Diplomarbeit

Fachbereich: Fachbereich 1 - Life Sciences and Engineering
Studiengang: Bioinformatik
Titel: Qualitätsmanagement ribosomaler RNA Sequenzen in der SILVA Datenbank

durchgeführt am Max Planck Institut für Marine Mikrobiologie Bremen

Betreuer (FH): Prof. Dr. rer. nat. Antje Krause
Betreuer (extern): Prof. Dr. Frank Oliver Glöckner
Vorgelegt von: Timmy Schweer
geb. am: 23.05.1982
Bremen, den 06.01.2011

Vorwort

Ich möchte mich an dieser Stelle bei den Menschen bedanken, die mir während meiner Zeit im Max Planck Institut in Bremen zur Seite standen. Mein Dank gilt besonders denen, die mich unterstützt haben und sich die Zeit nahmen, mich zu belehren. Frank Oliver Glöckner, der Kopf der ″Molecular Genomics Group″ermöglichte mir die Vollendung meines Studiums in seiner Gruppe und bekräftigte mich mit Vorschlägen und Ideen zu der Arbeit am SILVA Projekt.

In meiner anfänglichen Praktikumszeit wurde ich im Megx Team sehr entgegenkommend von Renzo Kottmann und Ivaylo Kostadinov betreut, unterwiesen und eingearbeitet. Später im SILVA Team halfen mir Christian Quast und Elmar Prüße mich zurecht zu finden. Ich danke Elmar Prüße für seine kurzen aber lehrreichen Ausflüge in die Programmierung, Anwendungsentwicklung und Datenbankmanagement. Während meiner Diplomarbeit übernahm Christian Quast einen großen Teil der Betreuung. Ich danke ihm für die Einarbeitung in die SILVA Pipeline und die Geduld, die er dabei an den Tag legte.

Einen Pokal für die besten Bürogenossen haben sich Jost Waldmann und Kai Finster verdient, ich danke ihnen für ein einmaliges Arbeitsklima und eine tolle Zeit. Besonders Jost hat mir mit seiner Rechtschaffenheit und seiner offenkundigen Meinung zu vielen Dingen sehr geholfen.

Vielen Dank auch an alle anderen Menschen die mich positiv beeinflusst haben: Melissa Duhaime, Wolfgang Hankeln, Dennis Fink, Carsten John, Michael Richter, Mina Bizic, Marc Weber und Christine Klockow.

Bremen, den 02.01.2011

Timmy Schweer

Inhaltsverzeichnis

Abkürzungsverzeichnis

API	Application Program Interface
BLAST	Basic Local Alignment Search Tool [Alt90]
CSV	Comma Separated Values
DNA	Desoxyribonukleinsäure
DSMZ	Deutsche Sammlung für Mikroorganismen und Zellkulturen [dsm]
DRMAA	Distributed Resource Management Application API
EMBL	European Molecular Biology Laboratory [emb]
ENA	European Nucleotide Archive [emb]
HMM	Hidden Markov Model
IUPAC	International Union of Pure and Applied Chemistry
MGG	Molecular Genomics Group des MPI Bremen
MPI	Max Planck Institut
NCBI	National Center for Biotechnology Information [ncb]
PCR	Polymerase Chain Reaction
RDP	Ribosomal Database Project [rdp]
rRNA	ribosomale Ribonukleinsäure
RNA	Ribonukleinsäure
SGE	Sun Grid Engine
SSU	Small Subunit - kleine Untereinheit der rRNA
LSU	Large Subunit - große Untereinheit der rRNA

LTP Living Tree Project [YRP+08, YLE+10]

SINA SILVA Incremental Aligner [Pru07]

PCR Polymerase Chain Reaction

POSIX Portable Operating System Interface [for uniX]

PT-Server ARB positional tree server

Kurzzusammenfassung

Das SILVA Projekt veröffentlicht regelmäßig aktuelle rRNA-Datenbanken mit alignierten Sequenzdaten der kleinen (16S/18S, SSU) und großen (23S/28S, LSU) Untereinheiten der Ribosomen von allen drei Domänen des Lebens: *Archaea*, *Bacteria* und *Eukarya*. SILVA verwendet die Sequenzdaten von EMBL/ENA und verknüpft diese mit Metadaten aus folgenden Quellen: *greengenes*, *LTP*, *Megx*, *EMBL*, *RDP*, *DSMZ* und *StrainInfo*. Für die Nutzung der rRNA Datenbank können verschiedene Dienste der Webseite *http://www.arb-silva.de* genutzt werden oder die ARB Software. Die ARB Software bietet ein umfangreiches Repertoire an Werkzeugen für die Arbeit mit Sequenzdaten und phylogenetischen Bäumen. Diese Arbeit gibt eine Einführung in die Grundlagen der SILVA Pipeline und erklärt das erforderliche Hintergrundwissen um die Prozesse nachvollziehen zu können. Anschließend wird im speziellen das Qualitätsmodul der SILVA Pipeline veranschaulicht und Optimierungsmöglichkeiten ermittelt und implementiert. Das Qualitätsmodul der SILVA Pipeline überprüft rRNA-Sequenzen auf bestimmte Fehlerspektren und weist ihnen einen entsprechenden Qualitätswert zu oder filtert sie heraus. Behandelt werden repetitive Nukleotide (*homopolymer*), mehrdeutige Basen (*ambiguous bases*) und vektorkontaminierte Sequenzen (*vector contaminated*). Am Ende der Arbeit werden die Veränderungen welche in der SILVA Pipeline vorgenommen wurden gezeigt, die Ergebnisse grafisch dargestellt und ausgewertet. Die Ergebnisse basieren auf den EMBL/ENA Sequenzdaten in der Version 102 und vergleichen zwei SILVA Datenbanken, die aus der bisherigen und der optimierten SILVA Pipeline entstanden sind.

1 Einleitung

In diesem Kapitel der Arbeit soll das Hintergrundwissen vermittelt werden, welches dabei hilft, die Arbeitsschritte im Hauptkapitel nachzuvollziehen. Als erstes soll ein Einblick in die Funktion von Ribosomen in Bezug auf die Biologie und der Phylogenie vermittelt werden (Kapitel 1.2). Anschließend wird die ARB-Software erläutert, wofür sie entwickelt wird und welche Module von ihr eine Rolle im SILVA-Projekt spielen (Kapitel 1.3 ab Seite 4). Nachfolgend gibt es eine Einführung in das SILVA-Projekt selbst, in der Grundfunktionen beschrieben werden und der Zusammenhang zur ARB-Software erklärt wird (Kapitel 1.4 ab Seite 6). Danach folgt ein Einstieg in das Qualitätsmanagement von Sequenzdaten mit den in SILVA verwendeten Qualitätsmerkmalen und am Ende der Einleitung werden die verwendeten Programmiersprachen (Seite 12) beschrieben sowie das *cloud-computing* erklärt (Seite 13).

1.1 Grundbegriffe

Die folgenden Begriffe finden häufige Verwendung in dieser Arbeit und werder daher genauer erläutert:

Homologie steht biologisch gesehen für eine gemeinsame stammesgeschichtliche Abstammung. Wird oft im Zusammenhang mit der Ähnlichkeit zweier oder mehrerer Sequenzen untereinander mit der daraus resultierenden Verwandtschaft verwendet.

Phylogenie steht für die stammesgeschichtliche Entwicklung der Abstammung voneinander, der Entstehung der Arten auf der Erde. Phylogenetische Zusammenhänge werden oft als Baumstruktur dargestellt.

Taxonomie steht für die Bestimmung und Benennung von Organismen nach taxonomischen Kriterien (Morphologie, Physiologie und Phylogenie).

Morphologie steht in der Biologie für die Form und Struktur von Organismen.

1.2 Ribosomen

Ribosomen bilden in jedem Organismus dieser Erde die Maschinerie der Proteinsynthese. In jedem Lebewesen wird die DNA in mRNA kopiert (Transkription) und durch das Ribosom in eine Aminosäurekette übersetzt (Translation), die sich dann von selbst zu einem funktionellen Protein faltet oder mit Hilfe anderer Proteine gefaltet wird. Ein mRNA-Strang kann gleichzeitig von vielen Ribosomen translatiert werden, sie wandern entlang der mRNA und synthetisieren seitlich heraus die Aminosäurekette. Dieser Ablauf hat sich im Laufe der Jahrtausende kaum verändert, dadurch spielen Ribosomen eine große Rolle bei der Erforschung der Evolution. Es wird vermutet, dass sich die Proteinsynthese schrittweise zu dem entwickelt hat, wie wir sie heute kennen, so dass es mit der RNA begann und die DNA im Laufe der Evolution entstand. [ABH+05]

Um die benötigte Proteinsynthese einer Zelle zu gewährleisten besitzt sie eine Vielzahl an Ribosomen. Eine *E.Coli* Zelle zum Beispiel enthält circa 15,000 Ribosomen, was ein Viertel ihres Trockengewichts ausmacht. Ribosomen setzen sich immer aus zwei Untereinheiten zusammen, der kleinen Untereinheit (SSU) und der großen Untereinheit (LSU). Sie bestehen zu etwa einem Drittel aus Proteinen und zwei Drittel aus RNA ([NC05] Seite 1129). Die RNA bildet den Kern und ist hauptsächlich verantwortlich für die katalytische Funktion des Ribosom. RNA mit katalytischer Funktion wie die Ribosomen werden auch als Ribozyme bezeichnet [ABH+05]. Die Proteine eines Ribosoms befinden sich meistens an der Oberfläche, also im äußeren Bereich um den RNA Kern und sorgen dadurch für die Stabilität und die Form. Ribosomen der Prokaryoten und Eukaryoten setzen sich aus unterschiedlichen Untereinheiten zusammen (siehe 1.2.1).

Die immer gleich gebliebene Funktion von Ribosomen begründet die hohe Konservierung auf sequenzieller und struktureller Ebene, die sich aber nur auf ein paar bestimmte Regionen beschränkt. Das besondere an rRNA-Sequenzen ist die Variation der Konservierung, sie haben sich genetisch sehr unterschiedlich stark verändert.

Ein Ziel in der Phylogenie ist die Erstellung eines "Baum des Lebens"(*Tree of Life*), der die Abstammung jedes auf der Erde befindlichen Lebens aufzeigt. Wegen der hohen Variation an Konservierung in der rRNA ist sie das am besten geeignete Instrument um den Entstehungszeitraum eines Organismus zu schätzen. Die Regionen der rRNA haben sich unterschiedlich stark verändert, so deuten hochkonservierte Regionen auf eine ältere Evolution hin als die weniger Konservierten. Durch dieses Wissen kann auf die Chronologie der Evolution geschlossen werden. Nach der derzeitigen Theorie der Evolution gibt folgende Ereignisse welche eine Veränderung der Gene bewirken können:

1. **Insertion**: Einfügen eines fremden Sequenzabschnitts oder einer Base.
2. **Deletion**: Verlust eines Sequenzabschnitts bzw. einer Base.
3. **Duplikation**: Verdopplung von Sequenzabschnitten oder Basen.
4. **Inversion**: Eine Sequenz wird dupliziert und aneinandergefügt, dadurch liegt es in umgekehrter Reihenfolge vor.
5. **Translokation**: Ein Austausch von Sequenzabschnitten innerhalb zwei nicht homologer Sequenzen.

Heute dient die rRNA-Sequenz als gängige Methode um die Phylogenie eines Organismus zu erforschen oder die mikrobielle Diversität eines bestimmten ökologischen Systems zu erfassen.

1.2.1 Unterschiede der Domäne

Prokaryoten und Eukaryoten besitzen zwar unterschiedliche Ribosomen, aber bei beiden wird das Ribosom aus zwei Untereinheiten gebildet, der kleinen Untereinheit (SSU) und der großen Untereinheit (LSU).

Typ	Ribosom	LSU	SSU
Bacteria / Archaea:	70S	50S(5S,23S)	30S(16S)
Eukaryoten:	80S	60S(5S,5.8S,28S)	40S(18S)

Tabelle 1.1 – Die Strukturunterschiede der Ribosomen bei *Archaea / Bacteria* und *Eukaryoten*

Die Einheit S steht für die "Svedberg"-Konstante, die durch die Sedimentationsgeschwindigkeit der jeweiligen Untereinheit bestimmt wird. Die "Svedberg"-Konstante ist dadurch nicht von der Masse eines Moleküls abhängig, sondern von der Form und deshalb nicht einfach addierbar.

1.2.2 rRNA als Marker

Es wird seit langem daran gearbeitet den *Baum des Lebens* zu erforschen, in dem jedes Lebewesen phylogenetisch klassifiziert ist und damit aufzeigt, wie sich das Leben auf der Erde entwickelt haben könnte. In der Entwicklung der Mikrobiologie war die Klassifizierung der Mikroorganismen nach Äußerlichkeiten wegen ihrer Größe kaum möglich und es mussten neue Wege gefunden

werden, dies zu tun. Die ersten Methoden Mikroorganismen anhand der Morphologie und des Metabolismus zu klassifizieren waren sehr schwierig und nicht verlässlich anzuwenden, was zu falschen phylogenetischen Identifizierungen führte. Hinzu kommt, dass bis heute nicht jeder Mikroorganismus erfolgreich kultiviert werden kann [Oli05]. Die Klassifizierung von Mikroorganismen mit Hilfe ribosomaler RNA Sequenzen organisierte die Systematik zwar komplett neu, ist aber eine Methode die sich bis heute als die Zuverlässigste erwiesen hat. Dabei wird die rRNA als Marker eingesetzt um Mikroorganismen phylogenetisch zu klassifizieren. Je höher die Ähnlichkeit zweier rRNA-Sequenzen, desto höher ist auch die Verwandtschaft zweier Organismen. Bereits vorhandene, annotierte rRNA-Sequenzen ermöglichen eine schnelle, einfache und genaue phylogenetische Zuordnung von neuem unbekanntem genetischen Material. Gensonden (*probes*) sind kurze Sequenzfragmente, die sehr spezifisch sind für bestimmte Organismen. Mit ihnen können selbst in großen Datenbanken sehr schnell Ergebnisse erzielt werden. Gensonden haben auch den Vorteil, dass kleine Sequenzstücke ausreichen um einen Organismus zu identifizieren. Die ARB-Software (siehe 1.3) stellt für die Entwicklung und Verifizierung von Gensonden spezielle Werkzeuge zur Verfügung. [Glo]

1.2.3 RNAmmer

RNAmmer [LHR+07] ist eine Software die entwickelt wurde um rRNA-Sequenzen zu erkennen. Zur Erkennung der rRNA-Strukturen wird ein *Hidden Markov Model* (HMM) verwendet, welches mit Daten aus dem *5S Ribosomal RNA Database Project* und dem *European Ribosomal RNA Database Project* trainiert wurde. Bei der Entwicklung von RNAmmer stellte sich heraus, dass das HMM besser geeignet ist, um die Sequenzvariationen von rRNA zu erkennen als die Verwendung des BLAST Algorithmus mit einer Datenbank aus bekannten rRNA-Sequenzen. Die aus dem Projekt entstandenen HMMs stehen als Bibliothek zur Verfügung und können als Paket mit dem RNAmmer heruntergeladen oder über die Webseite verwendet werden. RNAmmer wird in SILVA verwendet um in der EMBL/ENA Datenbank [emb] Sequenzen zu erkennen die rRNA-Sequenzen enthalten, die weder in der Datenbank, noch durch das *Ribosomal Database Projekt* [rdp] als solche identifizierbar sind.

