)_gene_promoter_partial_sequence dieser TF genauso nicht gefunden. Die gemeinsamen Bindestellen sind im Appendix 5 hinterlegt.

3.6.2 Module-Analyse

1. Cathepsine

Die Parameter für die Cathepsin-Module-Suche sind in Ta-

belle 3.6 aufgeführt.

Tabelle 3.6: Parameter für die Suche nach Modulen in AIModules und Genomatix für Cathepsine (* siehe Tabelle 1.2)

AIModules	Genomatix (ModelInspector)
La wurde auf sechs und Ld auf acht gesetzt. Vertebraten-Matrizen wurden aus der JASPAR Datenbank selektiert. Die Checkbox für die Modulesuche wurde aktiviert und die Schwelle, die definiert, wann ein TF für die Modulesuche in Frage kommt, wurde auf 3 gestellt (die Anzahl der eingegebenen Sequenzen).	Es wurde ausgewählt: The Vertebrate Modules Version 6.3 und Matrix Family Library Version 11.1. Beide Schwellen wurden folgend eingestellt: Threshold for number of elements: 100.0%, Maximum number of matches: 1000

Die Ergebnisse beziehen sich auf gefundene Module beider Orientierungen eines Stranges.

Tabelle 3.7: Ergebnisse für Cathepsine in der Module-Suche. AIModules fand viel mehr Module als Genomatix. Gemeinsame Module konnten jedoch nicht identifiziert werden. (* siehe Tabelle 1.2)

Sequenzname	Homo sapiens cathepsin V transcript variant-1 promoter	Bos taurus cathepsin-Z promoter	Mus musculus cathepsin-F-transcript-variant-X1 promoter 576-1076
Anzahl Module AIModules	486	1.497	640
Anzahl TFBS AIModules	47	79	60
Anzahl Module Genomatix	15	29	9
Gemeinsame Module	0	0	0

In allen drei Fällen hat das Tool AIModules mindestens um den Faktor 10 mehr Module gefunden als Genomatix, wobei gemeinsame Module nicht gefunden wurden (Tabelle 3.7).

2. IL-10

Die Parameter für die IL-10-Module-Suche sind in Tabelle 3.8 aufgeführt.

Tabelle 3.8: Parameter für die Suche nach Modulen in AIModules und Genomatix für IL-10 (* siehe Tabelle 1.2)

AIModules	Genomatix (ModelInspector)
La wurde auf sieben erhöht, da die Berechnung sehr lange gedauert hat, und Ld auf acht gesetzt. Vertebraten-Matrizen wurden aus der JASPAR Datenbank selektiert. Die Checkbox für die Modulesuche wurde aktiviert und die Schwelle, die definiert, wann ein TF für die Modulesuche in Frage kommt, wurde auf 3 gestellt (die Anzahl der eingegebenen Sequenzen).	Es wurde ausgewählt: The Vertebrate Modules Version 6.3 und Matrix Family Library Version 11.1. Beide Schwellen wurden folgend eingestellt: Threshold for number of elements: 100.0%, Maximum number of matches: 1000

Die Ergebnisse geben die gefundenen Module auf dem (+)- und (-)-Strang wieder.

AIModules hat in allen drei Fällen um mindestens den Faktor 10 mehr Module als Genomatix gefunden. Die manuelle Suche nach gemeinsamen Modulen zwischen beiden Systemen hat keine Ergebnisse zu Tage geführt (Tabelle 3.9).

Tabelle 3.9: Ergebnisse für IL-10 in der Module-Suche. AIModules identifizierte viel mehr Module als Genomatix. Gemeinsame Module liegen nicht vor. (* siehe Tabelle 1.2)

Sequenzname	X73536.1 H.sapiens promoter region of human IL-10 gene	AY486432.1 Macaca-mulatta interleukin-10-(IL-10) gene promoter region	AF121965.1 Mus-musculus interleukin-10-(IL10) gene promoter partial sequence
Anzahl Module AIModules	941	1.131	502
Anzahl TFBS AIModules	62	80	52
Anzahl Module Genomatix	13	15	3
Gemeinsame Module	0	0	0

Insgesamt gelang es durch die Promotorsuche aussagekräftige biologische Ergebnisse zu erzielen. Wir zeigen, dass die Module-suche viele Ergebnisse liefert und auch die Konserviertheit über Spezies hinweg berücksichtigt. Gleiches gilt auch bei RNA Motiven, die durch eine Uracil nach Thymin Konvertierung in der Applikation gewürdigt wird. Dieser Faden wird in der Diskussion wieder aufgenommen.

Kapitel 4

Diskussion

AIModules wurde entwickelt, da nach unseren Erkenntnissen zum Zeitpunkt der Erstellung dieser Dissertation nur zwei Softwareprodukte die Möglichkeit zur Suche nach Modulen boten. Diese Produkte mit Namen *Genomatix* (s. Punkt 9) respektive *TRANS-FAC* (s. Punkt 10) sind kommerziell, wobei Ersteres für eine Testperiode kostenfrei genutzt werden kann und Letzteres eine öffentliche und freie Version anbietet, die allerdings veraltete Matrizen aus dem Jahre 2005 enthält. Um den vollen Funktionsumfang zu nutzen, müssen beide Lösungen lizenziert werden. Aber auch dann steht der Code nicht frei zur Verfügung. *AIModules* ist dagegen kostenlos und im Code offengelegt (<https://zenodo.org/badge/latestdoi/363702392>). *AIModules* ist außerdem ein Beispiel für die FAIR¹ principles, da die von uns verwendeten Matrizen menschen- und maschinenlesbar sind und zudem die Originalbezeichner der Jaspar-DB enthalten (Findable). Außerdem sind die Daten unserer Applikation über die offene REST-Schnittstelle auslesbar (Accessible) und die Ergebnisse sind in einem Format herunterladbar (JSON, SVG, Excel/OpenOffice Calc), das mit kostenlosen und öffentlich zugänglichen Werkzeugen gelesen sowie bearbeitet werden kann (Interoperability). Zudem halten sich die Bezeichner der Matrizen an öffentliche Vorgaben und die Nutzung

¹<https://www.go-fair.org/fair-principles/>, Zugriff am 29. Juli 2021

sowie Verwertung der Ergebnisse aus *AIModules* stehen unter einer offenen Lizenz (Reusable).

Unser hier vorgestellter Webservice ist kostenlos und ermöglicht die Suche nach TFBSs und gemeinsamen Modulen. Dabei kann der Anwender eigene RNA- oder DNA-Sequenzen eingeben und diese mittels eigenen oder voreingestellten JASPAR-Matrizen [34] filtern. Die TFBS Suche findet für jede Sequenz einzeln Bindestellen anhand der vom Anwender festgelegten Parameter L_a und L_d . Dabei bestimmt der Parameter L_a die untere Schwelle, ab der eine Bindestelle valide ist und L_d die obere. Zusätzlich ist es möglich regulatorische Elemente auf der mRNA wie beispielsweise Polyadenylierungsstellen zu finden. Darüber hinaus kann der Webservice für die Modulesuche verwendet werden. Ein Modul ist hierbei ein Auftreffen von zwei TFBSs mit einem Abstand dazwischen. Der Anwender kann über einen Parameter bestimmen, ab wann ein Modul valide ist. Dazu wird angegeben in wie vielen mitgegebenen Sequenzen ein TF vorkommen muss, damit dieser in der Modulesuche berücksichtigt wird. Der Wert kann zwischen Eins und der Anzahl an eingefügten Sequenzen liegen (maximale Konserviertheit einer TFBS). Wenn ein Modul gefunden wird, ist dies das Ergebnis eines Suchalgorithmus. Dieser bestimmt, dass Module valide sind, wenn diese höchstens ± 200 bp auf unterschiedlichen Sequenzen vorkommen und dieses Vorkommen auf mindestens zwei Sequenzen verteilt ist. Das Tool *AIModules* ist performant im Vergleich zu *conTraV3*, da unser Tool für eine vergleichbare Analyse nur Sekunden benötigt, wohingegen *conTraV3* nach einer Stunde noch kein Ergebnis lieferte.

4.1 Verfügbare Tools

Zum Zeitpunkt des Erstellens dieses Dokuments waren nur zwei Produkte auf dem Markt, die Module vorhersagen konnten. *Genomatrix* bzw. *TRANSFAC* sind kommerziell und müssen kos-

tenpflichtig lizenziert werden. Ein weiteres Tool (*ModuleMaster*) soll auch Module erkennen können. Allerdings konnte diese Java-Webstart-Applikation auf unterschiedlichen Betriebssystemen nicht gestartet werden. Das ursprüngliche Institut konnte nicht weiterhelfen, da die Bioinformatikforschung eingestellt wurde. Zusätzlich ist die Benutzung einer Website, wie im Falle von AIModules, für den Endbenutzer einfacher als das Starten einer Java-WebStart-Applikation, die einerseits nicht aktuell zu sein scheint und andererseits ohne ein gültiges Zertifikat ausgeliefert wird, sodass eine Zertifikatswarnung angezeigt wird. Alle anderen Tools in Tabelle 1.2 konnten keine gemeinsamen Module vorhersagen. Das Tool AIModules kann als Webapplikation auf Servern deployed werden, aber auch On-site oder lokal auf einem PC oder Notebook betrieben werden.

4.2 Vergleich zwischen AIModules und Genomatix

Zusätzlich wurde *AIModules* mit der kommerziellen Lösung *Genomatix* verglichen und versucht Gemeinsamkeiten und Unterschiede in den Resultaten aufzuzeigen. Bei Genomatix handelt es sich um einen Webservice, der u.a. TFBS und Module auf Sequenzen findet. Gemeinsame Module auf N-Sequenzen werden allerdings nicht angezeigt, wohingegen AIModules genau das auch kann.

Für die Analyse wurden Cathepsin- und Interleukin-10 Sequenzen hergenommen und in beiden Lösungen verarbeitet. Die gefundenen TFBSs für Cathepsine sind in Tabelle 3.3 dargestellt und für IL-10 in Tabelle 3.5, wohingegen gefundene Module für Cathepsine in Tabelle 3.7 und für IL-10 in Tabelle 3.9 abgebildet sind. Aus den Resultaten wurde ersichtlich, dass die Anzahl an gefundenen TFBSs in beiden Lösungen unterschiedlich ist. Dem liegt zugrunde, dass die verfügbaren Matrizen unterschiedlich in

Anzahl und Qualität sind, und andererseits schwer vergleichbare Parameter Verwendung finden. Dies sind für AIModules La sowie Ld und für Genomatix 0.75 und *Optimized* (s. auch Abschnitt 2.6). Dadurch ist eine Normalisierung schwierig und ein Vergleich der Ergebnisse ergibt nur eine geringe Schnittmenge. Zudem teilen sich beide Systeme nicht alle und die gleichen Matrizen.

Mit der Moduleanalyse verhält es sich ähnlich wie mit der Suche nach TFBSs, *i.e.*, dass die Parameter für AIModules La, Ld und die aktivierte Checkbox, wohingegen die Parameter für Genomatix *Threshold for number of elements* und *Maximum number of matches* sind. Die Modulesuche in AIModules folgt einem strikten Algorithmus, um gemeinsame Module auf den eingegebenen Sequenzen zu finden. Dazu muss ein TFBS auf N Eingabesequenzen vorkommen, um für die Modulesuche herangezogen werden zu können. Die Anzahl an Vorkommnissen eines TFBS auf N Inputsequenzen kann vom Anwender angegeben werden und entspricht dem Grad der Konserviertheit. Der Fund einer TFBS auf dem sense- oder antisense-Strang ist dabei relevant. Zusätzlich besteht ein Modul aus zwei TFBSs mit einem Versatz von ± 200 bp. Dabei werden alle Permutationen der TFBSs getestet (AB, AC, AD, ..., BC, BD, ...). Das Modul ist erst dann valide, wenn es mit dem genannten Versatz in mindestens zwei der Inputsequenzen vorkommt. D.h. aber auch, dass ein hoher La und niedriger Ld dazu führen kann, dass ein TF nur in $N-1$ und nicht in N Inputsequenzen vorkommt, da dieses außerhalb der Schwellenwerte liegt und deshalb nicht mit in die Analyse aufgenommen wird. Dies kann geheilt werden, indem der entsprechende Schwellenwert für die Inputsequenzen erniedrigt wird. Dieses Verhalten kann z.B. beobachtet werden, wenn die Resultate aus AIModules für die TFBS Suche verglichen werden (Tabelle 3.3 (253) und 3.7 (47)). Das bedeutet, dass bei der aktivierten Modulesuche die Anzahl an gefundenen TFBSs niedriger ist, als bei der reinen Suche nach TFBSs. Im Tool Genomatix werden alle Inputsequenzen un-

abhängig voneinander auf bekannte Module hin analysiert. I. e., dass Genomatrix keine gemeinsamen Module anzeigt und dies deshalb für AIModules ein Alleinstellungsmerkmal darstellt. Weiterhin können im Tool Genomatrix vom Anwender Module mit einem Versatz spezifiziert und in die Modulesuche übernommen werden. Diese Unterschiede in der Modulesuche der beiden Systeme führt zu Unterschieden in der Menge an gefundenen Modulen. AIModules findet alle möglichen Module algorithmisch, wohingegen Genomatrix auf bekannte Kolokationen setzt. Dies hat die Konsequenz, dass AIModules sehr viele Module findet, die auch falsch-positiv sein können, wohingegen Genomatrix Module auslassen kann, die AIModules richtigerweise findet (falsch-negativ). Allerdings kann die Anzahl an validen gefundenen Modulen in AIModules erhöht werden, indem L_a erhöht oder L_d erniedrigt oder nur eine spezifische Selektion an Inputmatrizen des Anwenders verwendet oder eine Kombination aus den genannten Punkten angewandt wird. Die in beiden Systemen analysierten Cathepsin- und IL-10 Sequenzen haben keine Gemeinsamkeiten hinsichtlich der Module ergeben. Wie dem auch sei, sollten die *in silico* gefundenen Module aus beiden Systemen durch Experimente validiert werden.

4.3 Das Scoring

Da für die interne Berechnung des Scores log-Wahrscheinlichkeiten hergenommen werden, können die einzelnen Basen innerhalb einer putativen Bindesequenz aufsummiert werden. Aufgrund der Natur dieser log-Wahrscheinlichkeiten ist der Beitrag an jeder Base zum Gesamtscore maximal 2 Bit. Dies ist der Fall, wenn die Base der Consensus Sequenz entspricht. Das bedeutet z. B., dass bei einem 6-mer ein Score von maximal $6 \cdot 2 = 12$ möglich ist. Der vom Anwender zu vergebende Parameter L_a stellt den minimalen log Wahrscheinlichkeitswert dar, ab dem ein Score im Endergebnis berücksichtigt wird. Sollten Matrizen vorhanden sein, deren Grund-

lage ein Alignment von wenigen Sequenzen ist, kann deren Aussagekraft verstärkt werden, indem ein sogenannter *pseudocount* an jeder Position aufaddiert wird. Da nur die JASPAR-Matrizen in AIModules verwendet werden, die aus mehr als 20 alignierten Sequenzen bestehen, spielen *pseudocounts* eine geringe Rolle und werden standardmäßig mit eins vorbelegt — also 0,25 für jede Base. Die Consensus Sequenz stellt den maximal möglichen Score einer Bindestelle dar und wird intern im Parameter *Lm* hinterlegt. Für die Definition der oberen Schwelle für valide Bindestellen definiert der Anwender den maximalen Wert für das Wahrscheinlichkeitsdefizit (*Ld*). Dieser ist definiert als die Differenz zwischen der maximalen und der minimalen Log-Wahrscheinlichkeit (*Lm-La*). Dabei muss erwähnt werden, dass *La* und *Ld* in Konkurrenz zueinander stehen können: Ein höherer *Ld* favorisiert längere Matrizen mit weniger starken log-Wahrscheinlichkeiten und ein höherer *La* führt zur Bevorzugung von längeren gegenüber kürzeren Matrizen. Die in der Webapplikation verwendeten Parameter sind für *La* sieben und für *Ld* acht. Diese Werte können initial verwendet werden, um grob nach Bindestellen zu suchen und eine Vorauswahl bei den Matrizen zu wählen, um die folgende Modulesuche spezifischer zu gestalten. Da die Modulesuche und das Rendering des Ergebnisses auf dem Rechner des Anwenders erfolgt, ist diese Vorauswahl ein wichtiger Faktor für die Dauer der Berechnungen. Sollte entweder die Modulesuche oder das Rendering zu lange brauchen, kann die Sequenzanzahl oder die Matrizenanzahl reduziert oder *La* erhöht oder *Ld* verringert werden oder eine Kombination aus den genannten Punkten Anwendung finden.

Das Ergebnis aus AIModules inkludiert falsch-positive und falsch-negative Bindestellen. Dies kann raffiniert werden, indem z. B. kurze Sequenzen oder eine geringe Matrizenanzahl oder lange Matrizen mit einer großen alignierten Sequenzbasis oder eine Kombination aus all diesen Punkten verwendet wird. Zusätzlich kann über die Inputparameter *La* und *Ld* die Aussagekraft des Ergeb-

nisses erhöht werden. Die im Backend von AIModules verwendete Anwendung von *pseudocounts* anhand einer angenommenen uniformen Verteilung der Basenhäufigkeit kann in Hinblick auf eine stärkere Aussagekraft des Ergebnisses theoretisch verbessern. Mit der eingesetzten uniformen Häufigkeitsverteilung werden nämlich CG arme oder schwach AT-reiche Sequenzen nicht valider, denn AT reiche Bindestellen werden signifikanter, wohingegen GC reiche weniger signifikant gewichtet werden. Deshalb ist die bessere Alternative, dass die erzielten Ergebnisse mit den Ergebnissen aus einem Testset verglichen werden, die speziell für jede der analysierten Promotorsequenzen zusammengestellt wurden [50]. Soweit bekannt, ist dies zum Zeitpunkt der Erstellung dieser Arbeit allerdings in keinem Produkt inkludiert worden.

4.4 Analysegeschwindigkeit und Suche nach Polyadenylierungsstellen

Die Analysegeschwindigkeit zwischen AIModules und einem schon vorhandenen Tool (*conTraV3*) hat ergeben, dass AIModules für die Analyse derselben DNA-Sequenz und viel mehr Matrizen nur Sekunden benötigt, wohingegen die Analyse mit *conTraV3* mit nur einer Matrize nach einer Stunde ergebnislos abgebrochen wurde (siehe Abschnitt 3.5). Zusätzlich kann das Tool AIModules regulatorische Elemente auf mRNA wie beispielsweise Polyadenylierungsstellen [3] vorhersagen (siehe Abschnitt 3.3).