1.3 ARB-Software

ARB [LSW04] wird seit 1994 an der technischen Universität München entwickelt. Es ist eine Software mit der phylogenetische Zusammenhänge von Mikroorganismen studiert werden können. ARB basiert auf einer grafischen

Benutzeroberfläche (GUI) die für Unix/Linux Betriebssysteme entwickelt wird. Die Software integriert eine Vielzahl an unterschiedlichen Werkzeugen um Sequenzdaten zu analysieren, zu verarbeiten und phylogenetische Bäume zu erstellen. Die enthaltenen Methoden reichen von statistischen Funktionen über Alignments bis zur Verarbeitung von Stammbäumen. Einige dieser Algorithmen und Funktionen wurden aus dem GDE Projekt [SOW+] übernommen. Das für ARB entwickelte gleichnamige Datenformat ermöglicht das Speichern selbstdefinierter Metadaten die in ARB verarbeitet werden können. Die SILVA rRNA Datenbank muss von der Webseite bezogen werden, um sie bei der Berechnung von Stammbäumen und phylogenetischen Zuordnungen mit Hilfe der ARB-Software zu verwenden. ARB kann mit unterschiedlichen Datenformaten umgehen, standardmäßig werden EMBL, GenBank, DDBJ und FASTA für den Import unterstützt. Für den Import und den Export können dynamisch Datenformate selbst geschrieben werden (*EFT files*). Die Verwendung von ARB ist vielseitig; Verwandtschaftsbeziehungen von Mikroorganismen zu erkennen oder einige Proben quantitativ zu analysieren sind wohl nur ein Teil der möglichen Anwendungsgebiete. Die Ressourcen, die benötigt werden um ein performantes ARB System zu betreiben, sind stark abhängig von den verwendeten Datenbanken.

1.3.1 PT-Server

Der PT-Server (Positional Tree Server) ist ein Teil der ARB Software, er indiziert die Daten mit einem Suffixbaum und verarbeitet sequenzbasierende Suchen. Ein Suffixbaum ist eine Datenstruktur, die ein performantes arbeiten mit *String* Daten ermöglicht. Der PT-Server wird getrennt von der ARB-Software gestartet und muss für jede ARB Datenbank einen einmaligen Erstellungsprozess des Suffixbaums durchführen. Anschließend kann der PT-Server von mehreren Benutzern, lokal oder über das Netzwerk verwendet werden.

1.3.2 ARB-Probe

ARB-Probe unterstützt die Entwicklung und Validierung von Gensonden, diese dienen der gezielten Markierung und Erkennung von bestimmten Genen in Organismen. Gensonden werden komplementär zu der gesuchten Nukleotidstruktur synthetisiert und markiert, wodurch sie mit der Zielsequenz hybridisieren und nachgewiesen werden können. **Probematch** ist ein Teil der ARB-Software und kann mit Hilfe des *PT-Servers* beliebige ARB Datenbanken nach Sequenzen durchsuchen, die auf eine vom Benutzer angegebene Sequenz einer Gensonde ansprechen würden. Derzeit wird daran gearbeitet *Probematch* auf der Webseite

von SILVA (1.4.1) zur Verfügung zu stellen, um die Anwendung von Gensonden auf der SILVA Webdatenbank im Browser zu ermöglichen.

1.4 SILVA

Das SILVA-Projekt [PQK+07] entstand durch Zusammenarbeit des MPI für Marine Mikrobiologie in Bremen und der technischen Universität München, in der auch die ARB-Software (1.3) entwickelt wird. Der Name SILVA entstand aus der lateinischen Bedeutung ″Wald″im Zusammenhang mit phylogenetischen Bäumen. SILVA ist eine Datenbank die qualitätsgeprüfte und alignierte rRNA-Sequenzen enthält, sie kann mit der ARB-Software (1.3) genutzt und direkt auf der Webseite heruntergeladen werden. Die Webseite (1.4.1) bietet auch ein Teil der in ARB enthaltenen Werkzeuge wie den *SILVA INcremental Aligner* (1.4.3), einen Sequenzexporter und in Zukunft auch *Probematch* (1.3.2). Die SILVA-Datenbank enthält Sequenzdaten für alle drei Domainen des Lebens: *Archaea*, *Bacteria* und *Eukarya*. Alle Sequenzen sind auf Anomalien geprüft und enthalten Metadaten, wie unterschiedliche taxonomische Zuordnungen aus drei verschiedenen Quellen, sowie Kultur-, Typ-, Qualitäts- und Alignmentinformationen. Die genauen Quellen der Metadaten sind in Kapitel 2.1.1 auf Seite 17 angegeben.

SILVA basiert auf den Sequenzdaten der EMBL/ENA Datenbank, daher enthalten alle SILVA Versionen die jeweilige EMBL/ENA Versionsnummer auf der sie basieren. Auch die Veröffentlichungen von SILVA folgen denen der EMBL/ENA Datenbanken, die jedes Quartal erscheinen. Der gesamte Prozess der Erzeugung einer SILVA Veröffentlichung wird mit einer Pipeline realisiert, einer Abfolge von Programmen, in dem bestimmte Prozesse mit Sequenz- und Metadaten arbeiten. An der Entwicklung der Pipeline sind mehrere Entwickler beteiligt. Jeder Entwickler konzentriert sich dabei auf einen bestimmten Teil der Pipeline. Trotz der weit fortgeschrittenen Automatisierung der Pipeline werden einzelne Schritte, vor allem die Verifikation der Daten, weiterhin manuell ausgeführt. Dies führt dazu, dass die Veröffentlichung der Datensätze entwicklungs- und arbeitsaufwendig bleibt.

Von jeder SILVA Veröffentlichung wird eine *Ref* und eine *Parc* Version erstellt. Die *Ref* Datenbank enthält die hochqualitativen Sequenzen, die in ihrer vollen Länge vorliegen. Die *Parc* Datenbank dagegen enthält alle Sequenzen die länger als 300 Basen sind und ist damit sehr viel umfangreicher als die *Ref* Version.

Das verwendete ARB Datenformat macht es möglich zusätzlich zu den reinen Sequenzdaten alle Metadaten wie die verschiedenen taxonomischen

Zuordnungen, Alignment- und Qualitätsinformationen zu speichern. Darüber hinaus können eigene Metadaten definiert und in der ARB-Software angezeigt werden.

1.4.1 arb-silva.de

Die Webseite `http://www.arb-silva.de` ist ein Teil des SILVA Projekts und bietet eine Reihe von Diensten, Downloads, Informationen und Statistiken. Neben den kompletten *Parc* und *Ref* Datenbankversionen von SILVA können mit Hilfe des Browsers selektiv einzelne Sequenzen als ARB Datenbank oder FASTA Datei heruntergeladen werden. Der Browser ermöglicht die einfache Auswahl und Sammlung von Sequenzen in einem *cart*, eine Art Einkaufswagen. Auch das durchsuchen der gesamten Datenbank oder des Carts zum Beispiel nach einer bestimmten Accession Nummer, einem Publikationsjahr oder einer Sequenzlänge ist über das Webinterface möglich. Mit dem SINA Aligner (1.4.3) können auf der Webseite beliebige Sequenzen im FASTA Format hochgeladen und gegen die SILVA Datenbank aligniert werden. Das rechenaufwendige Alignment wird lokal im MPI Bremen auf einem Cluster von Rechnern ausgeführt und das Ergebnis per Datei als Download für den Benutzer bereitgestellt. Das ARB Tool *Probematch* soll in Zukunft auch über das Webinterface benutzbar gemacht werden. Es ermöglicht das Testen von Gensonden innerhalb der SILVA Datenbank.

1.4.2 Alignments

Der Grundgedanke eines Alignments besteht darin Gruppen von Sequenzen nach identischen Regionen zu durchsuchen, beziehungsweise ihre Ähnlichkeit zu bewerten. Dabei wird mit Hilfe einer Bewertungsmatrize der maximale Score gesucht, dass anhand der Bewertungsmatrize berechnete beste Alignment. Das Ziel eines Alignments ist immer dasselbe, durch das Aufdecken von Ähnlichkeiten innerhalb einer Gruppe von Sequenzen können Aussagen über gemeinsame Vorfahren gemacht werden. Der Alignmentprozess selbst kann dabei auf sehr unterschiedlichen Wegen durchgeführt werden. Zum Beispiel wird bei einem globalen Alignment versucht die gesamte Sequenz so gut wie möglich zu alignieren. Ein lokales Alignment hingegen positioniert einzelne Sequenzfragmente wodurch lokale Übereinstimmungen gefunden werden, die sich an völlig unterschiedlichen Positionen auf den Sequenzen befinden. Die Aligment Scores werden aus einer Bewertungsmatrize errechnet, die vorgibt welche Gegenüberstellung von Nukleotiden beziehungsweise Aminosäuren wie viele Punkte ergibt. Es werden für unterschiedliche Theorien und Anwendungszwecke unterschiedliche

Bewertungsmatrizen verwendet. Eines der bekanntesten Alignment Tools ist das von NCBI entwickelte BLAST oder BLAST+. Zur Erhöhung der Geschwindigkeit benutzt es eine heuristische Methode zur Berechnung des Alignments. Bei heuristischen Methoden wird die Genauigkeit einer Lösung zugunsten des Rechenaufwandes verringert. Eine heuristische Methode liefert nicht immer das optimale Ergebnis, ist aber weniger rechenaufwendig und dadurch schneller auszuführen. BLAST liefert eine Vielzahl von Ergebnissen bestehend aus allen Treffern, die anschließend ausgewertet werden können. [Alt90]

1.4.3 SINA Aligner

Der *SILVA Incremental Aligner* (SINA) ist im Rahmen der Diplomarbeit von Elmar Prüße [Pru07] am MPI Bremen entwickelt worden. SINA aligniert jede SILVA Sequenz gegen das SILVA SEED (1.4.4). Die resultierenden Alignmentbereiche (Start und Stop) markieren den Bereich einer Sequenz, der ribosomale RNA enthält. SINA aligniert nach folgendem Schema: es werden für jede *query* Sequenz fünf bis vierzig ähnlichen Sequenzen gesucht die bestimmten Kriterien entsprechen wie zum Beispiel der Sequenzlänge. Die Suche nach ähnlichen Sequenzen wird mit dem PT-Server (1.3) ausgeführt, der auf dem SILVA SEED (1.4.4) arbeitet. Die gefundenen Sequenzen dienen als Referenz für das Alignment. Die *query* Sequenz wird solange gegen die Referenzen aligniert bis der Alignment Score einem bestimmten Erwartungswert entspricht. Die in SINA verwendete Bewertungsmatrize wurde statistisch aus den existierenden manuell durchgeführten Alignments errechnet.

SINA erfüllt die drei wichtigsten Kriterien um sich für die SILVA Pipeline zu qualifizieren: Er kann große Sequenzmengen bewältigen, er ist in die ARB Softwareumgebung implementiert und das Alignment entspricht fast dem eines Experten. Hinzu kommt noch die gute Performanz von SINA, im Verhältnis zur Sequenzmenge wurde eine lineare Steigerung der benötigten Rechenzeit erreicht. Die weitere Erläuterung der Funktion von SINA übersteigt den Umfang dieser Arbeit, es wird daher nicht genauer darauf eingegangen.

1.4.4 SILVA SEED

Das SILVA SEED ist das Referenzalignment für die gesamte Pipeline, es enthält nur hochqualitative von Hand ausgewählte rRNA-Sequenzen. Das SEED wurde ursprünglich aus dem ARB Projekt übernommen und ist Produkt von mehr als zwanzig Jahren Arbeit. Jede Sequenz in der SILVA Pipeline entspricht zu einem gewissen Teil mindestens einer Sequenz aus dem SILVA SEED, das heißt: Umso phylogenetisch vielfältiger das SEED ist, desto reicher kann auch

die Phylogenie der gesamten SILVA Datenbank werden, vorausgesetzt die EMBL/ENA Datenbank ermöglicht dies. Das SILVA SEED enthält Sequenzen von allen drei Domainen des Lebens: *Bacteria*, *Archaea* und *Eukaryoten*. Zur Zeit enthält das SILVA SEED insgesamt 57,445 Sequenzen mit einer Länge von 151 bis 4,544 Basenpaaren. Die durchschnittliche Sequenzlänge beträgt 1,591 Basenpaare.

1.5 Qualitätsmanagement

Die Qualitätskontrolle bei Sequenzdaten wird durch den enormen Wachstum der weltweiten Gendatenbanken wie NCBI oder EMBL/ENA von immer größer werdender Bedeutung [Qua09]. Mit sinkendem Zeitaufwand liefern neue Sequenzierungstechniken immer größere Mengen an Sequenzdaten [HHM+07]. Große Institutionen wie EMBL und NCBI sind dafür verantwortlich Qualitätskontrollen zu etablieren und die Sequenzdaten bei der Veröffentlichung zu verifizieren. Ein Grund für fehlerhafte oder kontaminierte Sequenzen könnte sicherlich sein, dass die Veröffentlicher oft unter hohem Zeitdruck stehen und die Mühe der ausführlichen Qualitätskontrolle der Sequenzen scheuen. Erst nach einer Veröffentlichung wird einer Sequenz eine eindeutige Accession Nummer zugeteilt und erst dann ist die Publikation des zugehören Artikels (*papers*) möglich. Die Qualitätskontrolle bei reinen Sequenzdaten gestaltet sich schwierig, da viele Fehlerquellen vorhanden sind und es auch oft nicht eindeutig ist, ob es sich um Sequenzierungsfehler handelt oder biologische Ursachen hat. Eine ausführliche Dokumentation der Sequenzierungsprozesse und der Mitveröffentlichung dieser Informationen könnte Sequenzanomalien auch nach langer Zeit noch nachvollziehbar machen. Durch die Verwertung der zusätzlichen Informationen, lassen sich bestimmte Fehler auf spezifische Merkmale wie zum Beispiel die Sequenzierungsmethode determinieren. Die bestehenden Sequenzdatenbanken von NCBI und EMBL/ENA entstanden aus Sequenzierungsprozessen, die sich laufend weiterentwickelt und verändert haben. Dadurch können systematische Fehler die bei bestimmten Sequenzierungsprozessen entstehen schlecht ausgemacht werden und die Qualitätskontrolle kann keine Fehlerquellen von spezifischen Sequenzierungsprozessen berücksichtigen.

Bei jedem Prozess der Sequenzierung ist es wichtig, dass die fertigen Sequenzen auf Anomalien wie chimäre Fragmente, repetitive Teile oder Vektorkontamination überprüft werden. Wenn Sequenzierungsfehler unerkannt bleiben oder Vektorfragmente unwissend in der Sequenz bestehen kann das unerwünschte Folgen für ein Projekt haben. Mögliche Auswirkungen von fehlerhaften oder kontaminierten Sequenzdaten auf die Forschungsergebnisse [NCB09]:

- **Nutzloser Zeit- und Arbeitsaufwand**: Die Deutung einer Studie die auf fehlerhaften Sequenzen basiert kann zu falschen Ergebnissen und damit nutzlosem Verbrauch von Zeit, Finanzmitteln und Arbeitskraft führen.

- **Falsche Aussagen über die biologische Bedeutsamkeit**: Die Analysen einer Studie die auf fehlerhaften Sequenzen basieren führen zu irreführenden oder falschen phylogenetischen Zuordnungen. Die Folge ist die Annahme von falschen biologischen Kohärenzen.

- **Fehlerhafte Assemblierung von Contigs und falsche Gruppierung von ESTs**: Sequenzen die mit demselben Fremdmaterial kontaminiert sind könnten bei der Clusterbildung in die gleiche Gruppe geordnet werden, obwohl sie eigentlich keine Homogenität aufweisen oder bei der Assemblierung an völlig falschen Stellen zusammengefügt werden.

- **Fehlerfortpflanzung auf andere Projekte**: Das Publizieren von kontaminierten Sequenzen in öffentliche Datenbanken führt zu einer Fehlerfortpflanzung. Alle weiteren Analysen, welche die kontaminierten Sequenzen mit einbeziehen, können zu falschen Ergebnissen führen.

1.5.1 Mehrdeutige Basen (ambiguous bases)

Mehrdeutige Basen werden verwendet, wenn bei dem Sequenzierungsprozess nicht eindeutig ist welche Base an einer bestimmten Position vorkommt. Es gibt zwei Ursachen die eine Verwendung von mehrdeutigen Basen begründen, sie können im Prozess der Sequenzierung entstehen oder eine Varianz der genetischen Sequenzen innerhalb einer Gruppe von Organismen derselben Spezies darstellen. Ein Chromatogramm zeigt die Fluoreszenz-Signale der vier Basen nach einer Sequenzierung, dabei hat jede Base ein eigenes Farbsignal. Es kann vorkommen, dass die Signale sich überschneiden oder sogar vollständig überdecken, dadurch kann nicht mehr eindeutig festgestellt werden, um welche Base es sich handelt. Diese Fehler treten bei unterschiedlichen Sequenzierungsprozessen unterschiedlich häufig auf. [HHM+07]

Für die Darstellung einer nicht eindeutigen Basen werden Platzhalter verwendet. Jeder Platzhalter steht für eine bestimmte Gruppe von Nukleotiden. Welches Zeichen für welche Basen steht, wurde mit dem *IUB/IUPAC* Standard [BN74] festgelegt und wird in der folgenden Tabelle 1.2 dargestellt.