4.5 Interpretation der Ergebnisse aus *AIModules*

Das Ergebnis der Analysen aus AIModules wird im Browser gezeichnet und zeigt, dass TFBS überlappen (Abbildung 3.11). Dabei wird vorher im Backend das Ergebnis der Analyse durch die

Schwellenwerte La und Ld gefiltert, um nur starke Bindungen und Treffer mit einem hohen Score anzuzeigen. Die Ergebnisse müssen auch als *in silico* Ergebnisse gewürdigt und durch Experimente validiert werden, da die Vorhersage einer TFBS bedeutet, dass die Binding *in vitro* zustande kommen kann, aber nicht zwangsläufig bedeutet, dass dieser TF *in vivo* tatsächlich eine Rolle spielt.

Andererseits, wenn im Ergebnis der Motivanalyse eine TFBS fehlt, bedeutet dies nicht, dass es diese TFBS nicht gibt. Der Grund des Fehlens kann darin liegen, dass der Treffer/Score von einem hohen La oder niedrigen Ld ausgeschlossen wurde, oder dass dieser TF in der JASPAR Datenbank nicht oder noch nicht enthalten ist, obwohl diese viel aktueller und umfangreicher ist als z.B. die öffentliche TRANSFAC Datenbank. Diese Problematik findet sich allerdings auch in den kommerziellen Werkzeugen wieder.

Der Score (logarithmierte Wahrscheinlichkeitsquotienten) ist die Summe aus allen Basen einer Matrize, in der jede Base bis zu zwei Bit beitragen kann. Wenn jede Base zwei Bits an Informationen beiträgt, erhält man den maximalen logarithmierten Wahrscheinlichkeitsquotienten (maximum log likelihood ratio score (Lm)) bzw. die Consensus Sequenz [43, 44, 45, 55].

Die Selbstinformation (self information) eines Elements in einer Matrize ist

$$I(i) = -p_{i,j} * \log_2(p_{i,j}) \quad (4.1)$$

und der Informationsgehalt (Information Content (IC)) demnach

$$I(i) = - \sum p_{i,j} * \log_2(p_{i,j}) \quad (4.2)$$

und mit den Hintergrundfrequenzen ist der IC

$$I(i) = - \sum p_{i,j} * \log_2(p_{i,j}/p_j) \quad (4.3)$$

Der IC gibt den Grad der Konserviertheit an und beginnt bei Null für keine Konserviertheit an dieser Position bis zu zwei, was

für maximale Konserviertheit dieser Position steht. D.h., dass für eine Position, die konserviert ist, der Informationsgehalt bei zwei Bits liegt. Durch die logarithmierten Wahrscheinlichkeiten erhält man den Informationsgehalt der gesamten Matrize dadurch, dass der Informationsgehalt an jeder Position aufaddiert wird. Dadurch erhält man für ein erneutes Auftreten eines Treffers dieser Matrize 2^{IC} . Somit erhält man mit längeren Matrizen, die als Grundlage viele Sequenzen für das Alignment heranziehen, verlässlichere Treffer, und z. B. für einen konservierten 8-mer einen IC von $8 \cdot 2 = 16$. Dieser Informationsgehalt ist in Wirklichkeit niedriger, weil Pseudocounts verwendet werden.

Das Backend berechnet den Score mittels eines logarithmierten Wahrscheinlichkeitsquotienten (log likelihood ratio score)

$$L_a(X) = \sum \log_2(p_{i,j}) - \sum \log_2(u_j) \quad (4.4)$$

, wobei u_j das Hintergrundmodell (background model) für eine zufällige Sequenz darstellt. Eine Regel für den Schwellenwert L_a ist deshalb schwierig zu vereinbaren, da sich Matrizen in Länge und Qualität unterscheiden. Deshalb kann der Anwender zwei Parameter mitgeben. Neben L_a ist auch L_d vorhanden, das ein Defizit angibt, *i.e.*, dass eine obere Schwelle für valide Treffer definiert werden kann. Dabei gilt, je geringer L_d desto näher ist die obere Schwelle der Consensus Sequenz. Allgemein gilt, dass für kurze Matrizen L_d weniger als zwei bis vier liegen sollte.

Abschließend handelt es sich bei der *in silico* Analyse der TFBSs um Vorhersagen, die darauf hindeuten, dass ein TF *in vitro* binden kann. Da aber *in vivo* Parameter eine Rolle spielen, wie beispielsweise die Interaktion von TFs mit Chromatin (Konformation der DNA) oder die Menge an TFs und Kofaktoren, sollten Bindestellen durch Experimente validiert werden.

4.6 Anwendungsmöglichkeiten

Das Werkzeug *AIModules* bietet einen großen Funktionsumfang und stellt eine freie Alternative (*open source*) zu den kommerziellen Produkten zur Verfügung. Auf der einen Seite bietet unsere Applikation eine TFBS-Suche, die mit eigenen Sequenzen befüllt werden kann. Zudem erlaubt es unser Tool für die Analyse eigene Matrizen zu verwenden oder auf hinterlegte Matrizen aus der Jaspar-DB zurückzugreifen, sodass spezifische sowie umfangreiche Analysen möglich sind, die es auch zulassen, RNA-Motive zu untersuchen. Dies wurde am Beispiel einer Suche nach Polyadenylierungsstellen demonstriert (siehe Abschnitt 4.4), kann aber natürlich auf andere RNA-Motive genauso angewandt werden, wie beispielsweise die Consensussequenzen der spliceosomalen snRNP [30]. Hierzu benötigt der Anwender eine passende Matrize. Auf der anderen Seite wurde unsere Applikation durch eine Modulesuche supplementiert, sodass zusammenhängende DNA- und RNA-Motive detektiert werden können. Mit diesem Umfang sind biologische Untersuchungen möglich, die die zentrale Bedeutung von Promotoren würdigen, stellen diese doch eine wichtige Anpassungsmöglichkeit für einen Organismus dar, um über das An- und Abschalten von Promotoren die Genexpression zu regulieren. Somit können auch Stammbäume mittels konservierter Module über Organismen hinweg untersucht und phylogenetische Analysen erstellt werden.

Daneben können auch Chromosomal Territories oder Chromosomale Domänen identifiziert werden, wenn es dafür Matrizen zur Verfügung gibt. Zudem können Enhancer/Silencer mittels unseres Tools identifiziert werden. Neue Matrizen, die aus Alignments aus ENCODE [11] hervorgehen, können genauso für die Suche nach funktionellen Elementen in *AIModules* verwendet werden.

Das Tool ist bewusst einfach und transparent gestaltet, i. e. jeder Biologe kann seine eigene Promotorsuche durchführen — mit

einer wichtigen Einschränkung: Es ist kein KI-Tool, i. e., wenn unzusammenhängende Sequenzen verglichen werden, dann bekommt der Anwender nichts biologisch Interessantes als Ergebnis. Unsere Applikation wendet sich zwar auch an Studenten, aber es bedarf auch eines Experten oder Tutors mit Biologiewissen über Promotoren, um wichtige Erkenntnisse zu generieren.

4.7 Zukünftige Features

Das Tool AIModules ist als Schichtenmodell konzipiert und kann deshalb unkompliziert abgeändert werden. Dies betrifft nicht nur die Tatsache, dass die einzelnen Schichten durch andere Technologien ausgetauscht werden können, sondern auch die Erweiterung der Funktionalitäten im Hintergrund. Deshalb kann für zukünftige Implementierungen ins Auge gefasst werden Maschinenlernalgorithmen für die Mustererkennung mit einzubinden, denn durch die heutige Verfügbarkeit von Datenmassen sollte es möglich sein ein entsprechendes Modell zu trainieren, das für die TFBS- oder Moduleerkennung herangezogen werden kann. Dies könnte durch Frameworks wie *TensorFlow*, *PyTorch* etc. realisiert werden. Zusätzlich kann die Modulefilterung verbessert werden, indem bekannte TFBS-Kolokationen mit entsprechendem Versatz, sogenannte Module, in In-Memory-Datenbanken vorgehalten werden, die dann auf die Ergebnisse der TFBS-Suche angewandt werden.

Die Applikation ist auch als Docker Lösung verfügbar. Deshalb sollte es nicht sehr aufwendig sein das System auf einen Kubernetes Provider mit hohem Durchsatz zu deployen. Dort können auch Addons als Microservices oder als Serverless Komponenten hinzugefügt werden, um die Lastverteilung zu verbessern und ein Failover zu implementieren.

Zudem kann eine lizenzierte Version von Transfac Professional mit eingearbeitet werden, um die Qualität der Resultate zu verbessern. Als Alternative, so eine Genehmigung durch Trans-

fac vorliegt, können die öffentlich zugänglichen Transfac Matrizen mit Methoden des Data Minings extrahiert werden, um diese AI-Modules zuzuführen. Allerdings sind diese öffentlich zugänglichen Matrizen aus dem Jahre 2005 und somit veraltet.

Eine weitere Möglichkeit die Applikation zu verbessern ist die Modulesuche und das Rendering auf den Server auszulagern. Dies wurde für den Anfang nicht so realisiert, da alle drei Schichten der Applikation auf einem virtuellen Server laufen und es sonst zu Performanzminderungen gekommen wäre. Sollte AIModules in Zukunft auf potenterer Hardware betrieben werden, kann der Code der Modulesuche innerhalb von *nodejs*² auf dem Server zur Ausführung gebracht und mittels *Server-side rendering* sogar das Ergebnis als Bild direkt an den Aufrufer (Browser) übermittelt werden. Somit kann die Berechnungsgeschwindigkeit der Applikation spürbar erhöht werden.

Zudem können Lastspitzen durch die Implementierung einer *Pipeline* mittels eines *Caching-Servers* abgefangen werden. Hiermit wird ermöglicht, dass über das *First In – First Out* Prinzip Arbeitsaufträge auch bei einer hohen Last nacheinander abgearbeitet werden können.

Unser hier vorgestelltes Tool *AIModules* bietet nicht nur spannende Funktionen an, die nur kommerziellen Produkten vorbehalten sind, sondern es lädt durch eine klare Architektur und freie Lizenz zukünftige Entwickler ein neue und aufregende Funktionen einzubinden. Dabei ist uns nicht entgangen, dass unser Tool in einer Zeit der Cloud basierten Systeme für diese Welt nutzbar sein muss. Deswegen wurden Anstrengungen unternommen, um genau diese Anforderung zu erfüllen. Somit kann AIModules nicht nur klassisch auf Servern deployed werden, sondern auch lokal auf einem Notebook mit den dafür benötigten Servern, oder lokal als Dockerlösung, oder mit Anpassungen auch auf einem Kubernetes Cluster eines Cloud Providers. Neben der direkten Promotorsuche

²<https://nodejs.org/>, aufgerufen am 27. Juni 2021

sind zahlreiche weitere biologische Anwendungen durch AIModules denkbar und durch die Software ermöglicht, z.B. Matrizen für funktionelle Stellen, die aus ENCODE hervorgegangen sind oder Matrizen für Chromosomal Territories oder Chromosomale Domänen oder RNA Motive können alle durch die Software ebenso wie Promotoren identifiziert werden.

4.8 Fazit

Es gelang uns mit *AIModules* eine sehr wichtige Funktionalität zu ermöglichen, um mannigfaltige regulatorische Elemente *in silico* zu finden. Da eine Applikation mit solchen Möglichkeiten in der Vergangenheit immer wieder kommerzialisiert wurde, stellen wir durch unsere Lösung endlich eine freie und kostenlose Alternative zur Verfügung, die allen bisher zugänglichen Tools überlegen ist. Sie liefert schnelle Ergebnisse, gerade wenn auch ausreichendes biologisches Wissen beim Anwender vorhanden ist. AIModules kann flexibel Module in Promotoren erkennen, steht aber auch für andere Motive in Nukleinsäuren zur Verfügung.

Literaturverzeichnis

In der Computer-Wissenschaft greift das World Wide Web so stark um sich, das in dieser Promotionsarbeit neben den klassischen Literaturstellen teilweise auch nur Weblinks aufgeführt sind. Diese wichtigen Weblinks sind dann immer mit voller Adresse und Angabe des Zeitpunktes, zu dem sie in die Thesis eingebaut und überprüft wurden, angegeben.

- [1] Kel AE, Gössling E, Reuter I, Cheremushkin E, Kel-Margoulis OV, and Wingender E. Match: A tool for searching transcription factor binding sites in dna sequences. *Nucleic Acids Res*, 31(13):3576–9, 2003. doi:10.1093/nar/gkg585
- [2] Matrin Ankerl. How to create random colors programmatically. <http://martin.ankerl.com/2009/12/09/how-to-create-random-colors-programmatically/>, 2009. [Online; accessed October-2019].
- [3] Xie B, Jankovic BR, Bajic VB, Song L, and Gao X. Poly(a) motif prediction using spectral latent features from human dna sequences. *Bioinformatics*, 29(13):i316–25, 2013. doi:10.1093/bioinformatics/btt218
- [4] Chi-Nga Chow, Tzong-Yi Lee, Yu-Cheng Hung, Guan-Zhen Li, Kuan-Chieh Tseng, Ya-Hsin Liu, Po-Li Kuo, Han-Qin Zheng, and Wen-Chi Chang. Plantpan3.0: a new and updated resource for reconstructing transcriptional regulatory networks from chip-seq experiments in plants. *Nucleic Acids*

- Res*, 47(D1):D1155–D1163, 2019. doi:10.1093/nar/gky1081. PMID: 30395277
- [5] Chow CW, Rincón M, and Davis RJ. Requirement for transcription factor nfat in interleukin-2 expression. *Mol Cell Biol*, 19(3):2300–2307, 1999. doi:10.1128/mcb.19.3.2300
- [6] Gary D., Stormo Thomas D., Schneider Larry Gold, and Andrzej Ehrenfeucht. Use of the ‘perceptron’ algorithm to distinguish translational initiation sites in e. coli. *Nucleic Acids Research*, 10:2997–3011, 1982. <https://doi.org/10.1093/nar/10.9.2997>.
- [7] Ryan Dahl. Node.js. <https://nodejs.org/en/>, 2018. [Online; accessed 28-August-2018].
- [8] Kenneth Daily, Vishal R Patel, Paul Rigor, Xiaohui Xie, and Pierre Baldi. Motifmap: integrative genome-wide maps of regulatory motif sites for model species. *BMC Bioinformatics*, 37:495, 2011. <https://doi.org/10.1186/1471-2105-12-495>.
- [9] Inc. Docker. docker. www.docker.com, 2018. [Online; accessed 28-August-2018].
- [10] A. Khan et al. Jaspar 2018: update of the open-access database of transcription factor binding profiles and its web framework. *Nucleic Acids Res.*, 46:D260–D266, 2018. <https://doi.org/10.1093/nar/gkx1126>.
- [11] E. A. Feingold et. al. The encode (encyclopedia of dna elements) project. *Science*, 306:636–640, 2004. doi:10.1126/science.1105136
- [12] Lang F, Stournaras C, Zacharopoulou N, Voelkl J, and Alesutan I. Serum- and glucocorticoid-inducible kinase 1 and the response to cell stress. *Cell Stress*, 3(1):1–8, 2018. doi:10.15698/cst2019.01.170

- [13] Domènec Farré, Romà Roset, Mario Huerta, José E. Adsuara, Llorenç Roselló, M.Mar Albà, and Xavier Messeguer. Identification of patterns in biological sequences at the alggen server: Promo and malgen. *Nucleic Acids Res*, 31(13):3651–3653, 2003.
- [14] The Apache Software Foundation. Apache tomcat. <https://tomcat.apache.org/>, 2018. [Online; accessed 28-August-2018].
- [15] Genomatix. Explanation of scores from genomatix programs. https://www.genomatix.de/online_help/help/scores.html, 2021. [Online; accessed 14-Mai-2021].
- [16] Intrexon Bioinformatics Germany GmbH. Genomatix. <http://www.genomatix.de/>, 2018. [Online; accessed 13-November-2019].
- [17] Gnomehacker. Consensuslogo. https://en.wikipedia.org/wiki/File:LexA_gram_positive_bacteria_consensus_logo.png, 2018. [Online; accessed 31-August-2018].
- [18] Gnomehacker. Sequencelogo. https://en.wikipedia.org/wiki/File:LexA_gram_positive_bacteria_sequence_logo.png, 2018. [Online; accessed 31-August-2018].
- [19] Google. Angularjs. <https://angularjs.org/>, 2018. [Online; accessed 28-August-2018].
- [20] Roderic Guigo. An introduction to position specific scoring matrices. <http://bioinformatica.upf.edu/T12/MakeProfile.html>, 2003. [Online; accessed 31-August-2018].
- [21] Kim H and Ronai ZA. Prmt5 function and targeting in cancer. *Cell Stress*, 4(8):199–215, 2020. doi:10.15698/cst2020.08.228

- [22] Wu HC, Bulgakov VP, and Jinn TL. Pectin methylesterases: Cell wall remodeling proteins are required for plant response to heat stress. *Front Plant Sci*, 9:1612, 2018. doi:10.3389/fpls.2018.01612
- [23] Shahmuradov I. and Solovyev V. Nsite, nsiteh and nsitem computer tools for studying transcription regulatory elements. *Bioinformatics*, 31(21):3544–3545, 2015. doi:10.1093/bioinformatics/btv404
- [24] Jaspar. Jaspar. <http://jaspar.genereg.net/>, 2018. [Online; accessed 27-August-2018].
- [25] Kamieniarz-Gdula K and Proudfoot NJ. Transcriptional control by premature termination: A forgotten mechanism. *Trends Genet*, 35(8):553–564, 2019. doi:10.1016/j.tig.2019.05.005
- [26] Xiao K, Jehle F, Peters C, Reinheckel T, Schirmer RH, and Dandekar T. Ca/c1 peptidases of the malaria parasites *plasmodium falciparum* and *p. berghei* and their mammalian hosts—a bioinformatical analysis. *Biol Chem*, 390(11):1185–97, 2009. doi:10.1515/BC.2009.124
- [27] Aspal M and Zemans RL. Mechanisms of atii-to-ati cell differentiation during lung regeneration. *Int J Mol Sci*, 21(9):3188, 2020. doi:10.3390/ijms21093188
- [28] Gribskov M, McLachlan AD, and Eisenberg D. Profile analysis: Detection of distantly related proteins. *Proc Natl Acad Sci U S A*, 84(13):4355–8, 1987. doi:10.1073/pnas.84.13.4355
- [29] Sigvardsson M. Molecular regulation of differentiation in early b-lymphocyte development. *Int J Mol Sci*, 19(7):1928, 2018. doi:10.3390/ijms19071928

- [30] Wahl MC, Will CL, and Lührmann R. The spliceosome: design principles of a dynamic rnp machine. *Cell*, 136(4):701–18, 2009. doi:10.1016/j.cell.2009.02.009
- [31] Xavier Messeguer, Ruth Escudero, Domènec Farré, Oscar Nuñez, Javier Martínez, and M.Mar Albà. Promo: detection of known transcription regulatory elements using species-tailored searches. *Bioinformatics*, 18:333–334, 2002.
- [32] R. Münch, K. Hiller, H. Barg, D. Heldt, S. Linz, E. Winger, and D. Jahn. Prodoric: prokaryotic database of gene regulation. *Nucleic Acids Res.*, 31:266–269, 2003.
- [33] K. Nishida, M. C. Frith, and K. Nakai. Pseudocounts for transcription factor binding sites. *Nucleic Acids Research*, 37:939–944, 2008. <https://doi.org/10.1093/nar/gkn1019>.
- [34] Fornes O, Castro-Mondragon JA, Khan A, and et al. JaspAr 2020: update of the open-access database of transcription factor binding profiles. *Nucleic Acids Res.*, 48:D87–D92, 2019. doi: 10.1093/nar/gkz1001
- [35] University of California. Motifmap. <http://motifmap.igb.uci.edu/>, 2018. [Online; accessed 31-August-2018].
- [36] Oracle. Java. <https://www.oracle.com/technetwork/java/index.html>, 2018. [Online; accessed 28-August-2018].
- [37] D’haeseleer Patrik. What are dna sequence motifs? *Nature Biotechnology*, 24:423–425, 2006.
- [38] PlantPan3.0. Plantpan3.0. <http://plantpan.itps.ncku.edu.tw/>, 2019. [Online; accessed 19-Oktober-2019].
- [39] Prodoric. Prodoric[®] database. <http://prodoric.tu-bs.de/>, 2018. [Online; accessed 31-August-2018].

- [40] Promo. Promo. http://algggen.lsi.upc.es/cgi-bin/promo_v3/promo/promo_init.cgi?dirDB=TF_8.3, 2018. [Online; accessed 31-August-2018].
- [41] T. D. Schneider and R. M. Stephens. Sequence logos: A new way to display consensus sequences. *Nucleic Acids Res.*, 18:6097–6100, 1990. <http://dx.doi.org/10.1093/nar/18.20.6097>, <http://alum.mit.edu/www/toms/papers/logopaper/>.
- [42] T. D. Schneider, G. D. Stormo, and L. Gold. Information content of binding sites on nucleotide sequences. *Journal of Molecular Biology*, 188:415–431, 1986.
- [43] J. Schug. Using tess to predict transcription factor binding sites in dna sequence. *Curr Protoc Bioinformatics*, Chapter 2:Unit 2.6, 2008. doi:10.1002/0471250953.bi0206s21
- [44] J. Schug. Tess. <https://www.cbil.upenn.edu/tess>, 2018. [Online; accessed 30-September-2018].
- [45] Overton G.C. Schug, J. Tess: Transcription element search software on the www technical report cbil-tr-1997-1001-v0.0. 1997. doi:10.1.1.111.8401
- [46] Evgeny Smirnov, Matúš Hornáček, Tomáš Vacík, Dušan Cmarko, and Ivan Raška. Discontinuous transcription. *Nucleus*, 9:1:149–160, 2018. doi:10.1080/19491034.2017.1419112
- [47] Softberry. Softberry. <http://www.softberry.com/>, 2018. [Online; accessed 31-August-2018].
- [48] G. D. Stormo. Dna binding sites: representation and discovery. *Bioinformatics*, 16:16–23, 2000. <https://doi.org/10.1093/bioinformatics/16.1.16>.

- [49] Tair. Tair. <https://www.arabidopsis.org/tools/bulk/motiffinder/index.jsp>, 2018. [Online; accessed 31-August-2018].
- [50] Dave Tang. Transcription factor binding site prediction. <https://davetang.org/muse/2013/10/02/transcription-factor-binding-site-prediction/>, 2013. [Online; accessed 28-August-2019].
- [51] Transfac. Transfac profession vs. public. https://portal.genexplain.com/archive/documents/transfac_comparison.pdf, 2020. [Online; accessed 12-November-2020].
- [52] Matys V, Fricke E, Geffers R, Gössling E, Haubrock M, Hehl R, Hornischer K, Karas D, Kel AE, Kel-Margoulis OV, Kloos DU, Land S, Lewicki-Potapov B, Michael H, Münch R, Reuter I, Rotert S, Saxel H, Scheer M, Thiele S, and Wingender E. Transfac: transcriptional regulation, from patterns to profiles. *Nucleic Acids Res*, 31(1):374–8, 2003. doi:10.1093/nar/gkg108
- [53] Solovyev VV and Shahmuradov IA. Promh: Promoters identification using orthologous genomic sequences. *Nucleic Acids Res.*, 31(13):3540–3545, 2003.
- [54] Solovyev VV, Shahmuradov IA, and Salamov AA. Identification of promoter regions and regulatory sites. *Methods Mol Biol.*, 674:57–83, 2010. doi: 10.1007/978-1-60761-854-6_5
- [55] Wikipedia. Position weight matrix. https://en.wikipedia.org/wiki/Position_weight_matrix, 2018. [Online; accessed 31-August-2018].
- [56] Clemens Wrzodek, Adrian Schröder, Andreas Dräger, Dierk Wanke, Kenneth W. Berendzen, Marcel Kronfeld, Klaus Harter, and Andreas Zell. ModuleMaster: A new tool to decipher

- transcriptional regulatory networks. *Biosystems*, 99(1):79–81, 2010.
- [57] Clemens Wrzodek, Adrian Schröder, Andreas Dräger, and Andreas Zell. *ModuleMaster Documentation*. Center for Bioinformatics Tübingen, University of Tübingen, 2009. [Online; accessed 30-Juli-2021].
- [58] Chen X, Guo L, Fan Z, and Jiang T. Learning position weight matrices from sequence and expression data. *Comput Syst Bioinformatics Conf.*, 6:249–260, 2007.
- [59] Xiaohui Xie, Paul Rigor, and Pierre Baldi. Motifmap: a human genome-wide map of candidate regulatory motif sites. *Bioinformatics*, 25:167–174, 2009. <https://doi.org/10.1093/bioinformatics/btn605>.
- [60] Łukasz Kreft, Arne Soete, Paco Hulpiau, Alexander Botzki, Yvan Saeys, and Pieter De Bleser. Contra v3: a tool to identify transcription factor binding sites across species, update 2017. *Nucleic Acids Research*, 25:W490–W494, 2017. <https://doi.org/10.1093/nar/gkx376>
- [61] Łukasz Kreft, Arne Soete, Paco Hulpiau, Alexander Botzki, Yvan Saeys, and Pieter De Bleser. Contra v3: a tool to identify transcription factor binding sites across species, update 2017. <http://bioit2.irc.ugent.be/contra/v3/#/step/1>, 2017. [Online; accessed October-2020].

Kapitel 5

Appendix

Sequenzen für Analysen

Untenstehend sind die Sequenzen aufgeführt, die in dieser Arbeit für Analysen verwendet wurden. Diese Sequenzen finden sich auch in der Veröffentlichung, eingereicht bei BMC Bioinformatics.

Sequenz für den Vergleich mit conTraV3

```
>AJ223836.1 Chionodraco hamatus mRNA for cathepsin
CAGACGGAGAGACAGACAGACGGAGAGACACACACACACACAGACACACACACTCACAGACATGAGG
TCGGTGTGCTGCTGCTGTGTATCTGGACATGCAGGAGCTCAGCGCTGATCAGGGTGCCTCTGAGGAAG
GTGCCCACGATCCGCTCTCAGCTCCGCTCTGAAGTCTGCTGCAGGACTTCCTGGTGGAGAATCGGCCC
GACATGTTTCAGTCGCCCTACGCTCAGTGTTCCTCCCGCCGGACGCCCTCTCTGAGGCTGGGGCGCTCC
AGTGAGAAGATCTACAACCTTCATGGACGCTCAGTATTACGGTGACATCGCGTTGGGGACCCCGGAGCAA
AACTTCTCTGTGGTCTTTGACACCCGGATCGTCCGATCTGTGGGTGCCTTCGGCCTACTGCGTCAACGAA
GCCTGTGCGTTGCCAAAGCGCTTCAAGGCGTTAAATCCACGTCTTTCCTCCATGACGGTCGGCAGTTT
GGGATAAACTACGGATCAGGACACCTGCTGGGGTTCATGGGCCGAGACTACCTGATGGTGGCAGGTATG
ATGGTTAAGAGGCAGGAGTTCGGGAGTCGGTCTATGAGCCCGGCACTGCGTTTTTTGAAGGCGAGGTTT
GATGGCGTTTTGGGCTTGGGTTACCCGGCCCTGGCAGAGATCCTGGGAAACCCCGTTTTTCGACAACATG
TTGGCGCAGAACTTGTGGACAAGCCCATCTTCTCCTTCTACCTGAGCAGGAAATTAATGGAAGCCCA
GAGGGCGAGCTGTTGCTGGGCGGAACAGACGAGAGGTTGTACGACTTACCAATCAACTGGCTCCCCGTG
ACTGCTAAGGCCTACTGGCAGATCAAGATAGACAGCGTGGTGGTGCAGGGTGTGAATCCCTTCTGTCCC
CACGGCTGTGAGGCCATCGTCGACACGGAACTTCTCTCATCACTGGACCCACTGACGACATCCTGGAC
ATCCAGCAGCTGATCGGAGCCACGCCACCAACTTCGGAGAGTTCATCGTCGACTGTGCCAGGTTGTCC
AACTTTCCTCAACATCAACATTTTCGTCTCGGTGGGAAGGAGTACACTCTGACGTCCGACCAGTACATC
AGGAAGGAGATGCTCGGCGACAGGAAGTTATGCTTCAGTGGCTTCCAGGCCGTGGACATGATTTCTCC
GAAGGCCCCCTGTGGATTCTGGGAGATGTGTTTCTGACACAGTACTACAGCATTTTCGACAGAGGACAG
GACCGGTTCGGCTTCGCCATCGCTAGATAACCACTCGAAGTTTAAACGATAATACGAGAGCTGAAAAC TG
```

CAATAAATTCTACACAATGAC

Sequenzen für die Analyse von Polyadenylierungsstellen in mRNA

>NM_000600.5 Homo sapiens interleukin 6 (IL6), transcript variant 1, mRNA

```
ATTCTGCCCTCGAGCCCACCGGGAACGAAAGAGAAGCTCTATCTCCCCTCCAGGAGCCCAGCTATGAAC
TCCTTCTCCACAAGCGCCTTCGGTCCAGTTGCCTTCTCCCTGGGGCTGCTCCTGGTGTGCCTGCTGCC
TTCCCTGCCCCAGTACCCCCAGGAGAAGATTCCAAAGATGTAGCCGCCCCACACAGACAGCCACTCACC
TCTTCAGAACGAATTGACAAAACAAATTCGGTACATCCTCGACGGCATCTCAGCCCTGAGAAAGGAGACA
TGTAACAAGAGTAACATGTGTGAAAGCAGCAAAGAGGCACTGGCAGAAAACAACCTGAACCTTCCAAAG
ATGGCTGAAAAAGATGGATGCTTCCAATCTGGATTCAATGAGGAGACTTGCCTGGTAAAAATCATCACT
GGTCTTTTGGAGTTTGAGGTATACCTAGAGTACCTCCAGAACAGATTTGAGAGTAGTGAGGAACAAGCC
AGAGCTGTGCAGATGAGTACAAAAGTCCTGATCCAGTTCCCTGCAGAAAAAGGCAAAGAATCTAGATGCA
ATAACCACCCCTGACCCAACCACAAATGCCAGCCTGCTGACGAAGCTGCAGGCACAGAACCAGTGGCTG
CAGGACATGACAACTCATCTCATTCTGCGCAGCTTTAAGGAGTTCCTGCAGTCCAGCCTGAGGGCTCTT
CGGCAAATGTAGCATGGGCACCTCAGATTGTTGTTGTTAATGGGCATTCCCTTCTTCTGGTCAGAAACCT
GTCCACTGGGCACAGAACTTATGTTGTTCTCTATGGGAACTAAAAGTATGAGCGTTAGGACACTATTT
TAATTATTTTTAATTTATTAATATTTAAATATGTGAAGCTGAGTTAATTTATGTAAGTCATATTTATAT
TTTTAAGAAGTACCACTTGAAACATTTTTATGTATTAGTTTTGAAATAATAATGAAAGTGGCTATGCAG
TTTGAATATCCTTTGTTTCAGAGCCAGATCATTCTTGAAAGTGTAGGCTTACCTCAAATAAATGGCT
AACTTATACATATTTTTAAAGAAATATTTATATTGTATTTATATAATGTATAAATGGTTTTTATACCAA
TAAATGGCATTTTAAAAAATTCA
```

>NM_000594.4 Homo sapiens tumor necrosis factor (TNF), mRNA

```
AGCAGACGCTCCCTCAGCAAGGACAGCAGAGGACCAGCTAAGAGGGAGAGAAGCAACTACAGACCCCCC
CTGAAAACAACCCTCAGACGCCACATCCCCTGACAAGCTGCCAGGCAGGTTCTTCTCCTCACATACT
GACCCACGGCTCCACCCTCTCTCCCCTGAAAGGACACCATGAGCACTGAAAGCATGATCCGGGACGTG
GAGCTGGCCGAGGAGCGCTCCCCAAGAAGACAGGGGGGCCCCAGGGCTCCAGGCGGTGCTTGTTCCCTC
AGCCTCTTCTCCTTCCTGATCGTGGCAGGCGCCACCACGCTCTTCTGCCTGCTGCACTTTGGAGTGATC
GGCCCCCAGAGGGAAGAGTTCCCAGGGACCTCTCTAATCAGCCCTCTGGCCCAGGCAGTCAGATCA
TCTTCTCGAACCCCGAGTGACAAGCCTGTAGCCCATGTTGTAGCAAACCCTCAAGCTGAGGGGCAGCTC
CAGTGGCTGAACCGCCGGGCAATGCCCTCCTGGCCAATGGCGTGGAGCTGAGAGATAACCAGCTGGTG
GTGCCATCAGAGGGCCTGTACCTCATCTACTCCCAGGTCCTCTTCAAGGGCCAAGGCTGCCCTCCACC
CATGTGCTCCTCACCCACACCATCAGCCGCATGCGCGTCTCCTACCAGACCAAGGTCAACCTCCTCTCT
GCCATCAAGAGCCCCTGCCAGAGGGAGACCCCAGAGGGGGCTGAGGCCAAGCCCTGGTATGAGCCCATC
TATCTGGGAGGGTCTTCCAGCTGGAGAAGGGTGACCGACTCAGCGCTGAGATCAATCGGCCCGACTAT
CTCGACTTTGCCGAGTCTGGGCAGGTCTACTTTGGGATCATTGCCCTGTGAGGAGGACGAACATCCAAC
CTTCCCAAACGCCTCCCCTGCCCAATCCCTTTATTACCCCTCCTTCAGACACCCTCAACCTCTTCTG
GCTCAAAAAGAGAATTGGGGGCTTAGGGTCGGAACCCAAGCTTAGAAGCTTTAAGCAACAAGACCACCAC
TTGAAACCTGGGATTCAGGAATGTGTGGCTGCACAGTGAAGTGTGGCAACCACTAAGAATTCAAAC
TGGGGCCTCCAGAACTCACTGGGGCTACAGCTTTGATCCCTGACATCTGGAATCTGGAGACCAGGGAG
CCTTTGGTTCTGGCCAGAATGCTGCAGGACTTGAGAAGACCTCACCTAGAAATTGACACAAGTGGACCT
```

TAGGCCTTCCTCTCTCCAGATGTTTCCAGACTTCCTTGAGACACGGAGCCCAGCCCTCCCCATGGAGCC
 AGCTCCCTCTATTTATGTTTGCACCTTGATTTATTATTTATTTATTTATTTATTTATTTATTTACAGATG
 AATGTATTTATTTGGGAGACCGGGGTATCCTGGGGGACCCAATGTAGGAGCTGCCTTGGCTCAGACATG
 TTTTCCGTGAAAACGGAGCTGAACAATAGGCTGTTCATGTAGCCCCCTGGCCTCTGTGCCTTCTTTT
 GATTATGTTTTTTAAAAATTTTATCTGATTAAGTTGTCTAAACAATGCTGATTTGGTGACCAACTGTCA
 CTCATTGCTGAGCCTCTGCTCCCCAGGGGAGTTGTGTCTGTAATCGCCCTACTATTTCAGTGGCGAGAAA
 TAAAGTTTGCTTAGAAAAGAAA

>NM_020525.5 Homo sapiens interleukin 22 (IL22), mRNA

ACAAGCAGAATCTTCAGAACAGGTTTCCTTCCCAGTCACCAGTTGCTCGAGTTAGAATTGTCTGCAA
 TGGCCGCCCTGCAGAAATCTGTGAGCTCTTTCCTTATGGGGACCCTGGCCACCAGCTGCCTCCTTCTCT
 TGGCCCTCTTGGTACAGGGAGGAGCAGCTGCGCCCATCAGCTCCCCTGCAGGCTTGACAAGTCCAACCT
 TCCAGCAGCCCTATATCACCAACCGCACCTTCATGCTGGCTAAGGAGGCTAGCTTGGCTGATAACAACA
 CAGACGTTTCGTCTCATTGGGGAGAACTGTTCCACGGAGTCAGTATGAGTGAGCGCTGCTATCTGATGA
 AGCAGGTGCTGAACTTCACCCTTGAAGAAGTGTGTTCCCTCAATCTGATAGGTTCCAGCCTTATATGC
 AGGAGGTGGTGCCTTCTGGCCAGGCTCAGCAACAGGCTAAGCACATGTCATATTGAAGGTGATGACC
 TGCATATCCAGAGGAATGTGCAAAAGCTGAAGGACACAGTGAAAAAGCTTGGAGAGAGTGAGAGATCA
 AAGCAATTGGAGAACTGGATTTGCTGTTTATGTCTCTGAGAAATGCCTGCATTTGACCAGAGCAAAGCT
 GAAAAATGAATAACTAACCCCTTTCCCTGCTAGAAAATAACAATTAGATGCCCAAAGCGATTTTTTTTT
 AACCAAAAGGAAGATGGGAAGCCAAACTCCATCATGATGGGTGGATTCCAAATGAACCCCTGCGTTAGT
 TACAAAGGAAACCAATGCCACTTTTTGTTTATAAGACCAGAAGGTAGACTTTCTAAGCATAGATATTTAT
 TGATAACATTTTCATTGTAAGTGGTGTCTATACACAGAAAACAATTTATTTTTAAATAATTGTCTTTT
 TCCATAAAAAAGATTACTTTCCATTCCCTTAGGGGAAAAAACCCCTAAATAGCTTCATGTTTCCATAAT
 CAGTACTTTATATTTATAAATGTATTTATTATTATAAGACTGCATTTTATTTATATCATTTTATTA
 ATATGGATTTATTTATAGAAACATCATTCGATATTGCTACTTGAGTGTAAAGGCTAATATTGATATTTAT
 GACAATAATTATAGAGCTATAACATGTTTATTTGACCTCAATAAACACTTGGATATCCTAA