Zeichen	Base	Zeichen	Base
A	Adenin	M	A C (Amino)
C	Cytosin	S	G C
G	Guanin	W	A T
T	Thymin	B	G T C
U	Uracil	D	G A T
R	G A (Purin)	H	A C T
Y	T C (Pyrimidin)	V	G C A
K	G T (Keto)	N	A G C T

Tabelle 1.2 – Nach dem IUPAC Standard [BN74] werden die hier dargestellten Zeichen für mehrdeutige Basen in DNA/RNA Sequenzdaten verwendet.

1.5.2 Vektorkontamination (vector contamination)

In der Biotechnologie werden Vektoren als Transportmittel benutzt, um Gensequenzen in fremde Zellen einzuschleusen. Am häufigsten werden virale Vektoren oder Plasmide benutzt, um gezielt Gene in eine Zelle zu integrieren. Plasmide sind von Bakterien isolierte zirkuläre DNA Moleküle, die getrennt von den Chromosomen vorliegen in die gezielt Gensequenzen integriert werden können. Sie haben ihre eigene Translationsaktivität die durch eine hohe Anzahl positiv beeinflusst wird. Plasmide können eine hohe Translationsrate erreichen und besitzen im Vergleich zu viralen Vektoren die Fähigkeit lange Sequenzen aufnehmen zu können. Virale Vektoren funktionieren nach einem anderem Prinzip, sie integrieren ihre Fracht direkt in die Chromosomen einer Zelle und verändern damit das bestehende Erbgut.

Für die Sequenzierung werden die zu sequenzierenden Gene oder Regionen in einer hohen Anzahl und Reinheit benötigt. Dazu wird das Gen in einen Vektor integriert, in eine Zelle geschleust und mit Hilfe der modifizierten Zellen durch die Zellteilung vermehrt. Bei der anschließenden Isolierung des Gens werden Restriktionsenzyme verwendet, welche das Gen aus dem Vektor heraustrennen. Die verwendeten Restriktionsenzyme schneiden oft ungenau weshalb Vektorfragmente sehr häufig mitsequenziert werden. Werden diese Vektorfragmente vor der Publikation nicht erkannt und entfernt, kommen die Vektorreste mit in die öffentlichen Sequenzdatenbanken wie NCBI und EMBL. [EBI10]

1.5.3 Repetitive Nukleotide (repetitive nucleotides)

Tritt dasselbe Nukleotid an mehreren direkt aufeinander folgenden Positionen in einer Sequenz auf, ist dies mit hoher Wahrscheinlichkeit nicht natürlichen Ursprungs. Diese sogenannten Homopolymere entstehen bei der Sequenzierung, wenn dieselbe Base zu oft gelesen wird oder die Anzahl der identischen Basen nicht eindeutig interpretiert werden kann. Bei den unterschiedlichen Sequenzierungsmethoden ist dieses Problem unterschiedlich häufig [HHM+07].

1.5.4 Chimäre Sequenzen (chimera sequences)

Chimäre Sequenzen bestehen aus mehreren eigentlich nicht zusammengehörigen Sequenzteilen, dabei ist es unerheblich ob diese Teile aus demselben Organismus stammen oder nicht, wodurch sich die Erkennung denkbar schwierig gestaltet. Chimäre entstehen meistens bei der Amplifikation von DNA/RNA mit der PCR oder durch Verunreinigung der Proben durch fremde DNA. In der SILVA Pipeline wird zur Erkennung von Chimären eine modifizierte Version der Pintail [ACF+05] Software verwendet. Pintail wurde explizit entwickelt um Chimäre in 16S rRNA-Sequenzen zu erkennen, benötigt keine Lizenz und der Quellcode ist frei zur Verwendung. Mit Hilfe von Pintail werden für jede Sequenz bis zu zehn nahe Verwandte gesucht, herrscht phylogenetisch zwischen diesen eine große Distanz, erhöht sich die Wahrscheinlichkeit, dass es sich um eine chimäre Sequenz handelt. Es wird für jede Sequenz ein *"pintail quality"*Wert angegeben, der unabhängig von dem SILVA Qualitätswert ist. Zur Zeit arbeitet Jan Gerken im MPI Bremen an der Weiterentwicklung der SILVA Chimera Kontrolle, um es auch für die rRNA-Sequenzen von 23/28S und *Eukaryoten* einsetzen zu können.

1.6 Skriptsprache Python

Python ist eine Skriptsprache, wie bei fast allen Skriptsprachen muss für das entsprechende Betriebssystem ein Interpreter installiert werden, welcher den Quellcode in Maschinencode übersetzt und ausführt. Python kann auf Windows, Linux/Unix und Mac OS X Betriebssystemen installiert werden und ist dank der "Open Source Initiative"(OSI) Lizenz kommerziell frei nutzbar. Die Stärken von Skriptsprachen im Allgemeinen liegen in der schnellen Erlernbarkeit, der hohen Produktivität und der Möglichkeit den geschriebenen Code schnell zu debuggen da immer direkt der Quellcode vorliegt. Häufig werden Skriptsprachen zur Steuerung von anderen Programmen verwendet die in statischen Programmiersprachen wie C, C++ und Java geschrieben sind. Python hat

ein großes Repertoire an Standardbibliotheken und eine starke Community die für viele Anwendungsbereiche selbst geschriebene Bibliotheken bereitstellt. *http://www.python.org*

1.7 Statistiksprache R

R ist eine Skriptsprache für statistisches Rechnen und die Entwicklung von Graphen. Sie bietet eine Reihe von statistischen und grafischen Standardmethoden und ist durch die Möglichkeit zusätzliche Bibliotheken zu installieren einfach zu erweitern. Alle Graphen in dieser Diplomarbeit wurden mit dieser Sprache erstellt. R kann nach der GNU Lizenz frei genutzt werden und existiert für die Betriebssysteme Linux/Unix, Windows und Mac OS X. *http://www.r-project.org*

1.8 Cluster-Computing

Grundsätzlich geht es beim *Cluster-Computing* um die gemeinsame Nutzung von Rechnerressourcen. Es werden beliebig viele Knoten über ein lokales Netzwerk zu einem *Cluster* verbunden. Die Prozesse werden als Jobs innerhalb des Clusters auf die einzelnen Rechenknoten aufgeteilt und parallel abgearbeitet. Die Verteilung der Jobs auf die einzelnen Knoten ist ein Komplexer Prozess und muss entsprechend verwaltet werden. Für das Betreiben und Verwalten eines *Cluster* gibt es unterschiedliche Software, wobei die optimale Software den Anforderungen nach gewählt werden muss. Ein klarer Vorteil des *Cluster-Computing* ist die Erweiterungsmöglichkeit. Werden mehr Rechenressourcen benötigt, müssen lediglich neue Rechner gekauft und zu dem bestehenden *Cluster* hinzugefügt werden. *Cluster-Computing* bietet für die Produktivität und Prozessgeschwindigkeit riesige Vorteile und ist in der *Molecular Genomics Group* (MGG) am Max Planck Institut (MPI) Bremen fester Bestandteil der Arbeitsprozesse.

1.9 Sun Grid Engine (SGE)

In der Bioinformatik wird meistens mit sehr großen Datenmengen gearbeitet, was die benötigten Rechnerressourcen in die Höhe treibt. Im MPI Bremen wird eine Sun Grid Engine (SGE) zur Steuerung des Rechen-*Cluster* (siehe 1.8) mit 36 Rechnerknoten (*Nodes*) betrieben die allen Angestellten der *Molecular Genomics Group* (MGG) zur Verfügung steht. Die SGE ist zuständig für die

Annahme, die Vorbereitung, die Verteilung, die Verwaltung und die Ausführung aller Jobs, die von Benutzern eingereicht werden. Sie sorgt für den korrekten Ablauf und die optimale Auslastung der Prozesse auf den vorhandenen *Nodes*. Die SGE besitzt eine einfache Schnittstelle, Jobs können per Kommandozeilenbefehl oder vorhandener *API* an den *GridEngine Master* übergeben werden. Der *GridEngine Master* ist der Kopf der SGE, er nimmt alle Befehle der Benutzer entgegen und regelt die Verteilung und Überwachung. Für viele Programmiersprachen wie zum Beispiel C, C++ und Java kann die DRMAA API verwendet werden, um Jobs auf der SGE über Programme zu kontrollieren. Die *Distributed Resource Management Application API* (DRMAA) ist eine Standardschnittstelle um mit verschiedenen *Distributed Resource Management Systemen* (DRMS) wie zum Beispiel der Sun Grid Engine zu kommunizieren. Eine DRMAA Bibliothek ist für die meisten Programmiersprachen im Internet zu finden, genauere Informationen findet man auf *www.drmaa.org*. [sge]

2 Optimierung des SILVA Qualitätsmanagement

2.1 Die SILVA Pipeline

Die hier gesammelten Informationen stammen aus der Doktorarbeit von Christian Quast [Qua09], aus dem SILVA Paper [PQK+07] und den eigenen Erfahrungen mit der SILVA Pipeline. Wenn man in der Informatik von einer Pipeline spricht, dann handelt es ich um einen Ablauf von bestimmten Prozessen. In der SILVA Pipeline sind das Prozesse zur Erkennung, Verarbeitung und Filterung von rRNA-Sequenzen. Die SILVA Pipeline besteht aus unterschiedlichen Modulen die nacheinander ablaufen:

- **Import der Sequenz- und Metadaten(2.1.1)**
- **Qualitätskontrolle der importierten Sequenzen(2.1.2)**
- **Alignment der Sequenzen anhand des SILVA SEED (2.1.3)**
- **Verknüpfung der Sequenzdaten mit den Metadaten, anschließender Export der Datenbank für die Webseite und ARB (2.1.4)**

Die einzelnen Module können nur strikt hintereinander ausgeführt werden, intern teilen die Module die Sequenzdaten auf mehrere Blöcke auf um einen parallelen Prozessablauf zu ermöglichen und auf dem Rechen-*Cluster* (1.8) entsprechend zu skalieren. Die Anzahl der Prozesse und der Sequenzen die sie bearbeiten, ist ausschlaggebend für eine gute Skalierung. Die Parameter und Einstellungen für jedes Modul werden in einer zentralen Konfigurationsdatei festgelegt. Vor der Nutzung der SILVA Pipeline muss die Konfigurationsdatei angepasst werden. In Grafik 2.1 wird ein Ablaufdiagramm der bisherigen SILVA Pipeline gezeigt.

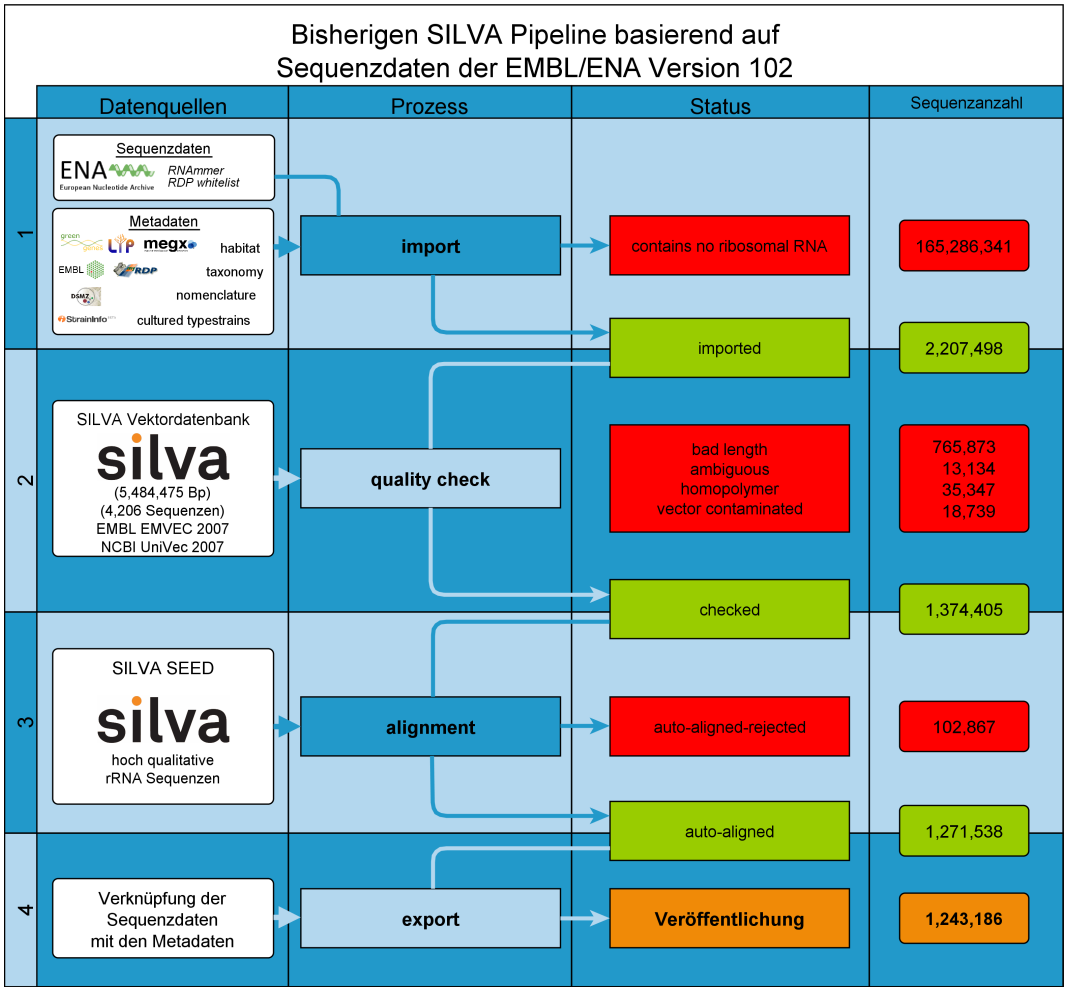

Abbildung 2.1 – Die Grafik zeigt den Ablauf der bestehenden SILVA Pipeline vor dieser Arbeit, die Zahlen stammen aus der SILVA102 Version. In den roten Feldern stehen die Herausgefilterten, in den grünen Feldern die Positiven. Bei der Veröffentlichung werden für den Export noch einmal verschiedene Filter angewendet um unterschiedliche Versionen zu produzieren (*parc* oder *ref* siehe 1.4). Eine ähnliche Übersicht mit den Veränderungen die während dieser Arbeit durchgeführt wurden ist auf Seite 40 im Kapitel 3.1 dargestellt.

2.1.1 Import

In diesem Modul werden Sequenz- und Metadaten in die Datenbank importiert. Die Sequenzdaten stammen aus der EMBL/ENA Datenbank, die Metainformationen aus folgenden unterschiedlichen Quellen:

- **Taxonomische Klassifizierungen: EMBL [emb], greengenes [gre] und RDP [rdp]**
- **Nomenklatur: DSMZ (Deutsche Sammlung für Mikroorganismen und Zellkulturen) [dsm]**
- **Typ- und Kulturinformationen: LTP [YRP$^+$08, YLE$^+$10], RDP [rdp], straininfo [DVMS05] und EMBL [emb]**

Die Verknüpfung der Metadaten mit den Sequenzinformationen findet hauptsächlich anhand der Accession Nummern statt. Grundsätzlich werden alle Metainformationen der Quellen in die Datenbank importiert und erst im Export Modul (2.1.4) mit den passenden Sequenzen verknüpft.

Bei der Importierung der **Sequenzdaten** werden als erstes alle als SSU/LSU und *"generic"*rRNA markierten Sequenzen importiert. Die Suche nach rRNA-Sequenzen gestaltet sich aber etwas komplexer als das einfache Filtern nach der Beschreibung einer Sequenz. Der Vergleich zwischen den als rRNA markierten Sequenzen aus EMBL/ENA und RDP anhand der Accession Nummern zeigte, dass in der EMBL/ENA Datenbank viele Sequenzen vorhanden sind die rRNA enthalten, aber nicht als solche deklariert sind. Dieses Problem wird mit der Erstellung einer "Whitelist"gehandhabt, welche alle Accession Nummern aus der RDP Datenbank enthält und diese dann aus der EMBL/ENA Datenbank importiert. Zusätzlich wird die Software RNAmmer [LHR$^+$07] dazu verwendet rRNA Strukturen zu finden und in die Datenbank zu übernehmen. Genauere Informationen zu der RNAmmer Software gibt es im Abschnitt 1.2.3 auf Seite 4.