Sequenzen für die Analyse von NFAT [3] durch AIModules

>NC_000067.6:131019345-131019844 Mus musculus strain C57BL/6J chromosome 1, GRCh38.p6 C57BL/6J, Mus musculus interleukin 10 (Il10) promoter
 AAATCAGCCCTCTCGGGTTTCCTTTGGGTAAGTGAAGTACTTCCGAGTCAGCAAGAAATA
 TCGGACGTTCAACCCAGGTTGAGTGGAGGAAACAATTATTTCTCAATCCTAATATGTTCTGGAATAGCC
 CATTATCCACGTCATTATGACCTGGGAGTCCGTGAATGGAATCCACAGATGAGGGCCTCTGTACATAG
 AACAGCTGTCTGCCTCAGGAAATACAACCTTTTAGTATTGAGAAGCTAAAAAGAAAAAAAAAATTTAAAGA
 GAGGTAGCCATACTAAAAATAGCTGTAATGCAGAAGTTCATTCCGACCAGTCTTTAGCGCTTACAAT
 GCAAAAAAAAAAGGAAAGGAAAAAAAAAAAAAGAAAGAAATTAACCTCAAAAATTGCATGGTTTGAAGAGG
 GAGGAGGAGCCTGAATAACAAAAACCTTTGCCAGGAAGGCCCCACTGAGCCTTCAGTATAAAAGGGGA
 CCAAGAACAGGAGGTCT

>NC_000074.6:c82403252-82402576 Mus musculus strain C57BL/6J chromosome 8, GRCh38.p6 C57BL/6J, Mus musculus interleukin 15 (Il15), transcript variant 2 promoter
 ACCCAAGTCCCTTTCCGTTTCCCTCCCGGTACAGTCTCTGTCCAGATCTCCTCCGGGCTTCTATGGGGA

AGCCAAACTGCCTCCCTGCAAGGCCAGTTGCTGTAGATGCAAGCAACCTGTGAACTCAGGCCAACTCCT
 TGAAACTTCACAGAGGCAAAGGCATTCCAGGACACACAGAGGCTGTGGCCAACTGCCAGGGGAGGAG
 ACTGCTCTCTGCTCTCAGTTGCCCTTCAGGTTCTGCGCCCTGGGGACCTGGCAGTGGCAGAATCATGTG
 GGCACCTGGTAAGGTGCCGGGACCATGTGTTCTCCCGCAACCTCCCAGGGTGTGATCTCCGCCTC
 AGCTTGGGCTCTTTCTTTCACTTTTCTGTTAGCTGGGGTTGGGACTCCCCGGCTGGAAAGCACTGGG
 GGAAACCGGGGAAACCCAGCTGATTCGCTCCTTGTGCCTTGATTGCTCCCGCTGGCTGCTGCCCTGCA
 TCCTGCACCCTTCAACCAGAACCCGATGGAGGTACAGAATGGGAGGTGGTAGTGCTGGTGGTGGTGGAT
 CAACAATGGAATTTTTTTTTTTTTCCGAAAGCCTACGCCCCGGGCCCTCCAGCTCTGGCTCTGCTCAG
 GCACCCTTTTCCCCTCAGCTGCCGGCCAGGCCCGCCCTCT
 >NC_000077.6:c53635202-53634702 Mus musculus strain C57BL/6J chromo-
 some 11, GRCh38.p6 C57BL/6J, Mus musculus interleukin 13 (Il13) promoter
 CATGCATTGCTTTGGTGATTTATCAGATACGTTTGTGAAATTAAGAAGAAAAAAGAAAAGAAAAGAAA
 GAAGAAAGGAAGGAAGGGAGGAAGGAAGAAAGAAAAGAAAGGAGAAAACTCCCAATTCTCACAGCTAA
 TAGACTGATACTCACCTGCCAAAGGGTGACAAGTACTTAGTGATCAAGGGGTCAGCATTGGGCTGGCT
 GCTCAGGAGCTTGGGGCGGTGACGGGTGGAATTAAGGGGGCGAAGTTAGCTTTGCTGATGCCACCG
 TGGAATAAACCACCCAGAACCTGGAACCTGTCCAGACCCTTCTCAATAAATCCACTAAATCAGAC
 TCTTTCTTTAGCGGCACTGGATTTTCCAAAAAAGAAAAAATAAATCAAGATGAGTAAAGATGTGG
 TTTTCAGATAATGCCAACAAAGCAGAGACCAGGGGTGAGGCGTCATCACTTTGGTTTATAAAAGCTGC
 TTCAACAGGCTAAGGCC

Sequenzen für die Analyse von Genomatix vs AImodules

Cathepsine:

>Homo_sapiens_cathepsin_V_transcript_variant-1_promoter
 CTTCTAAAGCCCTGATGCACCCACCCTTAAGTGAAGGTAATGCTTCTGCTGGGGGCTGTGCGTTCTTC
 CACTCCGTCCCACCTTTGCCGTGGACTAAACAGGAGCCACTGAACGAGAGTACCCTGGCTCCCGGCCCT
 GCAGATTTTACCAAAAACCTCTAGGACTAAGTTAAAGTAGGGACTGAAATACCACACAGAGCTCGCTGT
 AAACCACCCGCTCAGCGCAGTAAGTCGGGATAGTCAGTAGATCTGAGCCCTTGAGGGAGGCGCCTCCCC
 CTCCTCCGCGCAGCCCCGGGCGCCCGGCCAGCTTCCCGCAGCCAGGGTGTATTGAGGTAGGCGCGCC
 CAGACCTGAGACGGGTTGGGACTGGGCTGCGTCACGCGCGGGCTCTAAGCGCCCGGGGCCCGCCAGT
 GGCCGGCACAGCCAATCGCAGCGGGAAGGCGGTGGGGCGGGGAAGCCGCTGGAAACTTAAATCC
 CGAGGCGGGCGAACCTG
 > Bos_taurus_cathepsin-Z_promoter
 CTAGGGTGCTCAGGTGCTCGGTCATATCTGACTCTTTTGCAACCCCTGGGACTGTAGCCTGCCAAGCTC
 CTTTGTCAATGGGATTCTCCAGGGAAGAATACTGGAGTGGGTTGCCATGCCCTCCTCCATCAGGAAGCT
 AGAATAGGGTGCAGTTCTGCAGAGAGACCCCTGCCATCATCCCTGGAGGCTGGACCCAGGCGGGATGC
 GGGGATCCGGGGCATCTCTCCGACCAAAGGTCCCAAGACGCAAGGCTGGGGGTGGGGGGCCCTAAG
 AATGAATGAGGTCTGATAGATCAGAGGCAAGAGGCAAGTGGCGGGTTCGGGGCGGGTCCGCCGAGGCC
 TGGCGCGTGACCCGAGTCCGGACGGGCGGGCAGCCGGGCGGGCGGGGCGGGTACCCGGGGCAGCA
 GCGCGCCCCGCCCGGGGAGTGTCCCGCGGGCGCAGACACCTTAAGGGCTGCGGGCGGGGAGGAA
 GCCTGCGGAGGGGTGAC
 > Mus_musculus_cathepsin-F-transcript-variant-X1_promoter_576-1076
 GAGCCACCTTGCAATGATCCCCTGGTGTGCCACTGCCTGAGACGAAGAAAACAGTGGTGAAGAG

GTCAGCTATTGTTGGCCCTAGCTGGCAAACAATTGGAATGGTGGGTCCTGTCGAGGCATTGGTTACTCC
TGAGATACCAAGTTAGAGACCAAGATGCTCTGAAAAGGTGGGTGTTGGGGTGCTGGGTGTCTGCTGCTG
GCCTCCCTCTTTCACCAAGTTATGACTGCCTCCATGTGGCTCTCCTTTGAGGAACTTCTAGCTGCTCT
TGCCAGGACCTTGTTCAAAAGTGTCCCAAGACCGTCTTCTTTCTATGCCTCCAGCTCTGCAGTTTTGAA
GTCCTGGAAGAGCTAAAAGAACAAGTGTGCTGCTGAGGAGGGACTGTAGCCCAGTGAATGCCAAGGTCACA
GGTGTGGGATTGCTCAAGGAGACTGGGAGAGCCCAGACCATACTTGAGTCAGAATCCAGCCTTAATTA
CTTCTCTGTTCCGACCCT

IL-10:

>X73536.1_H.sapiens_promoter_region_of_human_IL-10_gene
AGCTTTCAGCAAGTGCAGACTACTCTTACCCACTTCCCCAAGCACAGTTGGGGTGGGGGACAGCTGAA
GAGGTGGAACATGTGCCTGAGAATCCTAATGAAATCGGGGTAAAGGAGCCTGGAACACATCCTGTGAC
CCCGCCTGTCTGTAGGAAGCCAGTCTCTGGAAAGTAAAATGGAAGGGCTGCTTGGGAACTTTGAGGAT
ATTTAGCCCACCCCTCATTTTTACTTGGGGAACTAAGGCCAGAGACCTAAGGTGACTGCCTAAGTT
AGCAAGGAGAAGTCTTGGGTATTCATCCCAGTTGGGGGACCCAATTATTTCTCAATCCCATTGTATT
CTGGAATGGGCAATTTGTCCACGTCACTGTGACCTAGGAACACGCGAATGAGAACCACAGCTGAGGGC
CTCTGCGCACAGAACAGCTGTTCTCCCAGGAAATCAACTTTTTTTAATTGAGAAGCTAAAAAATTATT
CTAAGAGAGGTAGCCCATCCTAAAAATAGCTGTAATGCAGAAGTTCATGTTCAACCAATCATTTTTGCT
TACGATGCAAAAATTGAAAATAAGTTTATTAGAGAGGTTAGAGAAGGAGGAGCTTAAGCAGAAAAAA
TCCTGTGCCGGGAAACCTTGATTGTGGCTTTTTAATGAATGAAGAGGCCTCCCTGAGCTTACAATATAA
AAGGGGACAGAGAGGTGAAGGTCTACACATCAGGGGCTTGCTCTTGCAAAACCAACCACAAGACAGA
CTTGCAAAAGAAGGCATGCACAGCTCAGCACTGC
>AY486432.1_Macaca-mulatta_interleukin-10-(IL-10)_gene_promoter_region
TTTGGGAAGGGGAAGTAGGGATAGGTAAGAGGAAAGTAAGGGACCTCCTATCCAGCCTCCATGGAATCC
TGACTTCTTTTCCCTGTTATTACAACCTCTTCCCTCCCATCTTTTAACTTTAGACTCCCGCCACAGAA
GCTTACAACATAAAAGAACTCTAAGGCCAATTTAATCCAAGTTTCATTCTATGTGTTGGAGATGGCAT
AGAGTAGGGTGAAGGAAACCAAACTTCTCAGTTGGCACTGGTGTACCCTTGTACAGGTGATGTAATCGCTC
TGTGCCTCAGTTTGTCTACTATAAAATAGAGATGGTAGGGGTGATGGTGTGAGCACTACCTGACTAGCATA
TAACAAGCTTTTCGGCAAGTGCAGACTACTCTTACCCACTTCCCCAAGCACAGTTGGGGTGGGGACAC
CTGGGACAGCTGAAGAGGAGGAAATGTGTGCCTGAGAATCCAAATGAAATCAGGGTAAAGGAGCCTGGA
GCACATCCTGTGACCTGCCTGTCTGTAGGAAGCCGGTCTCTGGAAGGTAAAATGGAAGAGCTGCTTG
GGAGCTTTGAGGATATTTAGCCACCCCTCATTTTTACTTGGGGAACTAAGGCCACAGACCTAAGG
TGACTGTCTAAGTTAGCAAGAAGAAGTCTTGGGTATTCACCCTGGGTGGGGGACCCAATTATTTCTC
AATCCCATTGTATTCTGGAATGGGCAATTTGTCCACGTCACTGTGACCTAGGAACACGCGAATGGGAAC
CCACAACCTGCGGGCTCTGCGCACAGAACAGCTGTTCTCCCAGGAAATCAACTTTTTTTAATTGAGAA
GCTAAAAAATTATTCTAAGAGAGGTAGCCATCCTAAAAATAGCTGTGCAGAAGTTCATGTTCAACCAA
TCCTTTTTGCTTACGATGCAAAATTTGAAAATAAGTTTATTAGAGAGGTTAGAGAAGGAGGAGCTCTA
AGCAGAAAAAATCCTGTGCCGGGAAACCTGTGATTGTGGCTTTTTATGAATGAAGAGGCCTCCCTGAGC
TTACAATATAAAAGGGGACAGAGAGGTGAAGTCTACACATCAGGGGCTTGCTCTTGCAAAACCAAAC
CACAAGACAGACTTGCAAAAGAAGGCATGCACAGCTCAGCACTGCTCTGTTGCCTGGTCTCC
>AF121965.1_Mus-musculus_interleukin-10-(IL10)_gene_promoter_partial_
sequence
TTCCTGTTCTACCAGCCTGGTGTGGTAACCCTCTCCAATGGGGCAGGCTTGAACCTGTGCCAACGA

AGATCCTCCCCGTA CTGATGCAGGAAGGACAGCCCGGGAGTGTACCCTCTACATGGGTCTACTTTTAT
TTAAGCAAACATTCCTGGTCAACAGGACGTGTAGCATTGCCCCCCCCCTGGGTACACAGAAAACA
GGTACCAGGAGGACAAGTAGTTGCTTGCCAGGGTACAGAATGAAAGGCAATAGGGGACTCTAGGCGAA
TGTTCTTCCCACCAA AACTGAGGTAGTAGGAGAAGTCCCTACTGAAGGGAAGGTCCAGACATAATCAAA
GGACTACCAGAGATCTCCAAT

Homo sapiens Immunglobulin

> Homo sapiens chromosome 14, GRCh38.p13 Primary Assembly IGH
105588394-105588894

TGGGGTGGGGCTCATGGAGTGGTGGGTGTTGGACTGAGACTCTGACCAGGGACAGGGGGATGGGGTCAC
AGCCAAGCCACTCCACCCCTACCCCATGCACACAGCACTCAGAGCCCAGGCCCTCCTCAGAGCCCC
ACAAAATCCTCTCTAGGGGCAGGGGAAAGAGCAAGACATGTCCCCACCCAGAGCAGGAACTGGGGTC
AGGGAGCTCAGGGGACTCAGCCACTCCATGGCAGAGCCCTGTTTAAATAA AACTTGTGTCTGGGATGGCC
TGGGTGAGAGGCCCTATCTAAGGAGCATGTTCAGAAACTGTGTGCTGGGATGAGACAGCTGGGTCCAA
CCGAGGCCCATGTTGCAGGAGCTGTGTAACCTGGGGCTGTCACCAGGCCTCTCTGTGCTGGGTTCCT
CCAGTGTAGAGGAGGAGGCAGGTACAGCCTGTCTCCTGGGGACATGGCATGAGGGCCGCTCCTCACAG
CGCATTCTGTGTTCCAGC

> Homo sapiens chromosome 2, GRCh38.p13 Primary Assembly IGKC
88857683-88858183

GGGCTCAGGGCCTGCTCTGCAGGGAGTTTTAGCCCAGCCAGCCAAAGTAACCCCGGGAGCCTGTTA
TCCCAGCACAGTCTTGAAGAGGCACAGGGGAAATAAAAAGCGGACGGAGGCTTTCCTTGACTCAGCCGC
TGCCTGGTCTTCTCAGACCTGTTCTGAATTCTAAACTCTGAGGGGGTCCGATGACGTGGCCATTCTTT
GCCTAAAGCATTGAGTTTACTGCAAGGTCAGAAAAGCATGCAAAGCCCTCAGAATGGCTGCAAAGAGCT
CCAACAAAACAATTTAGAACTTTATTAAGGAATAGGGGAAAGCTAGGAAGAAACTCAAAACATCAAGAT
TTTAAATACGCTTCTTGGTCTCCTTGCTATAATTATCTGGGATAAGCATGCTGTTTTCTGTCTGTCCCT
AACATGCCCTGTGATTATCCGCAAACAACACACCCAAGGGCAGAACTTTGTTACTTAAACACCATCCTG
TTTGCTTCTTTCCTCAGG

> Homo sapiens chromosome 1, GRCh38.p13 Primary Assembly PIGR
206946466-206946966

TATAAGGGGCATTTTTGTCCAGGCTAAGTAACCTAGGGAGTCGGAGGGGATTCCAGAGCAACTGGGGAT
ATGAGACCAAGGACTACGACAGCCACTCCTGCCACCTGTGCCCCATCAGATGATGTCAACTTCAAATCA
AGCATTGGGCCAGGTATTTTAGAGCTAATACCGGGCTATATCCTCTACCTGTAGATTTGGTATTACCAT
CCCTCTTTTCCAGATGAAAAGAAATAGGAAGGTGACTTGCCAAAGGTCTTGCAGCTAGAAAGCGACAGA
ACAGCATCTTACGCTTGACATTCTGTCCCTCATCCTGAAGCTGCAACGATGGAGGATTCCCAAGTAAC
AGAGTCTCCCCAAGGTCAAAGGAAACCAAATGGAGCCAGCCAGGAAGGCCAAAATGAAAGGAAAGCAAG
GGATCTGTGAGAGTCACATGACCCTGGCTGGCCACGGTGCCTGTGGGAGAGTGGCCCTTTAAGAGCCCA
GGTGTGGGTCAAACACTG

Interleukin Module und TFBSs Suche

Untenstehend die Tabelle der gefundenen Module für die Interleukinpromotoren (* siehe Tabelle 1.2)