2.1.2 Qualitätskontrolle

Die Qualitätskontrolle in der SILVA Pipeline dient der Prüfung der Sequenzen auf bestimmte Fehlerspektren und Qualitätsmerkmale. Diese Arbeit befasst sich hauptsächlich mit diesem Modul, daher wird dieser Prozess in den folgenden Kapiteln genau beschrieben. Die Erkennung von Chimären wird wegen ihrer Komplexität in dieser Arbeit nicht behandelt, es besteht bereits ein Ansatz, wie in Kapitel 1.5.4 beschrieben, dieser ist aber noch ziemlich unzuverlässig. Vor der Qualitätskontrolle werden alle Sequenzen, die kürzer als 300 Basenpaare sind

herausgefiltert. Der Erfahrung nach enthalten sie zu wenig genetische Informationen um phylogenetisch zuverlässig zugeordnet zu werden. Danach wird jeder Sequenz ein Qualitätswert zugewiesen der von folgenden Sequenzeigenschaften abhängig ist:

- **Mehrdeutige Basen (Seite 10)**
- **Vektorkontamination (Seite 11)**
- **Repetitive Basen (Seite 12)**
- **Chimäre Sequenzen (Seite 12)**

Jede Qualitätskontrolle berechnet den entsprechenden Prozentwert, vergleicht diesen mit dem vorgegebenen Grenzwert und markiert gegebenenfalls die Sequenz als fehlerhaft. Die Grenzwerte für die Qualitätskontrolle in der bisherigen SILVA Pipeline:

- **Maximal 2% mehrdeutige Basen**
- **Maximal 5% Vektorkontamination**
- **Maximal 2% repetitive Basen**
- **Minimale länge von 300 Basen**

Am Ende der Pipeline beim Export der Sequenzen werden die als fehlerhaft markierten Sequenzen nicht mit in die öffentliche SILVA Datenbank übernommen. Dadurch, dass die betroffenen Sequenzen nur markiert und nicht direkt gelöscht werden, kann immer nachvollzogen werden, wann und warum sie herausgefiltert wurden. Es wird für jede Sequenz die nicht herausgefiltert wurde ein Qualitätswert nach folgender Formel berechnet:

$$\mathbf{Sq} = 1 - \frac{\frac{\mathbf{M}}{100} + \frac{\mathbf{R}}{100} + \frac{\mathbf{V}}{100}}{3} * 100 \tag{2.1}$$

- **Sq**: Sequenzqualität
- **M**: Anteil an mehrdeutigen Basen
- **R**: Anteil an repetitiven Nukleotiden
- **V**: Anteil an Vektorkontamination

2.1.3 Alignment

Das Alignment Modul der SILVA Pipeline besteht aus der Ausführung des SINA Alignments (1.4.3) und der Übertragung der Ergebnisse in die Datenbank. Die erzeugten Ergebnisse wie Alignmentscore, Alignmentlänge, Start- und Stopp-Position werden im ARB Dateiformat gespeichert und können in der ARB-Software (1.3) verwendet werden. Alle Sequenzen die nicht aligniert werden können werden entsprechend markiert und dadurch nicht mit in die öffentliche SILVA Datenbank übernommen.

2.1.4 Export

Im Export Modul werden die Metadaten mit den Sequenzinformationen verknüpft und die endgültige SILVA Datenbank erstellt. Die importierten Metadaten (siehe 2.1.1) werden anhand der eindeutigen Accession Nummer mit den Sequenzdaten verknüpft.

Für unterschiedliche Anwendungszwecke der SILVA Datenbank werden unterschiedliche SILVA Versionen erstellt die sich in den Export Parametern unterscheiden. Die *parc* Version hat die niedrigsten Grenzwerte für die Qualität und enthält dadurch die meisten Sequenzen. Die *ref* Version besitzt höhere Grenzwerte und enthält entsprechend weniger Sequenzen von höherer Aligment- und Sequenzqualität. Je kleiner die Datenbank ist, desto niedriger sind auch die Ansprüche an die Hardware auf der sie verwendet wird. Für den Aligner Webservice wird eine optimierte SSU Datenbank verwendet, bei der alle Sequenzen entfernt wurden, die zu 99% identisch sind. Dies führt zu höherer Performanz und niedrigeren Anforderungen an den Arbeitsspeicher.

2.2 Problematiken

Folgend werden die Problematiken der drei Fehlerspektren in Bezug auf die SILVA Dienste erläutert. Die Qualitätskontrolle der SILVA Pipeline soll die aufgeführten Problematiken vermindern.

Mehrdeutige Basen: Ein Problem von mehrdeutigen Basen, insbesondere bei der Verwendung für die Phylogenie, ist die Erhöhung der Diversität verursacht durch die Mehrdeutigkeit. Sequenzen mit einem hohen Anteil an mehrdeutigen Basen sind durch die mögliche Variation der betroffenen Positionen schwieriger eindeutig phylogenetisch zu annotieren.

Vektorkontamination: Eine Vektorkontamination bei mehreren Sequenzen kann zu Ähnlichkeiten führen, die alleine durch die Vektorstruktur verursacht werden. Gleiche oder ähnliche Vektorfragmente werden bei einem Alignment als Homologie interpretiert, was dementsprechend falsche phylogenetische Zuordnungen verursachen kann.

Repetitive Nukleotide: Durch die Fehlinterpretation der Anzahl der Basen bei der Sequenzierung werden Basen eingefügt, genetisch gesehen findet eine Insertion statt. Bei einem Alignment bewirkt die Insertion eine geringere Ähnlichkeit zu echten Verwandten und kann dadurch zu einer fälschlich hohen phylogenetischen Distanz führen.

2.3 Grundlegende Änderungen

In dieser Arbeit soll die Qualitätskontrolle der SILVA Pipeline dokumentiert, validiert und optimiert werden. Ziel ist es, die bestehenden Ansätze passender zu den auftretenden Problemen zu gestalten und dadurch die Qualität der SILVA rRNA-Sequenzen zu erhöhen. In der Grafik 2.1 wird der Prozessablauf gezeigt wie er vor dieser Diplomarbeit vorzufinden war. Grundsätzlich muss für die Erkennung von Sequenzanomalien die Entstehung nachvollzogen werden um sie entsprechend zu beurteilen und zu verarbeiten.

Es gibt die in Kapitel 2.1.2 aufgeführten Arten von Qualitätsmerkmalen der SILVA Sequenzen, diese Merkmale sind aber bisher völlig unabhängig von der auftretenden Position. Bei allen Qualitätsmerkmalen sollte sich die Frage gestellt werden, ab wann sie die Informationen einer Sequenz wirklich negativ beeinflussen. Wenn die Kontamination sich auf eine bestimmte Position beschränkt, kann die Sequenz trotzdem einen hohen Anteil an korrekten genetischen Informationen enthalten.

Nach Absprache mit meinen Betreuern im MPI Bremen, Prof. Dr. F. O. Glöckner und Dr. C. Quast, kam der Gedanke, dass es besser wäre, das Alignment vor die Qualitätskontrolle zu versetzen. Dies führt zwar zu einem höheren Rechenaufwand weil vor der Qualitätskontrolle mehr Sequenzen vorhanden sind, die aligniert werden müssen, aber das stellt auf dem Rechencluster des MPI Bremen (siehe 1.8) kein Problem dar. Die Umstellung bringt den Vorteil, dass alle importierten Sequenzen auf rRNA Strukturen geprüft werden und anschließend im Qualitätsmodul auf die Alignmentinformationen zurückgegriffen werden kann. In der Qualitätskontrolle kann dadurch gezielt die Qualität der rRNA Regionen geprüft werden.

Die Grenzwerte für die bisherigen Qualitätskontrollen wurden anhand von

Graphen und Statistiken von Prof. Dr. F. O. Glöckner ermittelt. Die sogenannten *thresholds* sollten bei Änderungen im Ablauf der Qualitätskontrolle neu ermittelt werden.

Der Qualitätswert wird durch die Umstellung der Vektorkontrolle (siehe 2.5 auf Seite 22) unabhängig von der Vektorkontamination. Dies wurde so beschlossen, da die Vektorkontrolle nur noch auf den nicht alignierten Enden einer Sequenz stattfindet und dadurch keine Aussagekraft über die Qualität der rRNA-Region besitzt. Das bewirkt eine größere Änderung der gesamten Qualitätswerte in der SILVA Datenbank. Die Qualität der SILVA Sequenzen wird dann nach folgender Formel berechnet:

$$\mathbf{Sq} = 1 - \frac{\frac{\mathbf{M}}{100} + \frac{\mathbf{R}}{100}}{2} * 100 \tag{2.2}$$

- **Sq**: Sequenzqualität
- **M**: Anteil an mehrdeutigen Basen
- **R**: Anteil an repetitiven Nukleotiden

2.4 Mehrdeutige Basen

Die Kontrolle auf mehrdeutige Basen wird in der SILVA Pipeline auf allen Sequenzen angewendet, sie dient als Merkmal für die Qualität der Sequenzen und wird auch für die Berechnung des SILVA Qualitätswert verwendet (siehe Kapitel 2.1.2 auf Seite 17). Wie in Kapitel 1.5.1 auf Seite 10 beschrieben, verursachen mehrdeutige Basen unspezifischere Sequenzen und sind dadurch schwieriger eindeutig phylogenetisch zuzuordnen. Die Aussagekraft eines Alignments wird durch einen hohen Anteil an mehrdeutigen Basen verringert, sie müssen von dem Aligner berücksichtigt und entsprechend bewertet werden.

2.4.1 Prozess in SILVA102

Die derzeitige Kontrolle jeder Sequenz auf ihren Anteil an mehrdeutigen Basen gestaltet sich unkompliziert. Jede mehrdeutige Base in der gesamten Sequenz wird gezählt um anschließend den prozentualen Gesamtanteil zu berechnen. Alle Sequenzen bei denen der Anteil über einem Grenzwert liegt werden entsprechend markiert. Der festgelegte Grenzwert liegt bei 2% mehrdeutige Basen, alle die darüber liegen werden nicht mit in die SILVA Veröffentlichung exportiert.

2.4.2 Änderungen im Prozess

In diesem Modul wurde der zu kontrollierende Bereich auf den von SINA alignierten Bereich beschränkt. Es werden die mehrdeutigen Basen im alignierten Bereich gezählt und anschliessend der prozentuale Anteil anhand der Alignmentlänge ohne Gaps berechnet. Diese Änderung führt dazu, dass der berechnete Anteil an mehrdeutigen Basen die Qualität des rRNA Anteils beurteilt und nicht wie bisher die der gesamten Sequenz. Für dieses Modul werden keine weiteren sinnvollen Optimierungsmöglichkeiten gesehen, daher wurden keine weiteren Änderungen daran vorgenommen worden. Die Auswirkung der Änderung kann auf Seite 39 in Kapitel 3 betrachtet werden.

2.5 Vektorkontamination

Die Vektorkontrolle ist fester Bestandteil des Qualitätskontrollmoduls der SILVA Pipeline und dient der Erkennung des Kontaminationsgrads der Sequenzen mit Vektorresten, die nach der Sequenzierung nicht akkurat entfernt wurden. In dem Modul für Vektor Kontrolle wird BLAST dazu verwendet die Treffer zwischen den SILVA rRNA-Sequenzen und einer Vektordatenbank zu erfassen. Die Vektordatenbank muss frei von rRNA Vektoren sein, damit sichergestellt werden kann, dass ein Treffer einer SILVA Sequenz zu einem Vektor nicht auf gemeinsamen rRNA Strukturen beruht.

Die bisher verwendeten Vektor Daten sind eine Zusammenfassung der *EmVec* und *UniVec* Datenbank aus dem Jahr 2008. *EmVec* wird von EMBL zur Verfügung gestellt und enthält über 2000 von den aktuell am häufigsten verwendeten Vektoren. Äquivalent dazu gibt es die *UniVec* Vektordatenbank vom *National Center For Biotechnology Information* (NCBI), die zur Zeit ca. 3000 Einträge enthält. Beide Institutionen bieten Benutzern die Möglichkeit im Browser beliebige Sequenzen hochzuladen und auf Vektoren zu prüfen. Die Nutzung dieser Angebote kommt in der SILVA Pipeline aus Zugänglichkeits- und Performanzgründen nicht in Frage. Bei der Entwicklung der bestehenden SILVA Vektordatenbank wurden die rRNA enthaltenen Vektoren ohne Dokumentation erkannt und von Hand entfernt, es existiert kein Prozess, der die Erstellung beschleunigen und damit regelmäßige Aktualisierungen vereinfachen würde.

Die derzeitig verwendete Vektordatenbank ist veraltet und sollte für zuverlässigere Ergebnisse aktualisiert werden. Um die Vektordatenbank für die SILVA Pipeline zu erneuern, werden die Vektorsequenzdaten aus der *EmVec* und NCBI *UniVec* Datenbank zusammengefasst und Vektoren mit rRNA Anteilen entfernt. Je umfangreicher eine Vektordatenbank ist, desto sensitiver ist sie für die Erkennung von Vektoren. Durch eine optimierte Auswertung der

BLAST Ergebnisse kann eine spezifischere Erkennung durchgeführt werden. Ein BLAST Ergebnis, das eine sehr hohe Übereinstimmung (*percentage identity*) aufweist, spricht eindeutig für eine Kontamination. Alle BLAST Ergebnisse die einen niedrigere Übereinstimmung als neunzig Prozent besitzen, sollten generell verworfen werden.

Ziele in diesem Teil der Diplomarbeit sind die Erstellung und Verifizierung der Vektordatenbank zu automatisieren und die Erkennung der Vektorkontamination von rRNA-Sequenzen zu optimieren.

2.5.1 Prozess in SILVA102

Das bestehende Modul für die Erkennung der Vektorkontamination von rRNA-Sequenzen läuft nach folgendem Schema: Jede Sequenz wird mit Hilfe von BLAST gegen eine Vektordatenbank aligniert, die übereinstimmenden Basen werden gezählt und der prozentuale Anteil berechnet. Sequenzen die einen bestimmten prozentualen Wert übersteigen werden entfernt. Bei diesem Prozess treten bestimmte Zustände auf, die zu fehlerhaften Ergebnissen führen und nicht sinnvoll gelöst wurden. Sobald mehrere BLAST Treffer sich auf derselben Region überschneiden kommt es zur mehrfachen Zählung der gleichen Positionen, was zu fälschlich hohen Vektoranteilen führen kann. Vorteilhafter wäre es auch jeden BLAST Treffer nicht direkt als Kontamination zu werten sondern auch diesen auf seine Qualität zu prüfen. Die bestehende Lösung wurde unter hohem Zeitdruck erstellt und darauf getrimmt einen bestimmten Erwartungswert zu erfüllen.

2.5.2 SILVA-Vector-Importer

Eines der Ziele dieser Diplomarbeit ist die Erstellung und Verifizierung der Vektordatenbank für die Nutzung in der SILVA Pipeline zu automatisieren. Es soll den nötigen Aufwand minimieren und ein Update der Vektordatenbank bei jeder SILVA Veröffentlichung ermöglichen. Der SILVAVecImporter wird mit der Skriptsprache Python (siehe 1.6) implementiert. Das Skript sollte folgendem Programmablauf entsprechen:

1. Konvertierung von allen Sequenzdaten in das FASTA Format um die Verarbeitung mit Python zu ermöglichen.

2. Zusammenfassen der gegebenen Vektorsequenzen zur Verwendung als BLAST Datenbank.

3. Ausführen des Alignments mit vektorfreien rRNA-Sequenzen als *query* und der erstellten Vektordatenbank als *subject*.

4. Visualisierung der Ergebnisse mit R anhand derer die rRNA Vektoren ausfindig gemacht werden, die rRNA Strukturen enthalten.