Sequence Name	Module	Module Start	Module End	Sense
NC_000067.6:131019345-131019844 Mus musculus strain C57BL/6J chromosome 1; GRCm38.p6 C57BL/6J; Mus musculus interleukin 10 (Il10) promoter	MA0868.2 SOX8::MA0152.1 NFATC2	98	148	N
	MA0868.2 SOX8::MA1525.1 NFATC4	98	183	N
	MA0152.1 NFATC2::MA1621.1 Rbpjl	142	219	N
	MA0152.1 NFATC2::MA1620.1 Ptf1a(var.3)	142	218	N
	MA1621.1 Rbpjl::MA1620.1 Ptf1a(var.3)	206	218	N
	MA0868.2 SOX8::MA1100.2 ASCL1	98	217	N
	MA0152.1 NFATC2::MA1100.2 ASCL1	142	217	N
	MA1525.1 NFATC4::MA1100.2 ASCL1	174	217	N
	MA1621.1 Rbpjl::MA1100.2 ASCL1	206	217	N
	MA1620.1 Ptf1a(var.3)::MA1100.2 ASCL1	207	217	N
	MA0152.1 NFATC2::MA0868.2 SOX8	142	347	N
	MA1525.1 NFATC4::MA0868.2 SOX8	174	347	N
	MA1621.1 Rbpjl::MA0868.2 SOX8	206	347	N
	MA1620.1 Ptf1a(var.3)::MA0868.2 SOX8	207	347	N
	MA1100.2 ASCL1::MA0868.2 SOX8	208	347	N
	MA1621.1 Rbpjl::MA1525.1 NFATC4	206	368	N
	MA1620.1 Ptf1a(var.3)::MA1525.1 NFATC4	207	368	N
	MA1100.2 ASCL1::MA1525.1 NFATC4	208	368	N
	MA0868.2 SOX8::MA1525.1 NFATC4	338	368	N
	MA0152.1 NFATC2::MA0528.2 ZNF263	142	421	N
	MA1621.1 Rbpjl::MA0528.2 ZNF263	206	421	N
MA1620.1 Ptf1a(var.3)::MA0528.2 ZNF263	207	421	N	

Sequence Name	Module	Module Start	Module End	Sense
MA1100.2 ASCL1::MA0528.2 ZNF263		208	421	N
MA0152.1 NFATC2::MA0528.2 ZNF263		142	424	N
MA1621.1 Rbpjl::MA0528.2 ZNF263		206	424	N
MA1620.1 Ptf1a(var.3)::MA0528.2 ZNF263		207	424	N
MA1100.2 ASCL1::MA0528.2 ZNF263		208	424	N
MA0528.2 ZNF263::MA0528.2 ZNF263		410	424	N
MA1525.1 NFATC4::MA0868.2 SOX8		174	438	N
MA1621.1 Rbpjl::MA0868.2 SOX8		206	438	N
MA1620.1 Ptf1a(var.3)::MA0868.2 SOX8		207	438	N
MA1100.2 ASCL1::MA0868.2 SOX8		208	438	N
MA1525.1 NFATC4::MA0868.2 SOX8		359	438	N
MA0528.2 ZNF263::MA0868.2 SOX8		410	438	N
MA0528.2 ZNF263::MA0868.2 SOX8		413	438	N
MA1525.1 NFATC4::MA0867.2 SOX4		174	439	N
MA1621.1 Rbpjl::MA0867.2 SOX4		206	439	N
MA1620.1 Ptf1a(var.3)::MA0867.2 SOX4		207	439	N
MA1100.2 ASCL1::MA0867.2 SOX4		208	439	N
MA0868.2 SOX8::MA0867.2 SOX4		338	439	N
MA1525.1 NFATC4::MA0867.2 SOX4		359	439	N
MA0528.2 ZNF263::MA0867.2 SOX4		410	439	N
MA0528.2 ZNF263::MA0867.2 SOX4		413	439	N
MA0868.2 SOX8::MA0867.2 SOX4		429	439	N
MA0152.1 NFATC2::MA0597.1 THAP1		142	450	N
MA1621.1 Rbpjl::MA0597.1 THAP1		206	450	N
MA1620.1 Ptf1a(var.3)::MA0597.1 THAP1		207	450	N
MA1100.2 ASCL1::MA0597.1 THAP1		208	450	N
MA0528.2 ZNF263::MA0597.1 THAP1		410	450	N
MA0528.2 ZNF263::MA0597.1 THAP1		413	450	N
MA0868.2 SOX8::MA0816.1 Ascl2		98	217	N

Sequence Name	Module	Module Start	Module End	Sense
	MA0152.1 NFATC2::MA0816.1 Ascl2	142	217	N
	MA1525.1 NFATC4::MA0816.1 Ascl2	174	217	N
	MA1621.1 Rbpjl::MA0816.1 Ascl2	206	217	N
	MA1620.1 Ptf1a(var.3)::MA0816.1 Ascl2	207	217	N
	MA0816.1 Ascl2::MA0868.2 SOX8	208	347	N
	MA0816.1 Ascl2::MA1525.1 NFATC4	208	368	N
	MA0816.1 Ascl2::MA0868.2 SOX8	208	438	N
	MA0816.1 Ascl2::MA0867.2 SOX4	208	439	N
	MA0868.2 SOX8::MA1472.1 BHLHA15(var.2)	98	217	N
	MA0152.1 NFATC2::MA1472.1 BHLHA15(var.2)	142	217	N
	MA1525.1 NFATC4::MA1472.1 BHLHA15(var.2)	174	217	N
	MA0152.1 NFATC2::MA0624.1 NFATC1	129	183	R
	MA0152.1 NFATC2::MA0521.1 Tcf12	129	217	R
	MA0624.1 NFATC1::MA0521.1 Tcf12	174	217	R
	MA0152.1 NFATC2::MA0624.1 NFATC1	129	368	R
	MA0624.1 NFATC1::MA0624.1 NFATC1	174	368	R
	MA0521.1 Tcf12::MA0624.1 NFATC1	207	368	R
	MA0624.1 NFATC1::MA0152.1 NFATC2	174	367	R
	MA0521.1 Tcf12::MA0152.1 NFATC2	207	367	R
	MA0624.1 NFATC1::MA0152.1 NFATC2	359	367	R
	MA0152.1 NFATC2::MA0599.1 KLF5	129	421	R
	MA0624.1 NFATC1::MA0599.1 KLF5	174	421	R
	MA0521.1 Tcf12::MA0599.1 KLF5	207	421	R
	MA0624.1 NFATC1::MA0599.1 KLF5	359	421	R
	MA0152.1 NFATC2::MA0599.1 KLF5	361	421	R
	MA0152.1 NFATC2::MA0625.1 NFATC3	129	368	R
	MA0624.1 NFATC1::MA0625.1 NFATC3	174	368	R
	MA0521.1 Tcf12::MA0625.1 NFATC3	207	368	R
	MA0625.1 NFATC3::MA0152.1 NFATC2	359	367	R

Sequence Name	Module	Module Start	Module End	Sense
NC_000074.6:c82403252-82402576 Mus musculus strain C57BL/6J chromosome 8; GRCm38.p6 C57BL/6J; Mus musculus interleukin 15 (Il15); transcript variant 2 promoter	MA0625.1 NFATC3::MA0599.1 KLF5	359	421	R
	MA0152.1 NFATC2::MA0528.2 ZNF263	12	208	N
	MA0152.1 NFATC2::MA0597.1 THAP1	12	234	N
	MA0528.2 ZNF263::MA0597.1 THAP1	197	234	N
	MA0152.1 NFATC2::MA1621.1 Rbpjl	12	288	N
	MA0528.2 ZNF263::MA1621.1 Rbpjl	197	288	N
	MA0152.1 NFATC2::MA1620.1 Ptf1a(var.3)	12	287	N
	MA0528.2 ZNF263::MA1620.1 Ptf1a(var.3)	197	287	N
	MA1621.1 Rbpjl::MA1620.1 Ptf1a(var.3)	275	287	N
	MA0152.1 NFATC2::MA1100.2 ASCL1	12	286	N
	MA0528.2 ZNF263::MA1100.2 ASCL1	197	286	N
	MA1621.1 Rbpjl::MA1100.2 ASCL1	275	286	N
	MA1620.1 Ptf1a(var.3)::MA1100.2 ASCL1	276	286	N
	MA0152.1 NFATC2::MA0597.1 THAP1	12	317	N
	MA0528.2 ZNF263::MA0597.1 THAP1	197	317	N
	MA1621.1 Rbpjl::MA0597.1 THAP1	275	317	N
	MA1620.1 Ptf1a(var.3)::MA0597.1 THAP1	276	317	N
	MA1100.2 ASCL1::MA0597.1 THAP1	277	317	N
	MA1621.1 Rbpjl::MA1100.2 ASCL1	275	478	N
	MA1620.1 Ptf1a(var.3)::MA1100.2 ASCL1	276	478	N
	MA0152.1 NFATC2::MA0597.1 THAP1	12	482	N
	MA1621.1 Rbpjl::MA0597.1 THAP1	275	482	N
	MA1620.1 Ptf1a(var.3)::MA0597.1 THAP1	276	482	N
	MA1100.2 ASCL1::MA0597.1 THAP1	277	482	N
	MA1100.2 ASCL1::MA0597.1 THAP1	469	482	N

Sequence Name	Module	Module Start	Module End	Sense
	MA0528.2 ZNF263::MA0868.2 SOX8	197	561	N
	MA0597.1 THAP1::MA0868.2 SOX8	226	561	N
	MA1621.1 Rbpjl::MA0868.2 SOX8	275	561	N
	MA1620.1 Ptf1a(var.3)::MA0868.2 SOX8	276	561	N
	MA1100.2 ASCL1::MA0868.2 SOX8	277	561	N
	MA0597.1 THAP1::MA0868.2 SOX8	309	561	N
	MA1100.2 ASCL1::MA0868.2 SOX8	469	561	N
	MA0597.1 THAP1::MA0868.2 SOX8	474	561	N
	MA0528.2 ZNF263::MA0867.2 SOX4	197	562	N
	MA0597.1 THAP1::MA0867.2 SOX4	226	562	N
	MA1621.1 Rbpjl::MA0867.2 SOX4	275	562	N
	MA1620.1 Ptf1a(var.3)::MA0867.2 SOX4	276	562	N
	MA1100.2 ASCL1::MA0867.2 SOX4	277	562	N
	MA0597.1 THAP1::MA0867.2 SOX4	309	562	N
	MA1100.2 ASCL1::MA0867.2 SOX4	469	562	N
	MA0597.1 THAP1::MA0867.2 SOX4	474	562	N
	MA0868.2 SOX8::MA0867.2 SOX4	552	562	N
	MA0528.2 ZNF263::MA1525.1 NFATC4	197	566	N
	MA1621.1 Rbpjl::MA1525.1 NFATC4	275	566	N
	MA1620.1 Ptf1a(var.3)::MA1525.1 NFATC4	276	566	N
	MA1100.2 ASCL1::MA1525.1 NFATC4	277	566	N
	MA0597.1 THAP1::MA1525.1 NFATC4	309	566	N
	MA1100.2 ASCL1::MA1525.1 NFATC4	469	566	N
	MA0597.1 THAP1::MA1525.1 NFATC4	474	566	N
	MA0868.2 SOX8::MA1525.1 NFATC4	552	566	N
	MA0528.2 ZNF263::MA0152.1 NFATC2	197	578	N
	MA0597.1 THAP1::MA0152.1 NFATC2	226	578	N
	MA1621.1 Rbpjl::MA0152.1 NFATC2	275	578	N
	MA1620.1 Ptf1a(var.3)::MA0152.1 NFATC2	276	578	N

Sequence Name	Module	Module Start	Module End	Sense
MA1100.2 ASCL1::MA0152.1 NFATC2		277	578	N
MA0597.1 THAP1::MA0152.1 NFATC2		309	578	N
MA1100.2 ASCL1::MA0152.1 NFATC2		469	578	N
MA0597.1 THAP1::MA0152.1 NFATC2		474	578	N
MA0868.2 SOX8::MA0152.1 NFATC2		552	578	N
MA1525.1 NFATC4::MA0152.1 NFATC2		557	578	N
MA0528.2 ZNF263::MA0152.1 NFATC2		197	634	N
MA0597.1 THAP1::MA0152.1 NFATC2		226	634	N
MA1621.1 Rbpjl::MA0152.1 NFATC2		275	634	N
MA1620.1 Ptf1a(var.3)::MA0152.1 NFATC2		276	634	N
MA1100.2 ASCL1::MA0152.1 NFATC2		277	634	N
MA0597.1 THAP1::MA0152.1 NFATC2		309	634	N
MA1100.2 ASCL1::MA0152.1 NFATC2		469	634	N
MA0597.1 THAP1::MA0152.1 NFATC2		474	634	N
MA0868.2 SOX8::MA0152.1 NFATC2		552	634	N
MA1525.1 NFATC4::MA0152.1 NFATC2		557	634	N
MA0868.2 SOX8::MA1100.2 ASCL1		552	645	N
MA1525.1 NFATC4::MA1100.2 ASCL1		557	645	N
MA0152.1 NFATC2::MA1100.2 ASCL1		572	645	N
MA0152.1 NFATC2::MA1100.2 ASCL1		628	645	N
MA0528.2 ZNF263::MA0816.1 Ascl2		197	645	N
MA0597.1 THAP1::MA0816.1 Ascl2		226	645	N
MA1621.1 Rbpjl::MA0816.1 Ascl2		275	645	N
MA1620.1 Ptf1a(var.3)::MA0816.1 Ascl2		276	645	N
MA1100.2 ASCL1::MA0816.1 Ascl2		277	645	N
MA0597.1 THAP1::MA0816.1 Ascl2		309	645	N
MA1100.2 ASCL1::MA0816.1 Ascl2		469	645	N
MA0597.1 THAP1::MA0816.1 Ascl2		474	645	N
MA0868.2 SOX8::MA0816.1 Ascl2		552	645	N

Sequence Name	Module	Module Start	Module End	Sense
	MA0867.2 SOX4::MA0816.1 Ascl2	553	645	N
	MA1525.1 NFATC4::MA0816.1 Ascl2	557	645	N
	MA0152.1 NFATC2::MA0816.1 Ascl2	572	645	N
	MA0152.1 NFATC2::MA0816.1 Ascl2	628	645	N
	MA0528.2 ZNF263::MA1472.1 BHLHA15(var.2)	197	645	N
	MA0597.1 THAP1::MA1472.1 BHLHA15(var.2)	226	645	N
	MA1621.1 Rbpjl::MA1472.1 BHLHA15(var.2)	275	645	N
	MA1620.1 Ptf1a(var.3)::MA1472.1 BHLHA15(var.2)	276	645	N
	MA1100.2 ASCL1::MA1472.1 BHLHA15(var.2)	277	645	N
	MA0597.1 THAP1::MA1472.1 BHLHA15(var.2)	309	645	N
	MA1100.2 ASCL1::MA1472.1 BHLHA15(var.2)	469	645	N
	MA0597.1 THAP1::MA1472.1 BHLHA15(var.2)	474	645	N
	MA0868.2 SOX8::MA1472.1 BHLHA15(var.2)	552	645	N
	MA0867.2 SOX4::MA1472.1 BHLHA15(var.2)	553	645	N
	MA1525.1 NFATC4::MA1472.1 BHLHA15(var.2)	557	645	N
	MA0152.1 NFATC2::MA1472.1 BHLHA15(var.2)	572	645	N
	MA0152.1 NFATC2::MA1472.1 BHLHA15(var.2)	628	645	N
	MA0528.2 ZNF263::MA0816.1 Ascl2	197	478	N
	MA0597.1 THAP1::MA0816.1 Ascl2	226	478	N
	MA1621.1 Rbpjl::MA0816.1 Ascl2	275	478	N
	MA1620.1 Ptf1a(var.3)::MA0816.1 Ascl2	276	478	N
	MA1100.2 ASCL1::MA0816.1 Ascl2	277	478	N
	MA0597.1 THAP1::MA0816.1 Ascl2	309	478	N
	MA0816.1 Ascl2::MA0868.2 SOX8	469	561	N
	MA0816.1 Ascl2::MA0867.2 SOX4	469	562	N
	MA0816.1 Ascl2::MA1525.1 NFATC4	469	566	N
	MA0624.1 NFATC1::MA0152.1 NFATC2	399	407	R
	MA0624.1 NFATC1::MA0624.1 NFATC1	399	566	R
	MA0152.1 NFATC2::MA0624.1 NFATC1	401	566	R

Sequence Name	Module	Module Start	Module End	Sense
NC_000077.6:c53635202-53634702 Mus musculus strain C57BL/6J chromosome 11; GRCm38.p6 C57BL/6J; Mus musculus interleukin 13 (Il13) promoter	MA0152.1 NFATC2::MA0521.1 Tcf12	401	645	R
	MA0624.1 NFATC1::MA0521.1 Tcf12	557	645	R
	MA0624.1 NFATC1::MA0625.1 NFATC3	399	566	R
	MA0152.1 NFATC2::MA0625.1 NFATC3	401	566	R
	MA0625.1 NFATC3::MA0152.1 NFATC2	399	407	R
	MA0528.2 ZNF263::MA1621.1 Rbpjl	82	160	N
	MA0528.2 ZNF263::MA1620.1 Ptf1a(var.3)	82	159	N
	MA1621.1 Rbpjl::MA1620.1 Ptf1a(var.3)	147	159	N
	MA0528.2 ZNF263::MA1100.2 ASCL1	82	158	N
	MA1621.1 Rbpjl::MA1100.2 ASCL1	147	158	N
	MA1620.1 Ptf1a(var.3)::MA1100.2 ASCL1	148	158	N
	MA0528.2 ZNF263::MA0597.1 THAP1	82	162	N
	MA1621.1 Rbpjl::MA0597.1 THAP1	147	162	N
	MA1620.1 Ptf1a(var.3)::MA0597.1 THAP1	148	162	N
	MA1100.2 ASCL1::MA0597.1 THAP1	149	162	N
	MA0528.2 ZNF263::MA0528.2 ZNF263	82	254	N
	MA1621.1 Rbpjl::MA0528.2 ZNF263	147	254	N
	MA1620.1 Ptf1a(var.3)::MA0528.2 ZNF263	148	254	N
	MA1100.2 ASCL1::MA0528.2 ZNF263	149	254	N
	MA0528.2 ZNF263::MA1525.1 NFATC4	82	284	N
	MA1621.1 Rbpjl::MA1525.1 NFATC4	147	284	N
	MA1620.1 Ptf1a(var.3)::MA1525.1 NFATC4	148	284	N
	MA1100.2 ASCL1::MA1525.1 NFATC4	149	284	N
	MA0597.1 THAP1::MA1525.1 NFATC4	154	284	N
	MA0528.2 ZNF263::MA0152.1 NFATC2	82	375	N
	MA1621.1 Rbpjl::MA0152.1 NFATC2	147	375	N