5. Entfernung aller rRNA enthaltenen Vektoren aus der Datenbank.

Die Zusammenfassung der Vektorsequenzen zu einer Datenbank erfordert strikte Dateiformat Vorgaben oder die Erkennung und Konvertierung des Formats. Es sollten alle *subject* und *query* Dateien in das Standard Sequenzformat FASTA konvertiert werden. FASTA ist ein gängiges Format für Sequenzdaten, da es in reinem Textformat gespeichert wird lassen sich FASTA Dateien in einem beliebigen Texteditor öffnen und verarbeiten.

EmVec liegt standardmäßig im BLASTDB Format vor und muss ins FASTA Format konvertiert werden. Dem BLAST+ Softwarepaket liegen Programme bei, die zwischen BLASTDB und FASTA Format konvertieren können. Für die BLAST *query* Dateien, welche gegen die Vektordatenbank aligniert werden, ist es sinnvoll ARB Dateien zu unterstützen, um aus SILVA exportierte Sequenzen mit dem SILVAVecImporter verwenden zu können.

BLAST+ Prozesse können abhängig von der Sequenzanzahl sehr Rechenaufwendig und damit zeitaufwendig werden. Es ist möglich den BLAST+ Prozess per Parameter so zu konfigurieren, dass er sich auf mehrere Prozesse aufteilt um einen parallelen Ablauf mit der SGE (siehe 1.9) zu ermöglichen.

Die BLAST+ Ergebnisse sollen grafisch so dargestellt werden, dass Vektoren die rRNA Strukturen enthalten identifiziert werden können. Die Visualisierung wird mit der Statistiksprache R (siehe 1.7) umgesetzt, daher muss das BLAST+ Ergebnis in einem Format erzeugt werden, welches mit R importiert werden kann. Der Benutzer soll die Ergebnisse auswerten und entscheiden welche Vektoren entfernt werden sollten. Die Entfernung der rRNA Vektoren sollte vom Benutzer ausgeführt werden, da es schwierig ist dies mit einem Skript zu automatisieren.

Um die entgültige Vektordatenbank in der SILVA Pipeline benutzen zu können muss sie im BLASTDB Format vorliegen. Der SILVAVecImporter wird viele Parameter benötigen die vom Benutzer festgelegt werden müssen, daher sollte eine Konfigurationsdatei verwendet werden in der alle nötigen Einstellungen getätigt werden. Die Beschreibung der Parameter die in der Konfigurationsdatei einstellbar sind, sollten mit einer außenstehenden Person erarbeitet werden, um ein gut verständliches Ergebnis zu erzielen.

Implementation Für die Suche nach rRNA enthaltenen Vektoren werden rRNA Sequenzen benötigt die garantiert frei von Vektoren sind, deshalb werden für die Suche Sequenzen aus dem SILVA SEED (1.4.4) verwendet. Die Dateiformate der Sequenzdaten für den BLAST+ Prozess werden hauptsächlich anhand

der Dateiendungen erkannt. Beim BLASTDB Format müssen Dateien mit den Endungen *nhr*, *nin*, *nsd*, *nsi* und *nsq* mit demselben Namen vorhanden sein um entsprechend akzeptiert zu werden. Bei FASTA Dateien wird nach dem Größer Zeichen (>) gesucht das den Beginn eines FASTA Kopfes signalisiert. Bei ARB Dateien muss lediglich die Endung ARB vorhanden sein. Um ARB Dateien in das FASTA Format konvertieren zu können, sind zusätzliche Programme aus dem ARB Software Paket nötig, daher werden ARB Dateien nur bei vorliegender ARB Installation unterstützt. Der Installationspfad der ARB-Software muss für die Unterstützung von ARB Dateien in der Konfigurationsdatei angegeben werden. Alle Einstellungen und Parameter für den SILVAVecImporter werden über eine Konfigurationsdatei im *INI* Format gespeichert (siehe 12 unter *ConfigParser*). Für das BLASTDB Format werden zur Konvertierung Programme aus dem BLAST+ Paket benutzt, welches Grundvoraussetzung für die Benutzung des SILVAVecImporters ist. Um Kompatibilitätsprobleme zu vermeiden wird vor der Ausführung des BLAST+ Prozesses auf die korrekte **Version 2.2.23**+ kontrolliert. Die Verwendung von neueren BLAST+ Versionen kann zu Problemen führen, da sie andere Parameter benötigen könnten als vorgesehen.

Die Ausgabe der Ergebnisse von BLAST+ im CSV Format unterliegt einer Einschränkung. Es ist nicht möglich in der verwendeten BLAST Version 2.2.23+ das Feld, das die Länge der Sequenzen enthält, in den Ergebnissen ausgeben zu lassen. Bei der Verwendung von zyklisierten Vektorsequenzen (siehe 2.5.3) wird die Länge benötigt um ungültige Treffer (BLAST+ hits), die über die ursprüngliche Länge eines Vektors hinausgehen herauszufiltern. Bei Verwendung des XML Formats zur Ausgabe der Ergebnisse gibt es die Möglichkeit die Felder selbst festzulegen und damit die benötigte Sequenzlänge mit einzubeziehen. Das XML Format vergrößert die Dateigröße der Ergebnisse um das Fünffache und macht den Import und die weitere Verarbeitung in R sehr langsam. Daher wird das CSV Format für die BLAST Ergebnisse verwendet und ein Python Skript geschrieben das die Länge der einzelnen Sequenzen in einer Multi-FASTA Datei in die Kopfzeile der jeweiligen Sequenz schreibt. In R wird der entsprechende Teil dann wieder ausgelesen und kann für die Auswertung der Ergebnisse verwendet werden.

In der Struktur der FASTA Dateien muss aber nicht nur die Längeninformation der Kopfzeile hinzugefügt werden, es müssen auch Zeichen entfernt werden die zu Problemen bei der späteren Verarbeitung führen, wie Kommas oder Tabulatoren. Zusätzlich müssen im Header Schlüsselwörter für die *spike*-Sequenz, die Negativ- und die Positivkontrolle geschrieben werden, um sie für die Visualisierung entsprechend unterscheiden zu können.

Um ein Alignment mit BLAST+ zu beurteilen, muss klar sein wie ein guter

Treffer sich auf die Werte wie den *bitscore*, den *evalue* und *pident* auswirkt. Es werden Kontrollsequenzen in die BLAST *subject* und *query* Datenbank eingefügt um bestimmte Ergebnisse zu erzeugen. Die **Positivkontrolle** wird in unterschiedlichen Längen in die *subject* und die *query* Datenbank eingefügt und erzeugt damit optimale Ergebnisse mit gezielt verteilten Sequenzlängen. Die **Negativkontrolle**, bestehend aus pseudo zufallsgenerierten unterschiedlich langen Sequenzen, werden in die *subject* Datenbank eingefügt um zufällige Treffer zu erzeugen. Zusätzlich können vom Benutzer gewählte ***spike*-Sequenzen** in die *subject* Datenbank eingefügt werden, die in den Grafiken als *"spike sequence"*markiert werden. Die in dieser Arbeit verwendete *spike*-Sequenz ist eine vektorkontaminierte rRNA Sequenz die als Beispiel für einen rRNA Vektor dient. Die *spike*-Sequenz soll veranschaulichen, wie sich ein entsprechender BLAST Treffer in der grafischen Darstellung verhält.

Die Statistiksprache R (siehe: 1.7) wird dazu verwendet die entstehenden Datenmengen zu verarbeiten und entsprechend zu visualisieren. Die Ausführung des R Skripts wird mit Hilfe der *rpy2* Python Bibliothek ausgeführt, sie ermöglicht das direkte ausführen von R Code innerhalb eines Python Skripts. Der Pfad der von BLAST+ erzeugten CSV Datei wird in die Konfigurationsdatei geschrieben und anschließend mit R importiert. Die Ergebnisse aus der CSV Dateien werden in den Datentyp *data.frame* formatiert, der ähnlich einer Tabelle, Datenfelder mit Spaltennamen enthält. In diesem Schritt wird auch die Sequenzlänge aus den FASTA Headern geschnitten und in ein entsprechendes Datenfeld eingefügt. Anschließend werden die Ergebnisse nach ihren Informationen im Header aufgeteilt und getrennt voneinander verarbeitet. Die realen Treffer, die *spike*-Sequenz, die Positiv- und die Negativkontrolle werden in unterschiedlichen Farben dargestellt.

Die erzeugten Graphen werden in einem PDF Dokument gespeichert, das PDF Format unterstützt Vektorgrafiken und kann darüber hinaus auf den gängigsten Betriebssystemen identisch dargestellt werden. Um ein PDF Dokument zu betrachten wird lediglich der frei beziehbare *Adobe Reader* benötigt.

Auf der ersten Seite des PDF Dokuments ist die verwendete Konfigurationsdatei abgebildet, dadurch können die Ergebnisse jederzeit nachvollzogen werden. Anschließend werden die eigentlichen Graphen gezeigt, welche die Ergebnisse für zwei bestimmte Anwendungszwecke visualisieren. Der erste Graph (Anhang: 2.3) stellt alle BLAST Ergebnisse anhand des *bitscore* und der Länge des Alignments dar. Dieser Graph zeigt die Verteilung aller BLAST Treffer und ermöglicht eine Aussage über das Vorhandensein von Vektoren, welche rRNA enthalten. Der zweite Graph (Anhang: 4.2) entspricht genau dem ersten, nur wurden hier die BLAST Treffer nach Grenzwerten gefiltert. Alle Treffer sind nach einem *evalue* Wert kleiner als 0.05 und einem *pident* Wert von mindestens

90% gefiltert. Der dritte Graph (Anhang: 4.3) zeigt eine Listenansicht der FASTA Kopfzeile absteigend sortiert nach dem BLAST *bitscore*. In der Liste wird jeder BLAST Treffer nur einmal mit dem maximalen *bitscore* und der FASTA Kopfzeile der *subject* Sequenz angezeigt. Anhand dieser Darstellung kann die FASTA Kopfzeile der Vektorsequenz identifiziert, in der Vektordatenbank gesucht und entfernt werden. Ein Ablaufdiagramm des SILVAVecImporters wird in Grafik 2.2 auf Seite 29 gezeigt. Folgend werden die für den SILVAVecImporter verwendeten Python Bibliotheken aufgezählt und andeutungsweise erklärt:

1. **drmaa-python** [SO]: *DRMAA* ist ein API für Python um mit der SGE (siehe 1.9) zu kommunizieren und Jobs zu steuern. Es ermöglicht die Auslagerung von rechenintensiven Prozessen mittels *Cluster-Computing* (siehe 1.8) auf den Rechencluster des MPI.

2. **rpy2** [MWG]: *RPy2* ist ein Python Interface zu der R Programmiersprache für Statistik. Es können Objekte und Variablen zwischen Python und R geparst oder einfach nur R Code ausgeführt werden. Im SILVAVecImporter wird es lediglich für die Ausführung von R Skripten verwendet. Solange keine großen Datenobjekte zwischen R und Python geparst werden müssen, ist **rpy2** gut geeignet um die Stärken von Python mit R zu kombinieren.

3. **subprocess**: erlaubt das Erzeugen von neuen Prozessen innerhalb eines Python Skripts und das Handling mit deren Ein- und Ausgabepuffern.

4. **threading.Thread**: kann Python Klassen als eigene Threads starten, die parallel zum Hauptprozess arbeiten.

5. **ConfigParser**: ermöglicht das Lesen und Schreiben von Konfigurationsdateien im Windows INI Format. INI Dateien bestehen aus Sektionen (*sections*), Schlüsseln (*keys*) und Werten (*values*). Eine Sektion enthält immer einen beliebigen Namen der eckig umklammert ist (Beispiel: [main]). Jede Sektion kann beliebig viele Schlüssel und Werte Paare enthalten (Beispiel: datei=test.fasta). Der *ConfigParser* greift auf die INI Dateien zu und kann bestimmte Variablentypen wie Boolean, Integer oder String erzwingen. Die Konfigurationsdatei des SILVAVecImporters im INI Format wird im Anhang auf Seite 50 gezeigt.

6. **OptionParser**: diese Bibliothek bietet unterschiedliche Methoden um einem kommandobasierten Python Skript Parameter zu übergeben, sie zu verifizieren und wenn nötig, Fehlermeldungen zu auszugeben.

7. **signal**: mit dieser Bibliothek kann die Reaktion eines Python Prozesses auf Signale des Betriebssystem festgelegt werden. Bei allen *POSIX* kompatiblen Betriebssystem gibt es Standardsignale auf die Prozesse reagieren können oder sogar müssen. Wenn der Benutzer zum Beispiel STRG+C drückt um ein Programm zu unterbrechen, passiert im Hintergrund nichts anderes, als das dem Prozess ein SIGINT Signal gesendet wird. Eine sinnvolle Verwendung für *POSIX* Signale in einem Python Skript wäre das Auslösen eines Events wie zum Beispiel das erneute Einlesen einer Konfigurationsdatei oder das saubere Beenden aller Kindprozesse.

8. **os**: bietet viele Standardoperationen eines Betriebssystems wie Dateien löschen, Pfade zusammenfügen, Dateien auf Zugriffsrechte prüfen, auf Umgebungsvariablen zugreifen und vieles mehr.

9. **re**: eine Bibliothek für Methoden zur Verwendung von regulären Ausdrücken. Ermöglicht Strings auf bestimmte Inhalte zu testen oder sie nach einem Schema zu zerteilen.

10. **time**: wird benötigt um die Uhrzeit des Rechners auszulesen und wird meistens zum Messen von Prozesslaufzeiten benötigt.

11. **datetime**: wird zum Generieren von Zeit- und Datumsstempel für Dateinamen oder in Logdateien verwendet.

12. **random**: bietet eine Vielzahl von Methoden um pseudo Zufallszahlen zu generieren.

Anwendung Der SILVAVecImporter ist ein kommandobasiertes Skript dem eine Konfigurationsdatei per Parameter übergeben werden muss. Es liegt eine vorgefertigte Konfigurationsdatei vor (Anhang: 4.1), in welcher die Grundeinstellungen vorgenommen wurden. Einige Optionen müssen vor der erstmaligen Nutzung festgelegt werden, weil sie vom jeweiligem Benutzer und Arbeitsumgebung abhängig sind. Viele Parameter sind mit einem Kommentar versehen (Kommentarzeilen beginnen mit #), so dass dem Benutzer ein Eindruck der Funktion der jeweiligen Option vermittelt wird. Der SILVAVecImporter wurde entwickelt um Vektordatenbanken für die SILVA Pipeline zu erstellen, durch die Anpassung der Konfigurationsdatei kann er für die Anwendung und Auswertung beliebiger BLAST Alignments verwendet werden. Um eine erweiterte Hilfe zur Verwendung des SILVAVecImporters zu bekommen, muss das Skript mit dem Parameter ″–help″gestartet werden. Grundsätzlich arbeitet der SILVAVecImporter in vier unterschiedlichen Modi die jeweils einen Arbeitsschritt ausführen:

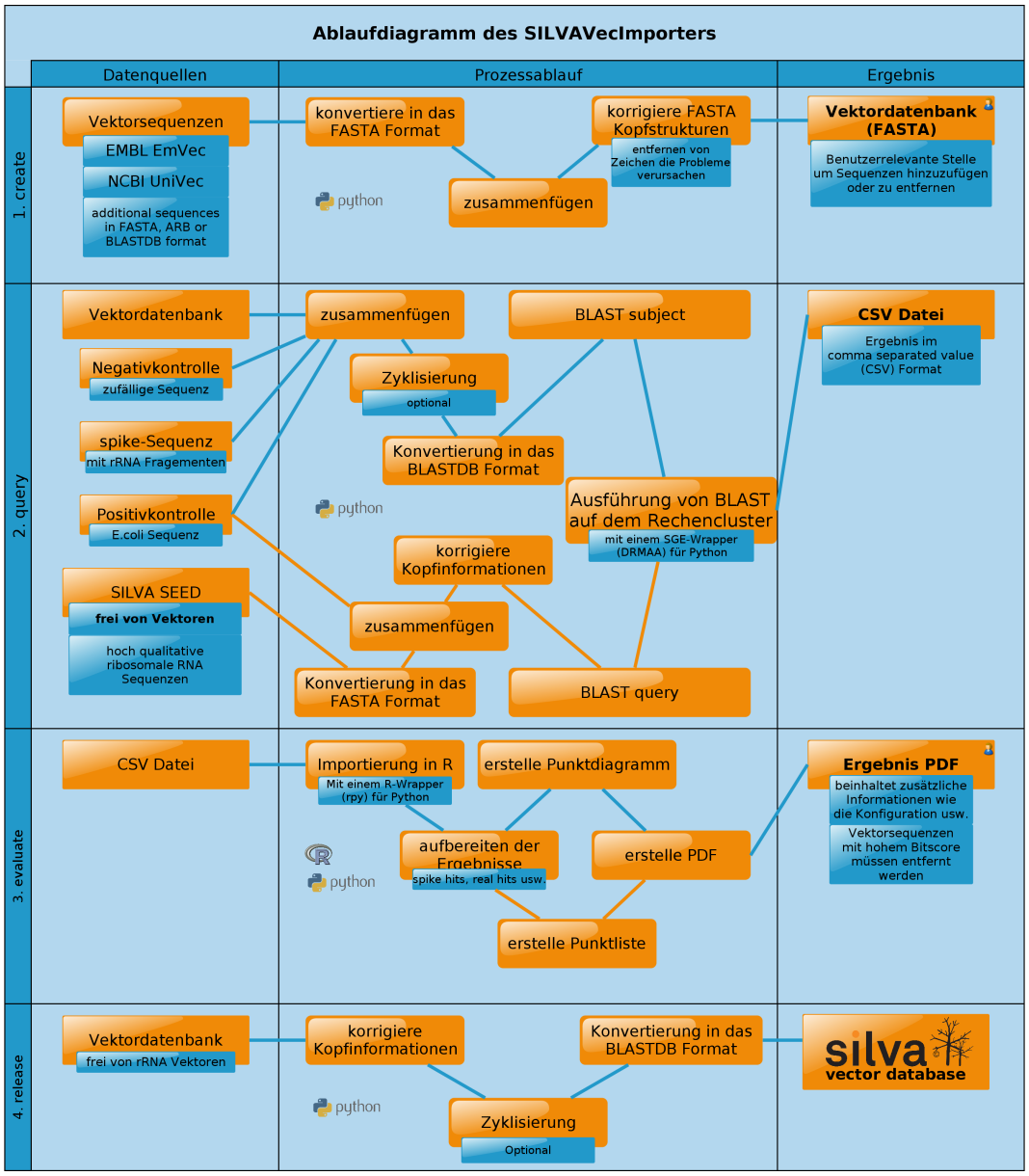

Abbildung 2.2 – Hier wird der Arbeitsprozess des SILVAVecImporters in einem Ablaufdiagramm dargestellt. Die in Zeilen aufgeteilten Arbeitsmodi *create*, *query*, *evaluate* und *release* werden in Kapitel ″Anwendung″auf Seite 30 erklärt. In der Datenquellen Spalte werden die für den Arbeitsschritt benötigten Daten aufgezählt. Die mittlere Spalte zeigt den Prozessablauf und die rechte Spalte das Ergebnis das produziert wird. Die Ergebnisse bei denen Handlungsbedarf vom Benutzer nötig ist, sind mit einem Kopf in der rechten oberen Ecke markiert.

1. Der **create** Modus erkennt die Dateiformate der Pfadangaben aus der Konfigurationsdatei in der *subject* Sektion, konvertiert diese in das FASTA Format, formatiert deren FASTA Kopfzeilen und fügt sie anschließend zu einer Datei zusammen. Zu den unterstützten Formaten gehören FASTA, BLASTDB und das ARB Format. Die Datei wird in dem Arbeitsverzeichnis mit dem Dateinamen erstellt welche in der Konfigurationsdatei definiert wurden.

2. Im **query** Modus wird der gesamte BLAST+ Prozess vorbereitet und ausgeführt. Zunächst wird überprüft ob alle Grundvoraussetzungen erfüllt sind, wie das Vorhandensein aller benötigten Dateien, die korrekte Funktion der SGE (Seite 13) und die benötigte BLAST+ Version. Dann werden entsprechende Kontrollsequenzen hinzugefügt (Positiv-, Negativkontrolle sowie *spike*-Sequenzen), die Vektorsequenzen wenn nötig zyklisiert (siehe 2.5.3), wenn nötig nochmals alle FASTA Kopfzeilen korrigiert und anschließend der BLAST+ Prozess gestartet. Nach Durchführung des Alignments werden alle Dateipfade der Ergebnisse des BLAST+ Prozesses in die Konfigurationsdatei geschrieben. Anschließend kann die Evaluation gestartet werden.

3. Der **evaluate** Modus erzeugt im Arbeitsverzeichnis des SILVAVecImporters ein PDF Dokument das die verwendete Konfigurationsdatei und die Ergebnisse enthält. Der Benutzer kann anhand der Graphen im PDF Dokument bestimmen ob die Vektordatenbank (im FASTA Format) Sequenzen mit rRNA Regionen enthält, die entfernt werden müssen. Wenn es zutrifft, dass bestimmte Sequenzen entfernt werden müssen, kann die Datenbank im Arbeitsverzeichnis des SILVAVecImporters mit einem Texteditor geöffnet und bearbeitet werden.

4. Der **release** Modus konvertiert die bestehende Vektordatenbank im Arbeitsverzeichnis des SILVAVecImporters in das BLASTDB Format um es in der SILVA Pipeline verwenden zu können. In diesem Modus muss dem Skript ein zusätzlicher Parameter für den Zielpfad übergeben werden. Es wird ebenfalls eine FASTA Datei im Zielverzeichnis erstellt, um die Datenbank im Texteditor kontrollieren zu können.

Es war nicht genügend Zeit vorhanden um den SILVAVecImporter auf Windows oder Mac Betriebssystemen zu testen. Die Konfigurationsdatei erlaubt aber die Parameter entsprechend anzupassen, so dass das Skript Plattformunabhängig funktionieren sollte.

Evaluation Anhand der erzeugten Punktdiagramme (Grafik 2.3 und Anhang 4.2) kann ermessen werden welche Vektorsequenzen rRNA Strukturen enthalten und für die Verwendung in der SILVA Pipeline entfernt werden müssen. Für den gezeigten Graph wurde BLAST+ verwendet um das SILVA SEED (1.4.4) gegen eine neu erstellte Vektordatenbank zu alignieren. Durch die Auftragung der Alignmentlänge auf der x-Achse und dem *bitscore* des Alignments auf der y-Achse bildet die Positivkontrolle eine grüne Linie die den optimalen Bereich markiert. Die Positivkontrolle besteht aus einer *E. coli* Sequenz die in unterschiedlich langen Fragmenten zerteilt und anschließend in die *query* und die *subject* Datei eingefügt wurde. Die Negativkontrolle wurde in das *subject* eingefügt und zeigt zufällige Ähnlichkeiten mit dem SILVA SEED. Die Negativkontrolle ist rot dargestellt und hat nur wenig Treffer erzeugt. Als *spike*-Sequenz wurde ein *E.Coli* Klonierungsvektor verwendet der rRNA enthält, er wurde in das *subject* eingefügt. Die Treffer der *spike*-Sequenz werden orange dargestellt und überdecken mit *bitscore* Werten von 0 bis 2.700 den gesamten Graphen. Die Treffer des SILVA SEED sind blau markiert und in der linken unteren Ecke mit *bitscore* Werten zwischen 0 und 150 zu sehen. Die Listenansicht im Anhang 4.3 zeigt die FASTA Kopfzeilen der Sequenzen absteigend sortiert nach dem *bitscore*. Jede Sequenz ist in der Liste nur einmal mit ihrem Maximalwert aufgelistet. Die besten Treffer des SILVA SEED mit der Vektordatenbank stammen von drei unterschiedlichen Vektorsequenzen und erreichten einen *bitscore* von ungefähr 150. Mit der *spike*-Sequenz verglichen ist das ein niedriger Wert, daher wurde eine Entfernung von Vektorsequenzen aus der Datenbank als nicht notwendig entschieden.

2.5.3 Zyklisierung von Vektorsequenzen

Die Ursachen für die Vektorkontamination von Sequenzdaten werden auf Seite 11 im Kapitel 1.5.2 besprochen. Bei der Erkennung von Vektorfragmenten in Sequenzen werden Vektordatenbanken verwendet, welche die Sequenzdaten in linearer Form enthalten. Die am häufigsten verwendeten Vektoren sind Plasmide und im Gegensatz zu fast allen anderen genetischen Molekülen besitzen sie eine zirkuläre Form. Bei der Erkennung können bestimmte Fälle auftreten, bei denen es von Vorteil ist diese zirkuläre Form zu berücksichtigen. Ein Ansatz um die zirkuläre Form in Alignmentprogrammen wie *BLAST+* zu berücksichtigen, ist die gesamte Sequenz noch einmal hinten anzufügen. Dadurch verdoppelt sich die Länge und es können Alignments auftreten, die das Ende und den Anfang einer Vektorsequenz mit einschließen. Es gibt bereits unterschiedliche Ansätze um die pseudo Zyklisierung von linearen Sequenzdaten in unterschiedlichen Längen durchzuführen. Es könnte bei einigen Abläufen bereits ausreichen nur

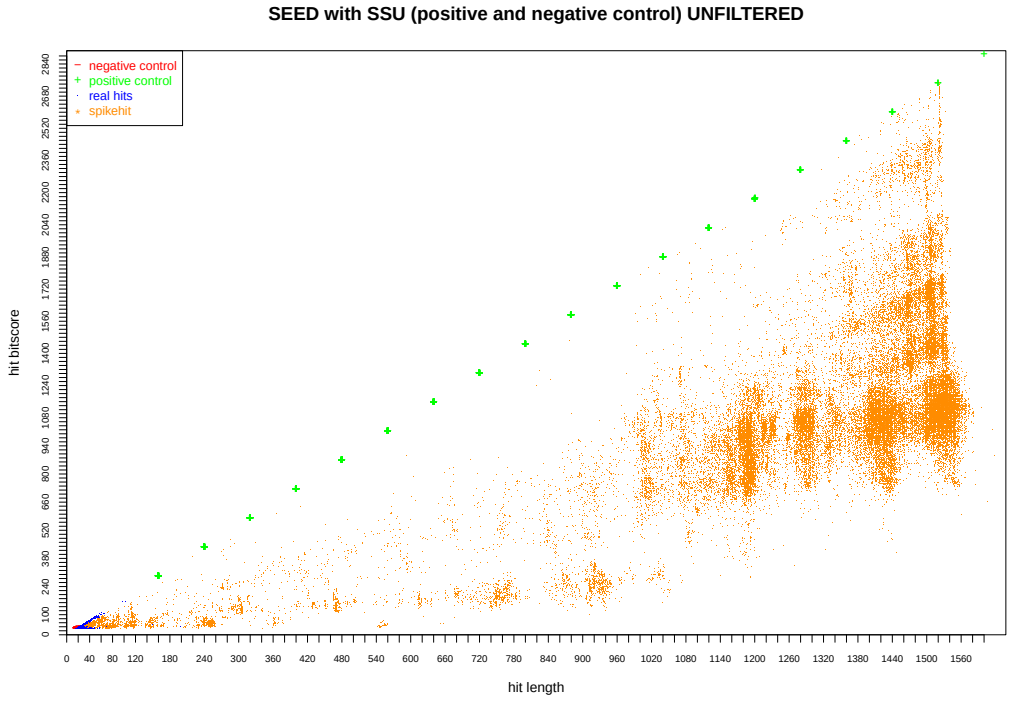

Abbildung 2.3 – Punktdiagramm mit allen BLAST Treffern erstellt mit dem **SILVAVecImporter**. Auf der x-Achse ist die Alignmentlänge und auf der y-Achse der BLAST *bitscore* aufgetragen. Die Positivkontrolle ist **grün**, die Negativkontrolle **rot**, die *spike*-Sequenz **orange** und die Treffer des SILVA SEED mit den Vektoren **blau** dargestellt.

einen Teil der Sequenz anzufügen um ein besseres Alignment zu erzielen. Es muss unbedingt berücksichtigt werden, dass durch die Erhöhung der Länge des Vektors auch Alignments auftreten können, die nicht mehr der wirklichen Länge des ursprünglichen Vektors entsprechen. Derartig fälschliche Alignments werden durch die Länge herausgefiltert. An einem Beispiel verdeutlicht: Ein Vektor der 200 Basen lang ist, wird durch die volle Zyklisierung 400 Basen. Wenn nun ein Alignment auftritt, welches abzüglich der Gaps länger als die ursprünglichen 200 Basen ist, wird es als ungültig herausgefiltert.

Evaluation der Zyklisierung Das Python Skript ″fastaMakeCyclic.py″kann FASTA Dateien auf ihre Zyklisierung testen, die Zyklisierung ausführen oder die Zyklisierung erkennen und entfernen. Es werden zwei Versuche durchgeführt, die aufzeigen sollen wie sich die Zyklisierung auf ein Alignment auswirkt. Un-

terschiedliche Sequenzen werden gegen die mit dem *SILVAVecImporter* erstellte Vektordatenbank aligniert. Dabei wird eine zyklisierte und eine unzyklisierte Vektordatenbank verwendet, anschließend werden die Ergebnisse grafisch miteinander verglichen. Bei dem ersten Alignment (Anhang 4.4, 4.5 und 4.6) wurden acht als vektorkontaminiert markierte Sequenzen aus der SILVA102 Datenbank verwendet. Die nicht zyklisierte Datenbank hat 3,113 BLAST Treffer erreicht, die zyklisierte Datenbank 5,831. Im direkten Vergleich ist der Mittelwert und der Maximalwert des *bitscores* bei der zyklisierten Vektordatenbank höher, der *bitscore* Mittelwert wurde um fünf erhöht, der Maximalwert circa 40. Im darauf folgendem zweiten Alignment (Anhang 4.7, 4.8 und 4.9) wurde ein *E.Coli* rRNA Klonierungsvektor getestet. Auch hier hat sich die Trefferanzahl von 1,829 bei der nicht zyklisierten auf 3,628 bei der zyklisierten Datenbank erhöht. Bei dem Klonierungsvektor hat sich der *bitscore* Mittelwert kaum erhöht, der Maximalwert aber ist um etwa 600 gestiegen. Anhand der erstellten Graphen wird gezeigt, dass sich die Zyklisierung von Vektorsequenzen positiv auf das Alignment mit BLAST auswirken kann. Die Erhöhung der Trefferanzahl ist klar mit der Verdopplung der Sequenzlänge zu erklären, daher wurde die Trefferanzahl auch nahezu verdoppelt. Der Vorteil der Zyklisierung variiert bei unterschiedlichen Sequenzen, was zu erwarten war. Die Verwendung einer zyklisierten Datenbank zur Erkennung von Vektorkontaminierten Sequenzen in der SILVA Pipeline wird als sinnvoll angesehen.

2.5.4 Änderungen im Prozess

Der *SILVAVecImporter* ist vollendet und der Prozess der Erstellung und Validierung einer Vektordatenbank damit einfacher gestaltet. Die Erkennung der Vektorkontamination in der SILVA Pipeline ist wie in Punkt 2.5.1 auf Seite 23 beschrieben teilweise fehlerhaft und bedarf noch einiger Optimierungen. Eine Grundidee ist, die Region, in der Vektoren gesucht werden, zu beschränken. Vektorreste sind nur am Anfang oder am Ende einer Sequenz zu finden, daher sollen auch nur diese Bereiche kontrolliert werden. Wenn *SINA* (1.4.3) nur einen Teil der Sequenz alignieren kann, entstehen nicht alignierte Bereiche vor (head) und hinter (tail) dem Alignment. Nur diese Bereiche (*cutoffhead* und *cutofftail*) werden nun auf Vektorkontamination geprüft. Eine weitere grundlegende Änderung wurde nach langer Überlegung zusammen mit meinem Betreuer Prof. Dr. F. O. Glöckner beschlossen. Die Vektorkontamination soll kein Kriterium mehr sein um die betroffenen Sequenzen aus der SILVA Datenbank auszuschließen, mit Ausnahme von Sonderfällen bei denen:

1. der Vektoranteil größer als die Hälfte der gesamten Sequenz ist,

2. sich der Vektoranteil an einer bestimmten Position im Alignment befindet, die in dem Aligment der *E. coli* Sequenz liegt (Sonderfall! circa: 0.01%).