Sequence Name	Module	Module Start	Module End	Sense
MA1620.1 Ptf1a(var.3)::MA0152.1 NFATC2		148	375	N
MA1100.2 ASCL1::MA0152.1 NFATC2		149	375	N
MA0597.1 THAP1::MA0152.1 NFATC2		154	375	N
MA1525.1 NFATC4::MA0152.1 NFATC2		275	375	N
MA0528.2 ZNF263::MA0868.2 SOX8		82	438	N
MA1621.1 Rbpjl::MA0868.2 SOX8		147	438	N
MA1620.1 Ptf1a(var.3)::MA0868.2 SOX8		148	438	N
MA1100.2 ASCL1::MA0868.2 SOX8		149	438	N
MA0597.1 THAP1::MA0868.2 SOX8		154	438	N
MA0528.2 ZNF263::MA0868.2 SOX8		243	438	N
MA1525.1 NFATC4::MA0868.2 SOX8		275	438	N
MA0152.1 NFATC2::MA0868.2 SOX8		369	438	N
MA0528.2 ZNF263::MA0867.2 SOX4		82	439	N
MA1621.1 Rbpjl::MA0867.2 SOX4		147	439	N
MA1620.1 Ptf1a(var.3)::MA0867.2 SOX4		148	439	N
MA1100.2 ASCL1::MA0867.2 SOX4		149	439	N
MA0597.1 THAP1::MA0867.2 SOX4		154	439	N
MA0528.2 ZNF263::MA0867.2 SOX4		243	439	N
MA1525.1 NFATC4::MA0867.2 SOX4		275	439	N
MA0868.2 SOX8::MA0867.2 SOX4		429	439	N
MA0528.2 ZNF263::MA0816.1 Ascl2		82	484	N
MA1621.1 Rbpjl::MA0816.1 Ascl2		147	484	N
MA1620.1 Ptf1a(var.3)::MA0816.1 Ascl2		148	484	N
MA1100.2 ASCL1::MA0816.1 Ascl2		149	484	N
MA0597.1 THAP1::MA0816.1 Ascl2		154	484	N
MA0528.2 ZNF263::MA0816.1 Ascl2		243	484	N
MA1525.1 NFATC4::MA0816.1 Ascl2		275	484	N
MA0152.1 NFATC2::MA0816.1 Ascl2		369	484	N
MA0868.2 SOX8::MA0816.1 Ascl2		429	484	N

Sequence Name	Module	Module Start	Module End	Sense
	MA0867.2 SOX4::MA0816.1 Ascl2	430	484	N
	MA0528.2 ZNF263::MA1472.1 BHLHA15(var.2)	82	484	N
	MA1621.1 Rbpjl::MA1472.1 BHLHA15(var.2)	147	484	N
	MA1620.1 Ptf1a(var.3)::MA1472.1 BHLHA15(var.2)	148	484	N
	MA1100.2 ASCL1::MA1472.1 BHLHA15(var.2)	149	484	N
	MA0597.1 THAP1::MA1472.1 BHLHA15(var.2)	154	484	N
	MA1525.1 NFATC4::MA1472.1 BHLHA15(var.2)	275	484	N
	MA0152.1 NFATC2::MA1472.1 BHLHA15(var.2)	369	484	N
	MA0868.2 SOX8::MA1472.1 BHLHA15(var.2)	429	484	N
	MA0867.2 SOX4::MA1472.1 BHLHA15(var.2)	430	484	N
	MA0521.1 Tcf12::MA0624.1 NFATC1	148	242	R
	MA0521.1 Tcf12::MA0624.1 NFATC1	148	284	R
	MA0624.1 NFATC1::MA0624.1 NFATC1	233	284	R
	MA0521.1 Tcf12::MA0152.1 NFATC2	148	283	R
	MA0624.1 NFATC1::MA0152.1 NFATC2	233	283	R
	MA0624.1 NFATC1::MA0152.1 NFATC2	275	283	R
	MA0521.1 Tcf12::MA0152.1 NFATC2	148	305	R
	MA0624.1 NFATC1::MA0152.1 NFATC2	233	305	R
	MA0624.1 NFATC1::MA0152.1 NFATC2	275	305	R
	MA0521.1 Tcf12::MA0599.1 KLF5	148	456	R
	MA0624.1 NFATC1::MA0599.1 KLF5	233	456	R
	MA0624.1 NFATC1::MA0599.1 KLF5	275	456	R
	MA0152.1 NFATC2::MA0599.1 KLF5	277	456	R
	MA0152.1 NFATC2::MA0599.1 KLF5	299	456	R
	MA0521.1 Tcf12::MA0625.1 NFATC3	148	284	R
	MA0624.1 NFATC1::MA0625.1 NFATC3	233	284	R
	MA0625.1 NFATC3::MA0152.1 NFATC2	275	283	R
	MA0625.1 NFATC3::MA0152.1 NFATC2	275	305	R
	MA0625.1 NFATC3::MA0599.1 KLF5	275	456	R

Untenstehend die Tabelle der gefundenen TFBSs für die Interleukinpromotoren

Sequence Name	Sequence Length	Matrix Title	Matrix Length	Hit No.	Hit Sense	Hit Start	Hit Stop	Hit Score (La)	Hit Max log likelihood ratio score (Lm) or matrix possible	Difference (Ld) (maxscore(Lm) - score(La))	Hit Oligo		
NC_000067.6:131019345-131019844 Mus musculus strain C57BL/6J chromosome 1; GRCm38.p6 C57BL/6J; Mus musculus interleukin 10 (Il10) promoter	500	MA1100.2 ASCL1	10	1	N	208	217	11.290903	13.919985	2.629083	AACAGCTGTC		
		MA0816.1 Ascl2	10	1	N	208	217	14.006852	16.538884	2.532033	AACAGCTGTC		
		MA1472.1 BHLHA15(var.2)	10	1	N	208	217	13.463911	13.956604	0.492693	AACAGCTGTC		
		MA0599.1 KLF5	10	1	R	412	421	9.668823	15.543266	5.874443	AGGGAGGAGG		
		MA0624.1 NFATC1	10	1	R	174	183	8.210574	12.519140	4.308567	AATGGAATCC		
		MA0152.1 NFATC2	7	1	N	142	148	9.609372	12.519140	2.909768	AAAGGAAAAA		
			2	R	129	135	8.063484	12.341468	4.277985	TTATCCA			
			3	R	361	367	10.239107	12.341468	2.102362	TGGAATA			
		MA0625.1 NFATC3	10	1	R	359	368	10.362899	13.675168	3.312269	AGGAAAAA		
		MA1525.1 NFATC4	10	1	N	174	183	8.090887	14.392408	6.301521	AAAGGAAAAA		
		MA1620.1 Ptfla(var.3)	12	1	N	359	368	8.522672	14.392408	5.869736	AATGGAATCC		
			14	1	N	207	218	11.556983	15.503908	3.946925	AAAGGAAAAA		
		MA1621.1 Rbpjl	14	1	N	206	219	11.141220	15.911069	4.769850	GAACAGCTGTCT		
		MA0867.2 SOX4	10	1	N	430	439	9.728276	14.601566	4.873290	AGAACAGCTGTCTG		
		MA0868.2 SOX8	10	1	N	98	107	8.181533	14.854627	6.673094	TAACAAAAAC		
						2	N	338	347	8.322264	14.854627	6.532362	GAAACAATTA
						3	N	429	438	8.402450	14.854627	6.452176	CTTACAATGC
						1	R	207	217	13.002089	15.776101	2.774012	ATAACAAAAA
		MA0521.1 Tcf12	11	1	R	207	217	13.002089	15.776101	2.774012	GAACAGCTGTC		
		MA0597.1 THAP1	9	1	N	442	450	8.693273	11.314089	2.620816	TTGCCAGGA		
		MA0528.2 ZNF263	12	1	N	410	421	15.152542	16.915819	1.763277	AGAGGGAGGAGG		
		NC_000074.6:c82403252-82402576 Mus musculus strain C57BL/6J chromosome 8; GRCm38.p6 C57BL/6J; Mus musculus interleukin 15 (Il15); transcript variant 2 promoter	666	MA1100.2 ASCL1	10	1	N	277	286	10.804723	16.915819	6.111095	GGGAGGAGGAGC
							2	N	469	478	9.914553	13.919985	4.005432
3	N						636	645	9.582197	13.919985	4.337788	GGCTGCTGCC	
1	N						469	478	10.315605	13.919985	3.604380	TCCAGCTGCC	
MA0816.1 Ascl2	10			1	N	469	478	9.355673	16.538884	7.183212	GGCTGCTGCC		
MA1472.1 BHLHA15(var.2)	10			1	N	2	N	636	645	9.632292	16.538884	6.906592	TCCAGCTGCC
						1	N	636	645	10.132307	13.956604	3.824296	TCCAGCTGCC
MA0624.1 NFATC1	10			1	R	399	408	8.079574	12.519140	4.439567	GCTGGAAGC		
						2	R	557	566	9.664478	12.519140	2.854663	AATGGAATTT

Sequence Name	Sequence Length	Matrix Title	Matrix Length	Hit No.	Hit Sense	Hit Start	Hit Stop	Hit Score (La)	Hit Max log likelihood ratio score (Lm) or matrix possible	Difference (Ld) (maxscore(Lm) - score(La))	Hit Oligo		
NC_000077.6:c53635202-53634702 Mus musculus strain C57BL/6J chromosome 11; GRCm38.p6 C57BL/6J; Mus musculus interleukin 13 (Il13) promoter	500	MA0152.1 NFATC2	7	1	N	12	18	8.473572	12.341468	3.867896	TTTTCCG		
				2	N	572	578	8.473572	12.341468	3.867896	TTTTCCG		
				3	N	628	634	9.852084	12.341468	2.489385	TTTTCCG		
		MA0625.1 NFATC3	10	4	R	401	407	8.323546	12.341468	4.017922	TGGAAAG		
				1	R	399	408	8.040663	13.675168	5.634504	GCTGGAAGC		
				2	R	557	566	9.561851	13.675168	4.113317	AATGGAATTT		
		MA1525.1 NFATC4	10	1	N	557	566	8.573176	14.392408	5.819232	AATGGAATTT		
				12	N	276	287	9.691503	15.503908	5.812405	GGGCACCTGGTA		
		Ptfla(var.3)											
		MA1621.1 Rbpjl	14	1	N	275	288	9.349614	15.911069	6.561456	TGGGCACCTGGTAA		
		MA0867.2 SOX4	10	1	N	553	562	11.818057	14.601566	2.783508	CAACAATGGA		
		MA0868.2 SOX8	10	1	N	552	561	11.378412	14.854627	3.476215	TCAACAATGG		
		MA0521.1 Tcf12	11	1	R	635	645	13.813023	15.776101	1.963078	CTCCAGCTGCC		
		MA0597.1 THAP1	9	1	N	226	234	8.221750	11.314089	3.092340	TTGCCCTTC		
				2	N	309	317	8.207882	11.314089	3.106207	CCTCCCGGA		
				3	N	474	482	8.355262	11.314089	2.958828	CTGCCCTGC		
		MA0528.2 ZNF263	12	1	N	197	208	14.731682	16.915819	2.184136	AGGGGGAGGAGA		
		MA1100.2 ASCL1	10	1	N	149	158	8.332904	13.919985	5.587081	CTCACCTGCC		
		MA0816.1 Ascl2	10	1	N	475	484	9.583758	16.538884	6.955126	AAAAGCTGCT		
		MA1472.1 BHLHA15(var.2)	10	1	N	475	484	8.229655	13.956604	5.726949	AAAAGCTGCT		
		MA0599.1 KLF5	10	1	R	447	456	11.424883	15.543266	4.118383	GGGGTGAGGC		
		MA0624.1 NFATC1	10	1	R	233	242	8.219935	12.519140	4.299206	GGTGGAATTA		
				2	R	275	284	10.054376	12.519140	2.464765	CGTGGAATA		
				1	N	369	375	12.341468	12.341468	0.000000	TTTTCCA		
		MA0152.1 NFATC2	7	2	R	277	283	9.702058	12.341468	2.639410	TGAAAT		
				3	R	299	305	9.171543	12.341468	3.169925	TGAAAC		
				1	R	275	284	9.975994	13.675168	3.699174	CGTGGAATA		
		MA1525.1 NFATC4	10	1	N	275	284	9.930748	14.392408	4.461660	CGTGGAATA		
		MA1620.1	12	1	N	148	159	10.008048	15.503908	5.495860	ACTCACCTGCC		
		Ptfla(var.3)											
		MA1621.1 Rbpjl	14	1	N	147	160	10.199248	15.911069	5.711821	TACTCACCTGCCCA		
		MA0867.2 SOX4	10	1	N	430	439	11.560954	14.601566	3.040612	CAACAAAGCA		
		MA0868.2 SOX8	10	1	N	429	438	8.156682	14.854627	6.697945	CCAACAAAGC		
		MA0521.1 Tcf12	11	1	R	148	158	8.710154	15.776101	7.065947	ACTCACCTGCC		
		MA0597.1 THAP1	9	1	N	154	162	9.368914	11.314089	1.945175	CTGCCAAA		
		MA0528.2 ZNF263	12	1	N	82	93	13.683952	16.915819	3.231867	GAAGGGAGGAAG		
				2	N	243	254	9.365203	16.915819	7.550616	CTGGGGCGGAAG		

Gemeinsame TFBSs für beide — AIModules und Genomatix

Die Ergebnisse der TFBS Suche aus AIModules und Genomatix waren sehr zahlreich. Deshalb wurden mittels eines *Python* Scripts (s. Abschnitt 5) Gemeinsamkeiten in den TF Familien zwischen den Ergebnissen beider Produkte gesucht. Das Ergebnis ist in den folgenden Tabellen für Cathepsine und IL-10 zu finden. (* siehe Tabelle 1.2)

Ergebnisse für *Cathepsine*

Tabelle 5.3: Homo_sapiens_cathepsin_V_transcript_variant-1_promoter

Genomatix TFBS	AIModules TFBS
V\$AHRARNT.01	MA0006.1 Ahr::Arnt, MA0259.1 ARNT::HIF1A, MA1464.1 ARNT2, MA0603.1 Arntl
V\$E2F6.01, V\$E2F3.01, V\$E2F2.01, V\$E2F1.01, V\$E2F6.01, V\$E2F4.01, V\$E2F4.01	MA0471.2 E2F6
V\$EGR1.04, V\$EGR1.04, V\$EGR1.04, V\$EGR1.04, V\$EGR1.02	MA0162.4 EGR1
V\$FOXP2.01	MA0481.3 FOXP1
V\$GCM1.01, V\$GCM1.02	MA0767.1 GCM2
V\$KLF2.01, V\$KLF12.01, V\$KLF6.01, V\$EKLF.01, V\$IKLF.01, V\$KLF12.01, V\$KLF6.01, V\$IKLF.01	MA0493.1 Klf1, MA1512.1 KLF11, MA1513.1 KLF15, MA0741.1 KLF16, MA1515.1 KLF2, MA1516.1 KLF3, MA0039.4 KLF4, MA0599.1 KLF5, MA1517.1 KLF6
V\$MAZ.03, V\$MAZ.03, V\$MAZR.01, V\$MAZ.01	MA1522.1 MAZ
V\$NKX31.03, V\$NKX31.01, V\$NKX31.03,	MA0672.1 NKX2-3, MA1645.1 NKX2-2, MA0063.2 NKX2-5, MA0673.1 NKX2-8, MA0124.2 Nkx3-1, MA0122.3 Nkx3-2, MA0674.1 NKX6-1
V\$NFAT5.02	MA0606.1 NFAT5, MA0624.1 NFATC1, MA0152.1 NFATC2, MA0625.1 NFATC3, MA1525.1 NFATC4
V\$NKX31.03, V\$NKX31.01, V\$NKX31.03,	s. NKX
V\$RFX1.01	MA1554.1 RFX7
V\$NRF1.02, V\$NRF1.01	MA0506.1 NRF1
V\$PLAGL1.01, V\$PLAGL1.01, V\$PLAGL1.01,	MA1615.1 Plagl1, MA1548.1 PLAGL2
V\$SPIB.01, V\$SP1.03, V\$SP1.01	MA0746.2 SP3, MA0747.1 SP8, MA1564.1 SP9
V\$TCFAP2B.01, V\$CTCF.01, V\$TCFAP2E.01, V\$CTCF.01, V\$TCFAP2B.01, V\$TCFCP2L1.01,	MA0092.1 Hand1::Tcf3, MA1648.1 TCF12(var.2), MA0522.3 TCF3, MA0632.2 TCFL5
V\$ZBTB3.01, V\$ZBTB7.03, V\$ZBTB7.03,	MA1649.1 ZBTB12, MA1650.1 ZBTB14, MA0695.1 ZBTB7C
V\$ZFP57.01	MA1583.1 ZFP57
V\$ZKSCAN3.01	MA1585.1 ZKSCAN1
V\$ZNF282.01, V\$ZNF263.01, V\$ZNF263.01, V\$ZNF219.01, V\$ZNF263.02, V\$ZNF444.01,	MA1653.1 ZNF148, MA0528.2 ZNF263, MA1630.1 Znf281, MA1655.1 ZNF341, MA1601.1 ZNF75D

Tabelle 5.4: Bos_taurus_cathepsin-Z_promoter

Genomatix TFBS	AIModules TFBS
V\$AHRARNT.02, V\$AHRARNT.01	MA0006.1 Ahr::Arnt, MA0259.1 ARNT::HIF1A, MA1464.1 ARNT2, MA0603.1 Arntl
V\$CEBPE.02	MA0837.1 CEBPE
V\$E2F1.01, V\$E2F1.01, V\$E2F4.01, V\$E2F4_DP1.01, V\$E2F4_DP1.01, V\$E2F1.01, V\$E2F4.01, V\$E2F2.01, V\$E2F2.01, V\$E2F2.01, V\$E2F4.01, V\$E2F1.01	MA0471.2 E2F6
V\$EBF1.01, V\$EBF1.01	MA0154.4 EBF1, MA1604.1 Ebf2, MA1637.1 EBF3
V\$ETV1.02, V\$ETV1.02	MA0761.2 ETV1, MA0764.2 ETV4, MA0765.2 ETV5
V\$GATA1.03, V\$GATA.01, V\$GATA3.02	MA0035.4 GATA1, MA0036.3 GATA2, MA0037.3 GATA3, MA0766.2 GATA5
V\$GCM1.02, V\$GCM1.02	MA0767.1 GCM2
V\$HES1.01	MA1099.2 HES1, MA0616.2 HES2
V\$PBX_HOXA9.01, V\$MEIS1B_HOXA9.01	MA1497.1 HOXA6, MA0594.2 HOXA9
V\$HOXB4.02, V\$HOXB4.02	MA1500.1 HOXB6
V\$GKLF.02, V\$GKLF.03, V\$KLF2.01, V\$KKLF.01, V\$KLF6.01, V\$KLF7.01, V\$IKLF.01, V\$KLF6.01, V\$IKLF.01, V\$IKLF.01, V\$KKLF.01	MA0493.1 Klf1, MA1511.1 KLF10, MA1512.1 KLF11, MA1513.1 KLF15, MA0741.1 KLF16, MA1515.1 KLF2, MA1516.1 KLF3, MA0039.4 KLF4, MA0599.1 KLF5, MA1517.1 KLF6
V\$MAZR.01, V\$MAZR.01, V\$MAZR.01, V\$MAZR.01, V\$MAZR.01, V\$MAZR.01	MA1522.1 MAZ
V\$SOX3.01, V\$SOX1.03	MA0078.1 Sox17, MA1563.1 SOX18, MA0442.2 SOX10, MA1120.1 SOX13, MA1152.1 SOX15, MA0143.4 SOX2, MA0867.2 SOX4, MA0077.1 SOX9
V\$PBX1_MEIS1.02, V\$MEIS1B_HOXA9.01	MA0498.2 MEIS1, MA0774.1 MEIS2, MA0775.1 MEIS3
V\$MYCMAX.03	MA0147.3 MYC
V\$MZF1.01	MA0056.2 MZF1
V\$NFATC1.01, V\$NFATC1.01	MA0152.1 NFATC2
V\$RFX4.01, V\$RFX2.01	MA1554.1 RFX7, MA0509.2 RFX1, MA0798.2 RFX3
V\$OSR2.01	MA1646.1 OSR2
V\$PLAGL1.01, V\$PLAGL1.01	MA1615.1 Plagl1, MA1548.1 PLAGL2
V\$RFX4.01, V\$RFX2.01	s. rfx
Weiter auf der nächsten Seite	