Bei der Auswertung der *BLAST+* Ergebnisse können mehrere Vektoren an derselben Position der Sequenz aligniert werden, wodurch mehrere Treffer auf der gleichen Region mitgezählt werden.

Um die Abdeckung der Vektorkontamination auf einer Sequenz korrekt zu berechnen wurde eine neue Klasse *SeqCoverage* (Anhang 4.10) in *C++* entwickelt. Jede Start- und Stopp- Position des Alignments auf der Sequenz wird an *SeqCoverage* übergeben und am Ende der prozentuale Anteil der Kontamination berechnet. *seqCoverage* beseitigt das Problem der mehrfachen Zählung von Alignments, die sich in derselben Region befinden.

Die *BLAST+* Ergebnisse werden vor der Registrierung mit *SeqCoverage* auf ihre *evalue* und *pident* Werte geprüft. Der *evalue* Wert wurde vom *NCBI* für *BLAST* entwickelt und soll die Wahrscheinlichkeit eines zufälligen Treffers wiedergeben. Ein *evalue* von 10 sagt aus, das ein Alignment zu 10% aus zufälliger Übereinstimmung entstanden sein könnte, ohne eine echt Homologie zu bezeugen. Erfahrungsgemäß ist 0.05 ein guter *evalue* Wert um *BLAST* Ergebnisse zu filtern ohne wirklich relevante Treffer zu übersehen.

Der *pident* Wert steht für die prozentuale Übereinstimmung des gesamten Alignments. Der *pident* Wert ist gerade bei der Erkennung von Vektoren sehr wichtig, da hier immer eine hohe Übereinstimmung mit dem Vektor vorhanden sein sollte. Alle BLAST Ergebnisse die einen *pident* Wert kleiner als 90% haben werden bei der Vektorerkennung nicht berücksichtigt. Die Filterung der Alignmentergebnisse nach den *evalue* und *pident* Werten ist erforderlich um eine Vektorkontamination korrekt zu erkennen.

Wenn die Vektordatenbank zyklisiert vorliegt (siehe 2.5.3), müssen Alignments, die länger als die ursprüngliche unzyklisierte Sequenz sind, herausgefiltert werden. Die Ergebnisse von BLAST+ zur werden im CSV Format gespeichert. Ein Problem, das bereits bei der Implementierung des *SILVAVecImporters* auftrat, ist die fehlende Möglichkeit beim CSV Format die Längen der Sequenzen im BLAST+ Ergebnis mit ausgeben zu lassen. Die vom *SILVAVecImporter* erstellte Vektordatenbank enthält den Längenparameter in der Kopfzeile und muss entsprechend herausgeschnitten werden, die Länge der *query* Sequenzen ist aus der SILVA Datenbank abrufbar. Es wurden die Fehler in der Erkennung der Vektorkontamination der SILVA Sequenzen behoben und die automatisierte Aktualisierung der Vektordatenbank implementiert. Die aktualisierte Vektordatenbank wird für die Pipeline verwendet. Die Auswirkungen der Änderungen werden ab Seite 39 gezeigt.

2.6 Repetitive Nukleotide

Repetitive Anteile können wie in Kapitel 1.5.3 auf Seite 12 beschrieben durch Sequenzierungsfehler entstehen. Ein gewisser Anteil an Homopolymeren kommt aber in sehr vielen Sequenzen vor. Ab wann eine Sequenz als repetitiv gilt muss für die SILVA Pipeline klar definiert sein, um einen entsprechenden Grenzwert für die Repetitivität setzen zu können.

2.6.1 Prozess in SILVA102

Jede Sequenz in der SILVA Pipeline wird auf ihren repetitiven Anteil überprüft. Eine Sequenz gilt als repetitiv, wenn sich dieselbe Base mehr als vier mal hintereinander wiederholt, jede weitere Base wird gezählt und mit in den prozentualen Gesamtanteil eingerechnet.

Beispielsequenz 10Bp:	**A G G G G G G U G A**
Repetitiver Anteil:	**A G G G G G G U G A**
Ergebnis:	**20% repetitiv**

Tabelle 2.1 – Ein Beispiel der Berechnung des repetitiven Anteils einer Sequenz in der SILVA Pipeline

Der Gesamtanteil an repetitiven Nukleotiden einer Sequenz darf in der derzeitigen SILVA Pipeline nicht über 2% liegen. Dieser Wert wurde von Prof. Dr. F. O. Glöckner durch die Erwartung eines bestimmten Ausmusterungsanteils festgelegt, entsprechend hat sich der Grenzwert von 2% als passend herausgestellt.

2.6.2 Repetitive Nukleotide im SILVA SEED

Eine Betrachtung der repetitiven Nukleotide im SILVA SEED könnte zeigen, wie viele repetitive Nukleotide in hochqualitativen Sequenzen enthalten sind. Die zusätzliche Darstellung der repetitiven Nukleotiden im Zusammenhang mit der taxonomischen Zuordnung des SILVA SEED könnte eine flexible Anwendung von Grenzwerten ermöglichen, die abhängig von der Taxonomie angewendet werden. Durch die Berücksichtigung von Organismen, die durch natürliche Ursachen einen erhöhten Anteil an repetitiven Nukleotiden haben, würden bei der Kontrolle auf repetitive Nukleotide weniger falsch positive Sequenzen auftreten. Der Anteil an repetitiven Nukleotiden im SILVA SEED soll grafisch dargestellt werden, die Sequenzen werden anhand der taxonomischen Zuordnung aufgeteilt. Es soll gezeigt werden wie sinnvoll flexible Grenzwerte für repetitive

Nukleotide anhand der Taxonomie sind. Durch die Verwendung der SILVA Pipeline werden die SEED Sequenzen in die Datenbank importiert und der Anteil an repetitiven Nukleotiden jeder SEED Sequenz berechnet. Durch die anschließende Visualisierung der repetitiven Eigenschaften und der Taxonomie des SEED kann die Verteilung betrachtet werden. Die Verwendung der gesamten Taxa würde einen unübersichtlichen Graphen ergeben, daher muss eine geringe taxonomische Pfadtiefe betrachtet werden, wie zum Beispiel Reich oder Stamm. Die Taxonomie ist nach folgender Hierarchie gegliedert: Die taxonomische

01. Reich	07. Familie
02. Abteilung / Stamm	08. Unterfamilie
03. Unterstamm	09. Gattung
04. Klasse	10. Art
05. Ordnung	11. Unterart
06. Unterordnung	

Klassifikation in der SILVA Datenbank liegt durch Semikolons getrennt in einer Zeichenkette vor. Nicht alle Organismen wurden vollständig bis zur Unterart durchklassifiziert, daher kommen Sequenzen vor, deren Taxonomie nur durch das Reich bestimmt ist.

Eine lange Recherche nach der richtigen Darstellungsmöglichkeit führte zum *Violinplot* [HN98], er kann die Verteilung und die Varianz in einem übersichtlichen Graphen darstellen. Zur vollständigen Interpretationsmöglichkeit wird noch eine zusätzliche Ordinatenachse (vertikal) auf der rechten Seite hinzugefügt, welche die Sequenzanzahl wiedergibt. Die Ergebnisse der Darstellung der repetitiven Nukleotide im SILVA SEED anhand der Taxonomie werden im Anhang gezeigt. Der erste Graph 4.13 zeigt die Verteilung der repetitiven Anteile in der ersten Taxa *Reich*. Darauf folgen dieselben Graphen jeweils in der zweiten Taxa *Stamm* 4.14 und dritten Taxa *Unterstamm* 4.15 und 4.16.

Interpretation der *violin plots* Der Violinplot 4.13 zeigt, dass das SILVA SEED hauptsächlich *Bacteria* Sequenzen enthält (>50,000) und nur wenige *Archaea* (<3000). Bei Betrachtung der Violinplots in Bezug auf die repetitiven Nukleotide wird festgestellt, dass der Hauptanteil der Sequenzen einen repetitiven Anteil zwischen null und einem Prozent hat. Es kommen einzelne Sequenzen vor, die einen sehr viel höheren Anteil von bis zu neun Prozent an repetitiven Anteilen besitzen. Sequenzen wie zum Beispiel die *Eukarya;Metazoa;Arthropoda;* auf Grafik 4.16 oder *Bacteria;Cyanobacteria;Chloroplast;* auf Grafik 4.15 erreichen die höchsten Anteile.

Der Umstand, dass der Anteil der repetitiven Sequenzen auch abhängig von der Spezies sein kann, führt zu der Folgerung, dass die verwendeten Grenzwerte in der SILVA Pipeline abhängig von der Taxonomie gemacht werden sollten. Durch diese Vorgehensweise könnte die natürliche Varianz bei repetitiven Anteilen von rRNA-Sequenzen anhand der Spezies berücksichtigt werden. Um die Grenzwerte bei repetitiven Sequenzen für unterschiedliche Taxonomien zu bestimmen, sollte, wie bereits bei den Violinplots, das SILVA SEED verwendet werden. Die Implementierung der von der Taxonomie abhängigen Grenzwerte für repetitive Sequenzen in der SILVA Pipeline wird nach der Interpretation der Grafiken als sinnvoll angesehen.

2.6.3 Änderungen im Prozess

Ein Prozess in der SILVA Pipeline, der automatisch anhand des SILVA SEED eine Grenzwerttabelle für repetitive Anteile erstellt, würde die flexible Anwendung von Grenzwerten ermöglichen. Die Grenzwerte sollte dem Maximalwert der jeweiligen taxonomischen Zuordnung im SILVA SEED entsprechen. Die Erstellung einer solchen Tabelle kann mit wenigen Datenbank Befehlen getätigt werden, nur die gesamte Implementierung der Automatisierung nimmt am meisten Zeit in Anspruch. Leider ist für die weitere Planung und Implementation des Prozesses innerhalb dieser Diplomarbeit nicht genügend Zeit vorhanden. Folglich wird die Weiterentwicklung der Qualitätskontrolle für repetitive Sequenzen vorzeitig beendet und zu der Evaluierung übergegangen.

3 Ergebnis & Diskussion

Nach der Implementierung einiger Optimierungen in der SILVA Pipeline sollen nun die Auswirkungen gezeigt werden. Grafik 3.1 auf Seite 40 zeigt ein Ablaufdiagramm der optimierten SILVA Pipeline. Die Zahlen rechts neben dem Ablaufdiagramm zeigen die Anzahl an Sequenzen aus der bisherigen Datenbank, bei der Betrachtung sollte beachtet werden, dass das Alignment vor die Qualitätskontrolle verschoben wurde (2.3) und die Position der Zahlen neben dem Graphen entsprechend mitverschoben wurde. In den roten Feldern stehen die herausgefilterten Sequenzen und in den grünen Feldern die nicht herausgefilterten Sequenzen. Anhang 4.17, 4.18 und 4.19 stellen Vergleiche zwischen den Qualitätsmerkmalen der Datenbanken dar, die auf der bisherigen und der optimierten Pipeline basieren. Die Sequenzdaten beider Datenbanken stammen aus der EMBL Datenbank in der Version 102 und es werden alle Sequenzen in den Graphen dargestellt, deren entsprechender Anteil größer null ist.

Mehrdeutige Basen (4.17): die Anzahl an Sequenzen mit mehrdeutigen Basen ist in der optimierten Version von 157,115 auf 130,134 gesunken. Der Durchschnittswert aller Sequenzen mit einem Anteil größer null hat sich von 0.9% auf 0.8% verringert und damit kaum verändert. In der vorherigen Version gab einige Sequenzen mit einem Anteil an mehrdeutigen Basen größer 85% die in der optimierten Version nicht mehr vorhanden sind.

Vektorkontamination (4.18): die Sequenzen, bei denen eine Vektorkontamination erkannt wurde, haben sich von 1,045,151 auf lediglich 2,337 verringert. Der Durchschnittswert der Vektorkontamination bei Sequenzen, die Vektoranteile besitzen, ist von 2.2% auf 4.1% gestiegen. Der Maximalwert bei der bisherigen Datenbank zeigt 105.4% was auf die fehlerhafte Berechnung zurückzuführen ist.

Repetitive Nukleotide (4.19): die Anzahl an Sequenzen mit repetitiven Nukleotiden hat sich von 1,335,367 auf 1,145,696 verringert. Der durchschnittliche Anteil an repetitiven Nukleotiden bei Sequenzen mit einem Anteil größer Null beträgt bei der bisherigen 0.5% und in der optimierten Version 0.4%.

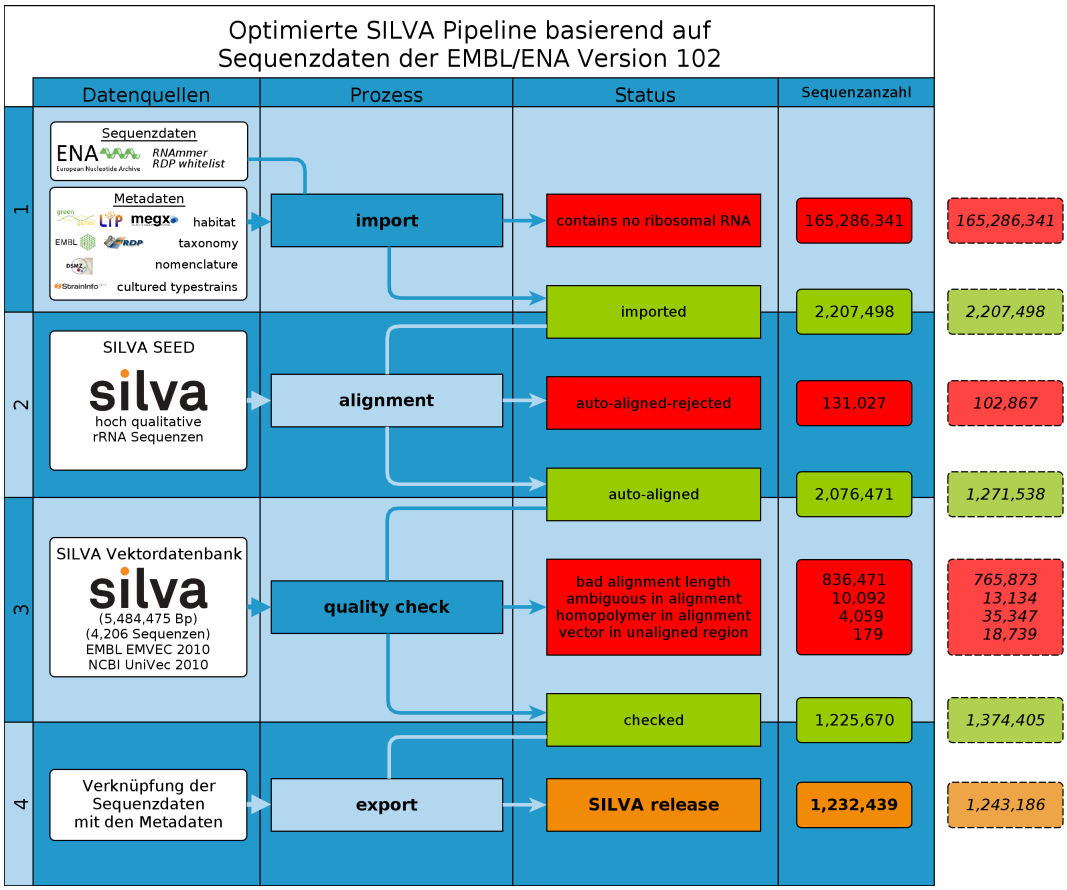

Abbildung 3.1 – Die Grafik zeigt den Ablauf der optimierten SILVA Pipeline am Ende dieser Arbeit.

3.1 Kontrolle der Grenzwerte

Die Grenzwerte für die Anteile an mehrdeutigen Basen und repetitiven Nukleotiden in der SILVA Pipeline könnten durch die vorgenommenen Optimierungen eine Neubestimmung erfordern. Die Balkendiagramme in Anhang 4.20 und 4.21 veranschaulichen ob dies erforderlich ist.

Für die Vektorkontrolle wurde der Grenzwert für die optimierte Pipeline auf 50% festgelegt. Vektorkontaminierte Sequenzen werden herausgefiltert, wenn der Anteil mehr als die Hälfte der gesamten Sequenz beträgt.