Tabelle 5.4 – Fortsetzung von vorheriger Seite

Genomatix TFBS	AIModules TFBS
V\$TCF11MAFG.01, V\$CTCF.01, V\$CTCF.01, V\$CTCF.01, V\$CTCF.01, V\$TCFAP2A.01, V\$TCFAP2A.01, V\$TCFAP2A.02, V\$TCFAP2B.01	MA0092.1 Hand1::Tcf3, MA1648.1 TCF12(var.2), MA0522.3 TCF3, MA0830.2 TCF4, MA0769.2 TCF7, MA0632.2 TCFL5
V\$VDR_RXR.05, V\$VDR_RXR.04, V\$VDR_RXR.01	MA0693.2 VDR
V\$ZBTB7.03, V\$ZBTB7.03, V\$ZBTB7.03, V\$ZBTB7.03	MA1649.1 ZBTB12, MA1650.1 ZBTB14
V\$ZNF345.01, V\$ZNF217.01, V\$ZNF263.01, V\$ZNF263.02, V\$ZNF76_143.01, V\$ZNF76_143.01, V\$ZNF219.01, V\$ZNF444.01, V\$ZNF704.01, V\$ZNF704.01, V\$ZNF219.01, V\$ZNF704.01, V\$ZNF704.01, V\$ZNF263.02	MA1653.1 ZNF148, MA0528.2 ZNF263

Tabelle 5.5: Mus_musculus_cathepsin-F-transcript-variant-X1_-promoter_576-107

Genomatix TFBS	AIModules TFBS
V\$BATF.01	MA1634.1 BATF, MA0462.2 BATF::JUN, MA0835.2 BATF3
V\$JUNB.01	MA0462.2 BATF::JUN, MA0099.3 FOS::JUN, MA1135.1 FOSB::JUNB, MA1128.1 FOSL1::JUN, MA1137.1 FOSL1::JUNB, MA1142.1 FOSL1::JUN, MA1143.1 FOSL1::JUN, MA1130.1 FOSL2::JUN, MA1132.1 JUN::JUNB, MA0490.2 JUNB
V\$BATF.01	s. BATF
V\$CEBPA.01, V\$CEBPG.01, V\$CEBPA.01	MA0466.2 CEBPB, MA0836.2 CEBPD, MA0837.1 CEBPE
V\$RORA.01	MA0071.1 RORA
V\$DLX1.02, V\$DLX1.02	MA0879.1 Dlx1, MA0885.1 Dlx2, MA0880.1 Dlx3, MA0881.1 Dlx4, MA1476.1 DLX5, MA0882.1 DLX6
V\$EN1.02	MA0027.2 EN1
V\$ESRRA.02, V\$ESRRA.01, V\$ESRRA.03	MA0592.3 ESRRA
Weiter auf der nächsten Seite	

Tabelle 5.5 – Fortsetzung von vorheriger Seite

Genomatix TFBS	AIModules TFBS
V\$FOXP1_ES.01	MA0047.3 FOXA2, MA1683.1 FOXA3, MA0042.2 FOXI1, MA0614.1 Foxj2, MA1103.2 FOXK2, MA0033.2 FOXL1, MA1489.1 FOXN3, MA0157.2 FOXO3, MA0848.1 FOXO4, MA0849.1 FOXO6, MA0481.3 FOXP1, MA0593.1, MA0850.1 FOXP3 FOXP2
V\$HIC1.02	MA0739.1 Hic1, MA0738.1 HIC2
V\$MEIS1B_HOXA9.01, V\$MEIS1B_HOXA9.01, V\$HOXC6.01, V\$HOXC9.01, V\$HOXC8.01	MA0719.1 RHOXF1, MA1495.1 HOXA1, MA0899.1 HOXA10, MA0900.2 HOXA2, MA1496.1 HOXA4, MA0158.2 HOXA5, MA1497.1 HOXA6, MA1498.1 HOXA7, MA0594.2 HOXA9, MA0902.2 HOXB2, MA0903.1 HOXB3, MA1499.1 HOXB4, MA0904.2 HOXB5, MA1500.1 HOXB6, MA1501.1 HOXB7, MA1502.1 HOXB8, MA0905.1 HOXC10, MA1504.1 HOXC4, MA1505.1 HOXC8, MA0912.2 HOXD3, MA1507.1 HOXD4, MA0910.2 HOXD8, MA0913.2 HOXD9, MA0630.1 SHOX, MA0720.1 Shox2
V\$ISL2.01	MA0914.1 ISL2
V\$ISX.01	MA0654.1 ISX
V\$JUNB.01	MA1135.1 FOSB::JUNB, MA1137.1 FOSL1::JUNB, MA1132.1 JUN::JUNB
V\$KLF12.01, V\$KLF12.01, V\$EKLF.01, V\$KLF2.01, V\$KKLF.01, V\$GKLF.02	MA1512.1 KLF11, MA1515.1 KLF2, MA0039.4 KLF4, MA1517.1 KLF6
V\$KLF12.01, V\$KLF12.01, V\$EKLF.01, V\$KLF2.01, V\$KKLF.01, V\$GKLF.02	s. KLF
V\$LMX1B.01, V\$LMX1A.02, V\$LMX1B.01	MA0702.2 LMX1A, MA0703.2 LMX1B
V\$SOX5.01, V\$SOX7.02, V\$SOX5.01, V\$SOX7.03, V\$SOX9.09, V\$SOX17.02	MA0078.1 Sox17, MA1563.1 SOX18, MA0442.2 SOX10, MA1120.1 SOX13, MA1562.1 SOX14, MA1152.1 SOX15, MA0143.4 SOX2, MA0867.2 SOX4, MA0087.1 Sox5, MA0868.2 SOX8, MA0077.1 SOX9
V\$MEIS1B_HOXA9.01, V\$MEIS1.02, V\$PBX1_MEIS1.02, V\$MEIS1B_HOXA9.01	MA0498.2 MEIS1, MA0775.1 MEIS3
V\$MSX1.01	MA0666.1 MSX1, MA0708.1 MSX2, MA0709.1 Msx3
V\$NKX25.05, V\$NKX29.01, V\$NKX25.02	MA0672.1 NKX2-3, MA0063.2 NKX2-5, MA0673.1 NKX2-8
Weiter auf der nächsten Seite	

Tabelle 5.5 – Fortsetzung von vorheriger Seite

Genomatix TFBS	AIModules TFBS
V\$NKX25.05, V\$NKX29.01, V\$NKX12.01, V\$NKX61.01, V\$NKX12.01, V\$NKX61.03, V\$NKX25.02	MA0672.1 NKX2-3, MA0063.2 NKX2-5, MA0673.1 NKX2-8, MA0674.1 NKX6-1, MA0675.1 NKX6-2, MA1530.1 NKX6-3
V\$RFX5.01	MA1554.1 RFX7
V\$OSR2.01	MA1646.1 OSR2
V\$PAX6.03, V\$PAX6.03, V\$PAX4.02, V\$PAX4.02	MA0068.2 PAX4
V\$PRDM14.01, V\$PRDM4.01, V\$PRDM15.01	MA0508.3 PRDM1, MA1647.1 PRDM4
V\$PARAXIS.01	MA0718.1 RAX, MA0717.1 RAX2
V\$SNAI2.01	MA1558.1 SNAI1, MA0745.2 SNAI2, MA1559.1 SNAI3
V\$SOX5.01, V\$SOX7.02, V\$SOX5.01, V\$SOX7.03, V\$SOX9.09, V\$SOX17.02	s. sox
V\$TEAD4.01	MA0808.1 TEAD3, MA0809.2 TEAD4
V\$ZNF771.01, V\$ZNF217.01, V\$ZNF263.01	MA1593.1 ZNF317, MA1655.1 ZNF341, MA1657.1 ZNF652

Ergebnisse für *IL-10*

Tabelle 5.6: X73536.1_H.sapiens_promoter_region_of_human_IL-10_gene

Genomatix TFBS	AIModules TFBS
V\$BARX2.01, V\$BARX2.02	MA0875.1 BARX1, MA1471.1 BARX2
V\$CEBPE.02, V\$CEBPE.01, V\$CEBPB.01, V\$CEBPB.01, V\$CEBPE_ATF4.02, V\$CEBPE.02, V\$CEBPE.02	MA0102.4 CEBPA, MA0466.2 CEBPB, MA0836.2 CEBPD, MA0837.1 CEBPE, MA0019.1 Ddit3::Cebpa
V\$CREB.02	MA0638.1 CREB3, MA0608.1 Creb3l2, MA1474.1 CREB3L4
V\$RORA2.01, V\$RORA.01	MA0071.1 RORA
V\$DLX2.01, V\$DLX3.02, V\$DLX1.01	MA0879.1 Dlx1, MA0885.1 Dlx2, MA0880.1 Dlx3, MA0881.1 Dlx4, MA1476.1 DLX5, MA0882.1 DLX6
V\$EBF1.01	MA1604.1 Ebf2, MA1637.1 EBF3
V\$EN1.01	MA0027.2 EN1
V\$ESRRA.02	MA0592.3 ESRRA
Weiter auf der nächsten Seite	

Tabelle 5.6 – Fortsetzung von vorheriger Seite

Genomatix TFBS	AIModules TFBS
V\$FOXA1.01	MA0148.4 FOXA1, MA0047.3 FOXA2, MA1683.1 FOXA3, MA1606.1 Foxf1, MA0479.1 FOXH1, MA0042.2 FOXI1, MA1103.2 FOXK2, MA1489.1 FOXN3, MA0157.2 FOXO3, MA0848.1 FOXO4, MA0481.3 FOXP1, MA0850.1 FOXP3
V\$GBX1.01	MA0889.1 GBX1, MA0890.1 GBX2
V\$HOXB3.01, V\$HOXB3.01, V\$PHOX2.01, V\$HOX_PBX.01, V\$HOX1-3.01, V\$HOXB8.01, V\$HOXC4.01, V\$HOXC13.02, V\$HOX1-3.01, V\$PHOX2.01, V\$HOXA1.01, V\$HOXC13.01, V\$HOXB3.01, V\$HOXC4.01, V\$HOXC9.01, V\$HOXB4.01, V\$RHOX6.01	MA0719.1 RHOXF1, MA1495.1 HOXA1, MA0899.1 HOXA10, MA0900.2 HOXA2, MA1496.1 HOXA4, MA0158.2 HOXA5, MA1497.1 HOXA6, MA1498.1 HOXA7, MA0594.2 HOXA9, MA0902.2 HOXB2, MA0903.1 HOXB3, MA1499.1 HOXB4, MA0904.2 HOXB5, MA1500.1 HOXB6, MA1501.1 HOXB7, MA1502.1 HOXB8, MA0905.1 HOXC10, MA1504.1 HOXC4, MA1505.1 HOXC8, MA0912.2 HOXD3, MA1507.1 HOXD4, MA0910.2 HOXD8, MA0913.2 HOXD9, MA0629.1 Rhox11, MA0630.1 SHOX, MA0720.1 Shox2
V\$INSM1.01, V\$INSM1.01	MA0155.1 INSM1
V\$ISL2.01, V\$ISL2.01	MA1608.1 Isl1, MA0914.1 ISL2
V\$ISX.01	MA0654.1 ISX
V\$EKLF.01, V\$KLF2.01, V\$GKLF.01, V\$KLF2.01, V\$KLF6.01, V\$GKLF.01, V\$GKLF.02	MA1515.1 KLF2, MA0039.4 KLF4, MA0599.1 KLF5, MA1517.1 KLF6
V\$LHX2.01, V\$LHX3.02, V\$LHX3.02, V\$LHX6.01	MA1518.1 LHX1, MA0700.2 LHX2, MA0704.1 Lhx4, MA1519.1 LHX5, MA0658.1 LHX6, MA0705.1 Lhx8, MA0701.2 LHX9
V\$LMX1A.02, V\$LMX1A.02	MA0702.2 LMX1A
V\$MAX.03	MA0058.3 MAX
V\$SOX1.03, V\$SOX9.02, V\$SOX1.03, V\$SOX13.01, V\$SOX5.01	MA0078.1 Sox17, MA1563.1 SOX18, MA0442.2 SOX10, MA1120.1 SOX13, MA1152.1 SOX15, MA0143.4 SOX2, MA0867.2 SOX4, MA0087.1 Sox5, MA0868.2 SOX8, MA0077.1 SOX9
V\$MEF2A.01	MA0052.4 MEF2A, MA0660.1 MEF2B, MA0497.1 MEF2C, MA0773.1 MEF2D
V\$MNT.01	MA0825.1 MNT
V\$MSX.01, V\$MSX1.01	MA0666.1 MSX1, MA0708.1 MSX2, MA0709.1 Msx3

Weiter auf der nächsten Seite

Tabelle 5.6 – Fortsetzung von vorheriger Seite

Genomatix TFBS	AIModules TFBS
V\$MYBL1.02	MA0100.3 MYB
V\$NKX25.02, V\$NKX12.01, V\$NKX61.02, V\$NKX25.02, V\$NKX31.01, V\$NKX63.01	MA0672.1 NKX2-3, MA0063.2 NKX2-5, MA0503.1 Nkx2-5(var.2), MA0673.1 NKX2- 8, MA0124.2 Nkx3-1, MA0674.1 NKX6-1, MA0675.1 NKX6-2, MA1530.1 NKX6-3
V\$MYOD.02, V\$MYOD.02	MA0499.2 MYOD1
V\$MYOGENIN.03, V\$MYOGENIN.03	MA0500.2 MYOG
V\$NFAT.01, V\$NFAT5.02, V\$NFAT5.01	MA0606.1 NFAT5, MA0624.1 NFATC1, MA0152.1 NFATC2, MA0625.1 NFATC3, MA1525.1 NFATC4
V\$NKX25.02, V\$NKX12.01, V\$NKX61.02, V\$NKX25.02, V\$NKX31.01, V\$NKX63.01	s. NKX
V\$NR2F6.01	MA0017.2 NR2F1, MA1111.1 NR2F2
V\$RFX1.01, V\$RFX3.03, V\$RFX1.01	MA1554.1 RFX7
V\$PAX5.01, V\$PAX2.01, V\$PAX2.01, V\$PAX5.03, V\$PAX8.01, V\$PAX7.01, V\$PAX6.03	MA0067.1 Pax2, MA0068.2 PAX4
V\$POU3F3.01	MA1115.1 POU5F1, MA0792.1 POU5F1B, MA0628.1 POU6F1, MA1549.1 POU6F1(var.2), MA0793.1 POU6F2
V\$PRDM14.01, V\$PRDM14.01, V\$PRDM14.01	MA0508.3 PRDM1, MA1647.1 PRDM4
V\$PRDM14.01, V\$PRDM14.01, V\$PRDM14.01	s. prdm
V\$RBPJK.02	MA1116.1 RBPJ, MA1621.1 Rbpjl
V\$SOX1.03, V\$SOX9.02, V\$SOX1.03, V\$SOX13.01, V\$SOX5.01	s. SOX
V\$STAT.01, V\$STAT6.01, V\$STAT3.02, V\$STAT6.01, V\$STAT6.01, V\$STAT3.02, V\$STAT.01	MA0137.3 STAT1, MA0144.2 STAT3, MA0518.1 Stat4, MA0519.1 Stat5a::Stat5b
V\$TCF21.01, V\$TCF21.01, V\$TCF21.01	MA0092.1 Hand1::Tcf3, MA0091.1 TAL1::TCF3, MA0521.1 Tcf12, MA1648.1 TCF12(var.2), MA0832.1 Tcf21, MA0522.3 TCF3, MA0769.2 TCF7
V\$TCF21.01, V\$TCF21.01, V\$TCF21.01	s. TCF
V\$TEAD4.01	MA0090.3 TEAD1, MA1121.1 TEAD2, MA0808.1 TEAD3, MA0809.2 TEAD4
V\$VAX2.01	MA0722.1 VAX1, MA0723.1 VAX2
V\$VDR_RXR.06, V\$VDR_RXR.05, V\$VDR_RXR.06	MA0693.2 VDR
V\$ZBTB7.01	MA1649.1 ZBTB12
Weiter auf der nächsten Seite	

Tabelle 5.6 – Fortsetzung von vorheriger Seite

Genomatix TFBS	AIModules TFBS
V\$ZNF444.01, V\$ZNF282.01, V\$ZNF300.01, V\$ZNF282.01	MA0528.2 ZNF263, MA1630.1 Znf281, MA1593.1 ZNF317, MA1655.1 ZNF341, MA1125.1 ZNF384
O\$ZSCAN4.01, V\$ZSCAN10.01	MA1602.1 ZSCAN29

Tabelle 5.7: AY486432.1_Macaca-mulatta_interleukin-10-(IL-10)_-
gene_promoter_region

Genomatix TFBS	AIModules TFBS
V\$CEBPE_ATF4.01, V\$ATF6.02, V\$CEBPE_ATF4.02 V\$BARX2.01, V\$BARX2.02	MA0604.1 Atf1, MA1632.1 ATF2, MA0833.2 ATF4 MA0875.1 BARX1, MA1471.1 BARX2
V\$CEBPB.01, V\$CEBPE_ATF4.01, V\$CEBPE.02, V\$CEBPE.01, V\$CEBPB.01, V\$CEBPB.01, V\$CEBPE_ATF4.02, V\$CEBPE.02, V\$CEBPE.02	MA0102.4 CEBPA, MA0466.2 CEBPB, MA0836.2 CEBPD, MA0837.1 CEB- PE, MA1636.1 CEBPG(var.2), MA0019.1 Ddit3::Cebpa
V\$CREB1.02, V\$CREB.02	MA0018.4 CREB1, MA0638.1 CREB3, MA0608.1 Creb3l2, MA1474.1 CREB3L4, MA1475.1 CREB3L4(var.2), MA0840.1 Creb5
V\$RORA2.01, V\$RORA.01	MA0071.1 RORA
V\$DLX2.01, V\$DLX3.02, V\$DLX1.01	MA0879.1 Dlx1, MA0885.1 Dlx2, MA0880.1 Dlx3, MA0881.1 Dlx4, MA1476.1 DLX5, MA0882.1 DLX6
V\$E2F1.01, V\$E2F7.02	MA0471.2 E2F6
V\$EBF1.01	MA1604.1 Ebf2, MA1637.1 EBF3
V\$EN1.01, V\$EN1.01	MA0027.2 EN1
V\$ERG.02	MA0474.2 ERG
V\$ESRRA.02, V\$ESRRA.04	MA0592.3 ESRRA
V\$ETS1.01	MA0098.3 ETS1, MA1484.1 ETS2
V\$ETV1.02	MA0761.2 ETV1, MA0763.1 ETV3, MA0764.2 ETV4, MA0765.2 ETV5, MA0645.1 ETV6
Weiter auf der nächsten Seite	

Tabelle 5.7 – Fortsetzung von vorheriger Seite

Genomatix TFBS	AIModules TFBS
V\$JUNDM2.01	MA1126.1 FOS::JUN(var.2), MA1127.1 FOSB::JUN, MA1136.1 FOSB::JUNB(var.2), MA1129.1 FOSL1::JUN(var.2), MA1142.1 FOSL1::JUN, MA1143.1 FOSL1::JUN(var.2), MA1131.1 FOSL2::JUN(var.2), MA1139.1 FOSL2::JUNB(var.2), MA1145.1 FOSL2::JUN(var.2), MA0488.1 JUN, MA1132.1 JUN::JUNB, MA1133.1 JUN::JUNB(var.2), MA0492.1 JUND(var.2)
V\$GATA1.01	MA0035.4 GATA1, MA0037.3 GATA3, MA0482.2 GATA4
V\$GBX1.01	MA0889.1 GBX1, MA0890.1 GBX2
V\$GFI1.01, V\$GFI1.02	MA0038.2 GFI1, MA0483.1 Gfi1b
V\$GRHL2.01	MA1105.2 GRHL2
V\$HIC1.01, V\$HIC1.01	MA0739.1 Hic1, MA0738.1 HIC2
V\$HMBOX.01	MA0895.1 HMBOX1
V\$HOXB6.01, V\$HOXD13.01, V\$HOX1-3.01, V\$HOXB8.01, V\$HOXC4.01, V\$HOXC13.02, V\$PHOX2.01, V\$HOXA1.01, V\$HOXC13.01, V\$HOXB3.01, V\$MEIS1A_HOXA9.01, V\$HOXB9.02	MA0719.1 RHOXF1, MA1495.1 HOXA1, MA0899.1 HOXA10, MA0900.2 HOXA2, MA1496.1 HOXA4, MA0158.2 HOXA5, MA1497.1 HOXA6, MA1498.1 HOXA7, MA0594.2 HOXA9, MA0901.2 HOXB13, MA0902.2 HOXB2, MA0903.1 HOXB3, MA1499.1 HOXB4, MA0904.2 HOXB5, MA1500.1 HOXB6, MA1501.1 HOXB7, MA1502.1 HOXB8, MA0905.1 HOXC10, MA1504.1 HOXC4, MA1505.1 HOXC8, MA1507.1 HOXD4, MA0910.2 HOXD8, MA0913.2 HOXD9, MA0630.1 SHOX, MA0720.1 Shox2
V\$INSM1.01, V\$INSM1.01, V\$INSM1.01	MA0155.1 INSM1
V\$ISL2.01	MA0914.1 ISL2
V\$ISX.01	MA0654.1 ISX
V\$GKLF.01, V\$KKLF.01, V\$EKLF.01, V\$KLF2.01, V\$GKLF.01, V\$EKLF.01, V\$KLF2.01, V\$KLF6.01, V\$GKLF.01, V\$GKLF.02	MA1515.1 KLF2, MA0039.4 KLF4, MA0599.1 KLF5, MA1517.1 KLF6
V\$LHX2.01, V\$LHX3.02, V\$LHX3.02, V\$LHX6.01	MA1518.1 LHX1, MA0700.2 LHX2, MA0704.1 Lhx4, MA1519.1 LHX5, MA0658.1 LHX6, MA0705.1 Lhx8, MA0701.2 LHX9

Weiter auf der nächsten Seite

Tabelle 5.7 – Fortsetzung von vorheriger Seite

Genomatix TFBS	AIModules TFBS
V\$LHX2.01, V\$LHX3.02, V\$LHX3.02, V\$LHX6.01	s. LHX
V\$LMX1A.02	MA0702.2 LMX1A
V\$MEF2.01, V\$MEF2A.01	MA0052.4 MEF2A, MA0660.1 MEF2B, MA0497.1 MEF2C, MA0773.1 MEF2D
V\$MEIS1A_HOXA9.01	MA0498.2 MEIS1, MA0774.1 MEIS2, MA0775.1 MEIS3
V\$MSX.01, V\$MSX1.01	MA0666.1 MSX1, MA0708.1 MSX2, MA0709.1 Msx3
V\$VMYB.01, V\$MYBL1.02, V\$MYBL1.02, V\$CMYB.01	MA0100.3 MYB
V\$NKX25.02, V\$NKX12.01, V\$NKX61.02, V\$NKX25.02, V\$NKX31.01	MA0672.1 NKX2-3, MA0503.1 Nkx2-5(var.2), MA0673.1 NKX2-8, MA0124.2 Nkx3-1, MA0674.1 NKX6-1, MA0675.1 NKX6-2, MA1530.1 NKX6-3
V\$MYOD.02	MA0499.2 MYOD1
V\$MYOGENIN.03	MA0500.2 MYOG
V\$MZF1.02, V\$MZF1.02, V\$MZF1.02, V\$MZF1.01	MA0056.2 MZF1
V\$NFAT.01, V\$NFAT5.01, V\$NFATC1.01, V\$NFATC1.01, V\$NFAT.01	MA0606.1 NFAT5, MA0624.1 NFATC1, MA0152.1 NFATC2, MA0625.1 NFATC3, MA1525.1 NFATC4
V\$NKX25.02, V\$NKX12.01, V\$NKX61.02, V\$NKX25.02, V\$NKX31.01	s. NKX
V\$NR2F6.01	MA0017.2 NR2F1, MA1111.1 NR2F2
V\$RFX2.01, V\$RFX1.01	MA1554.1 RFX7
V\$OTX2.01	MA0711.1 OTX1, MA0712.2 OTX2
V\$PAX2.01, V\$PAX3.03, V\$PAX5.01, V\$PAX5.03, V\$PAX2.01, V\$PAX2.01, V\$PAX9.01, V\$PAX5.03, V\$PAX6.03	MA0067.1 Pax2, MA0068.2 PAX4
V\$PRDM15.01, V\$PRDM14.01, V\$PRDM14.01, V\$PRDM14.01	MA0508.3 PRDM1
V\$PROX1.01, V\$PROX1.01	MA0794.1 PROX1
V\$SIX4.01, V\$SIX2.02	MA1118.1 SIX1
V\$SNAI3.01	MA1558.1 SNAI1, MA0745.2 SNAI2, MA1559.1 SNAI3
Weiter auf der nächsten Seite	

Tabelle 5.7 – Fortsetzung von vorheriger Seite

Genomatix TFBS	AIModules TFBS
V\$SOX1.03, V\$SOX9.02, V\$SOX1.03, V\$SOX13.01, V\$SOX5.01	MA0078.1 Sox17, MA1563.1 SOX18, MA0442.2 SOX10, MA1120.1 SOX13, MA1152.1 SOX15, MA0143.4 SOX2, MA0867.2 SOX4, MA0087.1 Sox5, MA0868.2 SOX8, MA0077.1 SOX9
V\$SRY.05	MA0084.1 SRY
V\$STAT1.01, V\$STAT.01, V\$STAT6.01, V\$STAT6.01, V\$STAT6.01, V\$STAT3.02, V\$STAT.01	MA0137.3 STAT1, MA0144.2 STAT3, MA0518.1 Stat4, MA0519.1 Stat5a::Stat5b
V\$TCF21.01, V\$TCF21.01, V\$TCF21.01	MA0092.1 Hand1::Tcf3, MA0091.1 TAL1::TCF3, MA0521.1 Tcf12, MA1648.1 TCF12(var.2), MA0832.1 Tcf21, MA0522.3 TCF3, MA0830.2 TCF4, MA0769.2 TCF7
V\$TCF21.01, V\$TCF21.01, V\$TCF21.01	s. TCF
V\$TEAD4.01	MA0090.3 TEAD1, MA1121.1 TEAD2, MA0808.1 TEAD3, MA0809.2 TEAD4
V\$TEF.01, V\$TEF_HLF.01	MA0843.1 TEF
V\$VDR_RXR.01, V\$VDR_RXR.04, V\$VDR_RXR.03, V\$VDR_RXR.05, V\$VDR_RXR.06	MA0693.2 VDR
V\$YY2.01	MA0748.2 YY2
V\$ZBTB3.01	MA1649.1 ZBTB12, MA1579.1 ZBTB26, MA1581.1 ZBTB6
V\$ZKSCAN3.01	MA1585.1 ZKSCAN1
V\$ZNF35.01, V\$ZNF263.02, V\$ZNF771.01, V\$ZNF444.01, V\$ZNF219.01, V\$ZNF300.01, V\$ZNF282.01	MA0528.2 ZNF263, MA1630.1 Znf281, MA1593.1 ZNF317, MA1655.1 ZNF341, MA1125.1 ZNF384
O\$ZSCAN4.01, V\$ZSCAN10.01	MA1602.1 ZSCAN29

Tabelle 5.8: AF121965.1_Mus-musculus_interleukin-10-(IL10)_gene_promoter_partial_sequence

Genomatix TFBS	AIModules TFBS
V\$E2F7.02	MA0471.2 E2F6
V\$ETV1.02	MA0761.2 ETV1, MA0764.2 ETV4, MA0765.2 ETV5
V\$FOXO1.01, V\$FOXP1_ES.01	MA0148.4 FOXA1, MA0047.3 FOXA2, MA1683.1 FOXA3, MA0845.1 FOXB1, MA0032.2 FOXC1, MA0846.1 FOXC2, MA0847.2 FOXD2, MA0041.1 Foxd3, MA0042.2 FOXI1, MA0614.1 Foxj2, MA1103.2 FOXK2, MA0033.2 FOXL1, MA0480.1 Foxo1, MA0848.1 FOXO4, MA0849.1 FOXO6, MA0481.3 FOXP1, MA0593.1 FOXP2, MA0850.1 FOXP3
V\$HIC1.01	MA0739.1 Hic1, MA0738.1 HIC2
V\$HOXD10.01	MA0719.1 RHOXF1, MA1495.1 HOXA1, MA0594.2 HOXA9, MA1501.1 HOXB7
V\$KLF6.01, V\$IKLF.01, V\$KLF2.01, V\$GKLF.02	MA0741.1 KLF16, MA1515.1 KLF2, MA0039.4 KLF4, MA0599.1 KLF5
V\$LHX6.02	MA1518.1 LHX1, MA0704.1 Lhx4, MA0705.1 Lhx8
V\$SOX9.03	MA1563.1 SOX18, MA0442.2 SOX10, MA1152.1 SOX15, MA0077.1 SOX9
V\$RFX5.01	MA1554.1 RFX7
V\$PAX6.05, V\$PAX3.02, V\$PAX6.04	MA0014.3 PAX5
V\$PLAGL1.02	MA1548.1 PLAGL2
V\$PRDM4.01	MA1647.1 PRDM4
V\$RBPJK.02	MA1116.1 RBPJ
V\$SMAD3.01	MA1153.1 Smad4, MA1557.1 SMAD5
V\$SOX9.03	s. SOX
V\$SRY.05	MA0084.1 SRY
V\$CTCF.01	MA0092.1 Hand1::Tcf3, MA0769.2 TCF7
V\$TEAD4.01	MA1121.1 TEAD2, MA0808.1 TEAD3, MA0809.2 TEAD4
V\$ZNF652.02, V\$ZNF263.02, V\$ZNF217.01, V\$ZNF444.01	MA0753.2 ZNF740, MA0528.2 ZNF263, MA1655.1 ZNF341, MA0116.1 Znf423, MA1601.1 ZNF75D

Python Skript für die Auswertung der TFBSs Ergebnisse aus Genomatix vs AIModules

Dieses Skript wollen wir auch veröffentlichen (eingereicht bei BMC Bioinformatics)

```

arrayCathepsinsHomoTFBSAIModules = ["Arnt","ALX3",...]
arrayCathepsinsBosTFBSAIModules = ["Arnt","HIF1A",...]
arrayCathepsinsMusTFBSAIModules = ["ALX","ARGFX",...]
arrayCathepsinsHomoTFBSGenomatix = ["V$WT1.01","V$MTF-1.01",...]
arrayCathepsinsBosTFBSGenomatix = ["V$DEC2.01","V$SIX.01",...]
arrayCathepsinsMusTFBSGenomatix = ["V$FTF.01","V$PEG3.01",...]
arrayIL10HomoTFBSAIModules = ["Arnt","ALX",...]
arrayIL10MacacaTFBSAIModules = ["Arnt","ALX3ARGFX",...]
arrayIL10MusTFBSAIModules = ["Arn","ALX3",...]
arrayIL10HomoTFBSGenomatix = ["O$ZSCAN4.01","V$LACTOFERRIN.01",...]
arrayIL10MacacaTFBSGenomatix = ["V$MZF1.02","V$SPI1.02",...]
arrayIL10MusTFBSGenomatix = ["V$THRB.01","V$ZNF652.02",...]

'''
// AIModules and Genomatix: one Matrix many Hits!

// When modules search is selected, then the tfbs button only shows the tfbs
from the modules, i.e. a smaller number of hits compared to just tfbs searches.
//IL10 Matrices:
//>MA0519.1 Stat5a::Stat5b
//>MA0520.1 Stat6
//>MA0144.2 STAT3
//>MA0517.1 STAT1::STAT2
//>MA0518.1 Stat4
//>MA0050.2 IRF1
//>MA0772.1 IRF7
//>MA1420.1 IRF5

Matrix Title Matrix Length Hit No. Hit Sense Hit Start Hit Stop Hit Score (La)
Hit Max log likelihood ratio score (Lm) or matrix possible
Difference (Ld) (maxscore(Lm) - score(La)) Hit Oligo

Homo
MA0519.1 Stat5a::Stat5b 11 1 N 480 490 9.140897 16.951823 7.810926 TATTCTAAGAG
MA0518.1 Stat4 14 1 N 626 639 12.487977 18.256690 5.768713 GTGCCGGGAAACCT
MA0137.3 STAT1 11 1 N 626 636 11.466994 17.702003 6.235008 GTGCCGGGAAA
MA0144.2 STAT3 11 1 N 626 636 13.489612 16.778350 3.288738 GTGCCGGGAAA

```

Macaca

```
MA0519.1 Stat5a::Stat5b 11 1 N 839 849 9.140897 16.951823 7.810926 TATTCTAAGAG
MA0518.1 Stat4 14 1 N 982 995 12.487977 18.256690 5.768713 GTGCCGGGAAACCT
MA0137.3 STAT1 11 1 N 982 992 11.466994 17.702003 6.235008 GTGCCGGGAAA
MA0144.2 STAT3 11 1 N 982 992 13.489612 16.778350 3.288738 GTGCCGGGAAA
```

'''

```
def findCommonTFBS(nameOfGeneAndSpecies, AIModulesTFBSs, genomatixTFBSs):
    result = []
    for i in range(len(AIModulesTFBSs)):
        subresult = ""
        for j in range (len(genomatrixTFBSs)):
            if AIModulesTFBSs[i] in genomatrixTFBSs[j]:
                subresult += genomatrixTFBSs[j] + ", "
        if len(subresult) > 0:
            result.append(AIModulesTFBSs[i] + " : " + subresult)
    print nameOfGeneAndSpecies
    print result
```

```
findCommonTFBS("Cathepsins Homo sapiens:", arrayCathepsinsHomoTFBSAIModules,
arrayCathepsinsHomoTFBSGenomatix)
findCommonTFBS("Cathepsins Bos taurus:", arrayCathepsinsBosTFBSAIModules,
arrayCathepsinsBosTFBSGenomatix)
findCommonTFBS("Cathepsins Mus musculus:", arrayCathepsinsMusTFBSAIModules,
arrayCathepsinsMusTFBSGenomatix)
```

```
findCommonTFBS("IL10 Homo sapiens:", arrayIL10HomoTFBSAIModules,
arrayIL10HomoTFBSGenomatix)
findCommonTFBS("IL10 Macaca mulatta:", arrayIL10MacacaTFBSAIModules,
arrayIL10MacacaTFBSGenomatix)
findCommonTFBS("IL10 Mus musculus:", arrayIL10MusTFBSAIModules,
arrayIL10MusTFBSGenomatix)
```

Verwendete Software

Das hier vorgestellte Tool *AIModules* wird unter der GPLv2 veröffentlicht.
Folgende Libraries/Tools wurden außerdem verwendet:

Tabelle 5.9: Verwendete Softwarebibliotheken/Tools und deren Lizenzen

Bibliothek/Tool	Lizenz
angularjs	MIT-Lizenz
apache tomcat	Apache License 2.0
bower	MIT License
docker	Apache License 2.0
eclipse	Eclipse Public License
eclipselink	Eclipse Public License
grunt	MIT License
jackson	Apache License 2.
java	GNU General Public License/Java Community Process
Jersey is dual licensed under 2 OSI approved licenses	COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL - Version 1.1)/ GNU GENERAL PUBLIC LICENSE (GPL Version 2, June 1991)
log4j	Apache-Lizenz 2.0
Maven	Apache-Lizenz 2.0
nodejs	MIT-Lizenz
npm	Artistic License 2.0
postgresql	BSD 2-clause
postgresql	PostgreSQL Lizenz
tessWms	CBIL
yeoman	BSD

Publikationsliste

Autoren: Muharrem Aydinli, Chunguang Liang, Thomas Dandekar

Title: *Promoter Motif and conserved module analysis in DNA and RNA using AlModules*

Manuskript eingereicht bei BMC Bioinformatics, aber noch nicht erschienen.

Bestätigung des Journals vom 17. Juli 2021 liegt vor.