Im Balkendiagramm für mehrdeutige Basen (4.20) kann klar gesehen werden, dass sich lediglich die Gesamtanzahl an Sequenzen mit mehrdeutigen Basen verringert hat, die Verteilung aber gleich geblieben ist. Eine Anpassung des

Grenzwertes für mehrdeutige Basen wird daher als nicht erforderlich angesehen.

Die Neubestimmung des Grenzwertes für repetitive Nukleotide stellt sich als ausgesprochen schwierig dar. Die Implementation von flexiblen Grenzwerten, die abhängig von der Taxonomie sind, konnte in dem Umfang dieser Arbeit nicht mehr durchgeführt werden. Das Balkendiagramm 4.21 zeigt eine Verringerung der Anzahl der Sequenzen mit repetitiven Anteilen und eine große Veränderung der Verteilung. Wurden bei einem vorherigen Grenzwert von 2% fast 40,000 Sequenzen herausgefiltert, wären es jetzt ungefähr noch 4,000 Sequenzen. Bei einem Grenzwert von 1.5% fällt im Gegensatz zu einem Grenzwert von 1% die Sequenzanzahl stark ab. Hinzu kommt, dass bei einem Grenzwert von 1% ungefähr dieselbe Sequenzanzahl umfasst wird wie in der bisherigen Qualitätskontrolle. Daher sollte der neue Grenzwert zwischen 1% und 1.5% angesetzt werden.

3.2 Fazit und Aussichten

Mehrdeutige Basen: in diesem Modul wurden keine Änderungen getätigt. Die Verringerung der Sequenzen mit mehrdeutigen Basen beruht daher allein auf der Beschränkung der Kontrolle auf den alignierten Bereich. Die Häufigkeit des Auftretens von mehrdeutigen Basen innerhalb der von SINA erkannten rRNA Regionen ist geringer. Eine Möglichkeit zur weiteren Optimierung der Kontrolle wäre, die Zerstreuung zu beurteilen. Wenn sich der Anteil an mehrdeutigen Basen auf einen einzelnen Bereich beschränkt, könnte dies positiver bewertet werden.

Vektorkontamination: die Beschränkung der Vektorkontrolle auf die von SINA nicht alignierten Enden der Sequenzen hat zu einer drastischen Verringerung der Vektorkontamination geführt. Insbesondere die fehlerhafte Berechnung der Vektoranteile der SILVA Sequenzen hat zu fälschlich hohen Werten geführt. Ein gewisser Anteil an vektorkontaminierten Sequenzen könnte in der optimierten Version mit aligniert worden sein, wobei der größere Teil an ehemals als vektorkontamiert erkannte Sequenzen wohl falsch positive waren. Die Vektorkontrolle hat durch die Beschränkung auf die nicht alignierten Enden kaum noch Aussagekraft über die Qualität einer Sequenz und wurde entsprechend aus der Formel zur Berechnung der Qualität einer Sequenz entfernt.

Repetitive Nukleotide: in diesem Modul wurde außer der Beschränkung der Kontrolle auf den alignierten Bereich aus Zeitgründen keine weitere Optimierung implementiert. Die Abnahme der Sequenzen bei denen repetitive Nukleotide

erkannt wurde kann als alleine mit der Beschränkung der Kontrolle auf den von SINA alignierten Bereich begründet werden. Als nächstes sollte mit der Implementation der flexiblen Grenzwerte anhand des SILVA SEED begonnen werden. Dies erfordert eine automatische Importierung des SILVA SEED in der Pipeline und die anschließende Berechnung der repetitiven Anteile.

Schlusswort: die Qualitätskontrolle in der SILVA Pipeline soll die Qualität der rRNA Sequenzen beurteilen. Durch die Anwendung der Qualitätskontrolle auf die alignierten Bereiche wurde erreicht, dass ausschliesslich die Qualität des rRNA Anteils einer Sequenz überprüft wird und nicht wie bisher die der gesamten Sequenz, deren rRNA Anteil sehr gering sein könnte. Durch diese Änderung wurde die Aussagekraft des berechneten Qualitätswertes einer Sequenz verstärkt. Die fehlerhafte Erkennung der Vektorkontamination von Sequenzen wurde korrigiert und nur Extremfälle bei denen der Vektoranteil mehr als die Hälfte einer Sequenz ausmacht werden herausgefiltert. Insgesamt sind die angewandten Optimierungen ein Schritt in die richtige Richtung, es sind aber noch weitere Optimierungsmöglichkeiten vorhanden. Ein möglicher nächster Schritt wäre es, einen Teil der Qualitätskontrolle direkt während des Alignments durchzuführen um bestimmte Fehlerspektren besser beurteilen zu können.

Literaturverzeichnis

[ABH+05] Alberts, Bruce ; Bray, Dennis ; Hopkin, Karen ; Johnson, Alexander ; Lewis, Julian ; Raff, Martin ; Roberts, Keith ; Walter, Peter: *Lehrbuch der molekularen Zellbiologie*. 3. Aufl. Weinheim : Wiley-VCH, 2005. – ISBN 9783527311606

[ACF+05] Ashelford, Kevin E. ; Chuzhanova, Nadia A. ; Fry, John C. ; Jones, Antonia J. ; Weightman, Andrew J.: At least 1 in 20 16S rRNA sequence records currently held in public repositories is estimated to contain substantial anomalies. In: *Applied and Environmental Microbiology* 71 (2005), Dezember, Nr. 12, 7724–7736. http://dx.doi.org/10.1128/AEM.71.12.7724-7736.2005. – DOI 10.1128/AEM.71.12.7724–7736.2005. – ISSN 0099–2240. – PMID: 16332745

[Alt90] Altschul, S: Basic Local Alignment Search Tool. In: *Journal of Molecular Biology* 215 (1990), Nr. 3, 403–410. http://dx.doi.org/10.1006/jmbi.1990.9999. – DOI 10.1006/jmbi.1990.9999. – ISSN 00222836

[BN74] Biochemical Nomenclature, JUPAC—IUB C.: *ABBREVIATIONS AND SYMBOLS FOR NUCLEIC ACIDS, POLYNUCLEOTIDES AND THEIR CONSTITUENTS*. http://www.iupac.org/publications/pac/1974/pdf/4003x0277.pdf. Version: 1974

[dsm] *DSMZ - Deutsche Sammlung von Mikroorganismen und Zellkulturen GmbH German Collection of Microorganisms and Cell Cultures*. http://www.dsmz.de/. http://www.dsmz.de/

[DVMS05] Dawyndt, P. ; Vancanneyt, M. ; Meyer, H. D. ; Swings, J.: Knowledge accumulation and resolution of data inconsistencies during the integration of microbial information sources. In: *IEEE Transactions on Knowledge and Data Engineering* 17 (2005), Nr. 8, 1111–1126. http://dx.doi.org/10.1109/TKDE.2005.131. – DOI 10.1109/TKDE.2005.131. – ISSN 1041–4347

[EBI10] EBI: *Nucleotide Analysis - Checking for vector contamination , an introduction / 2can Support Portal / EBI*. http://www.ebi.ac.uk/2can/tutorials/nucleotide/vector.html. `http://www.ebi.ac.uk/2can/tutorials/nucleotide/vector.html`. Version: 2010

[emb] *EMBL Heidelberg - The European Molecular Biology Laboratory*. http://www.embl.de/. `http://www.embl.de/`

[Glo] Gloeckner, Frank O.: *Phylogenie und in situ-Identifizierung von Prokaryonten in limnischen und marinen Ökosystemen*. 1998, Technische Universität München, Diss.

[gre] *greengenes.lbl.gov - Aligned 16S rDNA data and tools*. http://greengenes.lbl.gov. `http://greengenes.lbl.gov`

[HHM+07] Huse, Susan M. ; Huber, Julie A. ; Morrison, Hilary G. ; Sogin, Mitchell L. ; Welch, David: Accuracy and quality of massively parallel DNA pyrosequencing. In: *Genome Biology* 8 (2007), Nr. 7, R143. `http://dx.doi.org/10.1186/gb-2007-8-7-r143`. – DOI 10.1186/gb–2007–8–7–r143. – ISSN 14656906

[HN98] Hintze, Jerry L. ; Nelson, Ray D.: Violin Plots: A Box Plot-Density Trace Synergism. In: *The American Statistician* 52 (1998), Nr. 2, 181. `http://dx.doi.org/10.2307/2685478`. – DOI 10.2307/2685478. – ISSN 00031305

[LHR+07] Lagesen, Karin ; Hallin, Peter ; Rødland, Einar A. ; Staerfeldt, Hans-Henrik ; Rognes, Torbjørn ; Ussery, David W.: RNAmmer: consistent and rapid annotation of ribosomal RNA genes. In: *Nucleic Acids Research* 35 (2007), Nr. 9, 3100–3108. `http://dx.doi.org/10.1093/nar/gkm160`. – DOI 10.1093/nar/gkm160. – ISSN 1362–4962. – PMID: 17452365

[LSW04] Ludwig, Wolfgang ; Strunk, Oliver ; Westram, Ralf: ARB: a software environment for sequence data. In: *Nucleic Acids Research* 32 (2004), Nr. 4, 1363–1371. `http://dx.doi.org/10.1093/nar/gkh293`. – DOI 10.1093/nar/gkh293. – ISSN 1362–4962

[MWG] Moreira, Walter ; Warnes, Gregory R. ; Gautier, Laurent: *RPy - A simple and efficient access to R from Python*. http://rpy.sourceforge.net/. `http://rpy.sourceforge.net/`

[NC05] Nelson, David L. ; Cox, Michael M.: *Lehninger Biochemie*. 3. vollst. überarb. und erw. Aufl., 1. korrigierter Nachdr. Berlin [u.a.] : Springer, 2005. – ISBN 9783540418139

[ncb] *NCBI - National Center for Biotechnology Information*. http://www.ncbi.nlm.nih.gov/. http://www.ncbi.nlm.nih.gov/

[NCB09] NCBI: *NCBI: Contamination in Sequence Databases*. http://www.ncbi.nlm.nih.gov/VecScreen/contam.html. Version: Juli 2009

[Oli05] Oliver, James D.: The viable but nonculturable state in bacteria. In: *Journal of Microbiology (Seoul, Korea)* 43 Spec No (2005), Februar, 93–100. http://www.ncbi.nlm.nih.gov/pubmed/15765062. – ISSN 1225–8873. – PMID: 15765062

[PQK+07] Pruesse, E. ; Quast, C. ; Knittel, K. ; Fuchs, B. M. ; Ludwig, W. ; Peplies, J. ; Glockner, F. O.: SILVA: a comprehensive online resource for quality checked and aligned ribosomal RNA sequence data compatible with ARB. In: *Nucleic Acids Research* 35 (2007), Nr. 21, 7188–7196. http://dx.doi.org/10.1093/nar/gkm864. – DOI 10.1093/nar/gkm864. – ISSN 0305–1048

[Pru07] Pruesse, Elmar: *Incremental Approach to Multiple Sequence Alignment using Directed Acyclical Graphs*. MPI Bremen, University Bremen, Diss., Juli 2007

[Qua09] Quast, Christian: *Tool and Database Development for the Phylogenetic Classification and Functional Characterisation of Organisms*. Bremen, Universität Bremen, Diss., September 2009

[rdp] *Ribosomal Database Project*. http://rdp.cme.msu.edu/. http://rdp.cme.msu.edu/

[sge] *Sun Grid Engine*. http://wikis.sun.com/display/GridEngine/Home. http://wikis.sun.com/display/GridEngine/Home

[SO] Sirola, Enrico ; Ong, Chengsoon: *drmaa-python - Project Hosting on Google Code*. http://code.google.com/p/drmaa-python/. http://code.google.com/p/drmaa-python/

[SOW+] Smith, S. ; Overbeek, R. ; Woese, C.R. ; Gilbert, W. ; Gillevet, P.: *The Genetic Data Environment (GDE)*. http://www-bimas.cit.nih.gov/gde_sw.html. http://www-bimas.cit.nih.gov/gde_sw.html

[YLE+10] Yarza, Pablo ; Ludwig, Wolfgang ; Euzéby, Jean ; Amann, Rudolf ; Schleifer, Karl-Heinz ; Glöckner, Frank O. ; Rosselló-Móra, Ramon: Update of the All-Species Living Tree Project based on 16S and 23S rRNA sequence analyses. In: *Systematic and Applied Microbiology* 33 (2010), Nr. 6, 291–299. http://dx.doi.org/10.1016/j.syapm.2010.08.001. – DOI 10.1016/j.syapm.2010.08.001. – ISSN 07232020

[YRP+08] Yarza, Pablo ; Richter, Michael ; Peplies, Jörg ; Euzeby, Jean ; Amann, Rudolf ; Schleifer, Karl-Heinz ; Ludwig, Wolfgang ; Glöckner, Frank O. ; Rosselló-Móra, Ramon: The All-Species Living Tree project: A 16S rRNA-based phylogenetic tree of all sequenced type strains. In: *Systematic and Applied Microbiology* 31 (2008), Nr. 4, 241–250. http://dx.doi.org/10.1016/j.syapm.2008.07.001. – DOI 10.1016/j.syapm.2008.07.001. – ISSN 07232020

Abbildungsverzeichnis

4 Anhang

```
[SUBJECT_FILES]
#Sequenzdaten der Vektoren.
#Müssen im FASTA, BLASTDB oder ARB Format vorliegen!
#Die Dateiendung der BLASTDB Dateien darf nicht mit angegeben werden.
#Die Schlüsselnamen müssen unterschiedlich sein!
#Beispiele:
#EmVec = /megx/inputfile1.fasta
#UniVec = /megx/inputfile2

[SUBJECT_SPIKE_FILES]
#Dateipfade zu FASTA Dateien die als spike-Sequenz verwendet werden.
#die Sequenzdaten werden zum BLAST subject hinzugefügt (zur Vektordatenbank).
#Die angegebenen Sequenzen MÜSSEN gleichlang oder länger sein als der TO Wert
#in der SHRED_AND_GEN Sektion, ansonsten gibt es eine entsprechende Fehlermeldung.
#Beispiel:
U72488 = %(basename)s/seqData/U72488-rRNA-Vector.fasta

[QUERY_FILES]
#PFADE zu Dateien im BLASTDB, FASTA oder ARB Format
#Beispiel:
#SSU = /megx/home/tschweer/SILVADATA/blastdb/SSURef_102_SILVA_NR_99

[BLAST_RESULT_FILES]
#Hier werden die Dateifpade der Ergebnisse von BLAST automatisch eingetragen.
#Die Dateien werden in einem bestimmten Format gespeichert, daher sollten
#hier nur Ergebnisse verwendet werden, die mit diesem Skript erstellt wurden.
SEED2010-12-01 =
/megx/home/tschweer/SILVADATA/SilvaVecImporter/BLASTRESULT_SEED-2010-12-01-002.csv

[SHRED_AND_GEN]
#Hier wird festgelegt wie die Positivkontrolle und die Negativkontrolle aufgeteilt
werden.
FROM = 80
TO = 1600
STEPS = 80

[MAIN]
#Zeige erweiterte Debugging Informationen?
VERBOSE = false
#Füge die Positivkontrolle hinzu?
INSERT_POSITIVE_CONTROL = true
#Füge die Negativkontrolle hinzu?
INSERT_NEGATIVE_CONTROL = true
#Der Dateiname der Vektordatenbank die im CREATEmode erstellt wird (subject)
DATABASEFASTAFILE = VectorDB.fasta
#Arbeitsverzeichnis des Programms, hier werden alle Ergebnisse und temporäre Dateien
erstellt
DATADIRECTORY = /megx/home/tschweer/SILVADATA/SilvaVecImporter
#Dateipfad zur FASTA Datei der Positivkontrolle, kann beliebige Sequenz sein!
#Mehr als eine Sequenz ist wegen der Geschwindigkeit und Dateigröße nicht empfohlen!
POSITIVECONTROLFASTAFILE = %(basename)s/seqData/positiveControl.fasta

[CYCLIZING]
#Zyklisierung der Vektordatenbank?
MAKE_SUBJECT_CYCLIC = true
#Anzahl an Basen die auf beiden Seiten einer Sequenz in entgegengesetzer
#Leserichtung in identischer Abfolge vorkommen müssen, damit die Sequenz
```

Abbildung 4.1 – Konfigurationsdatei des SILVAVecImporters

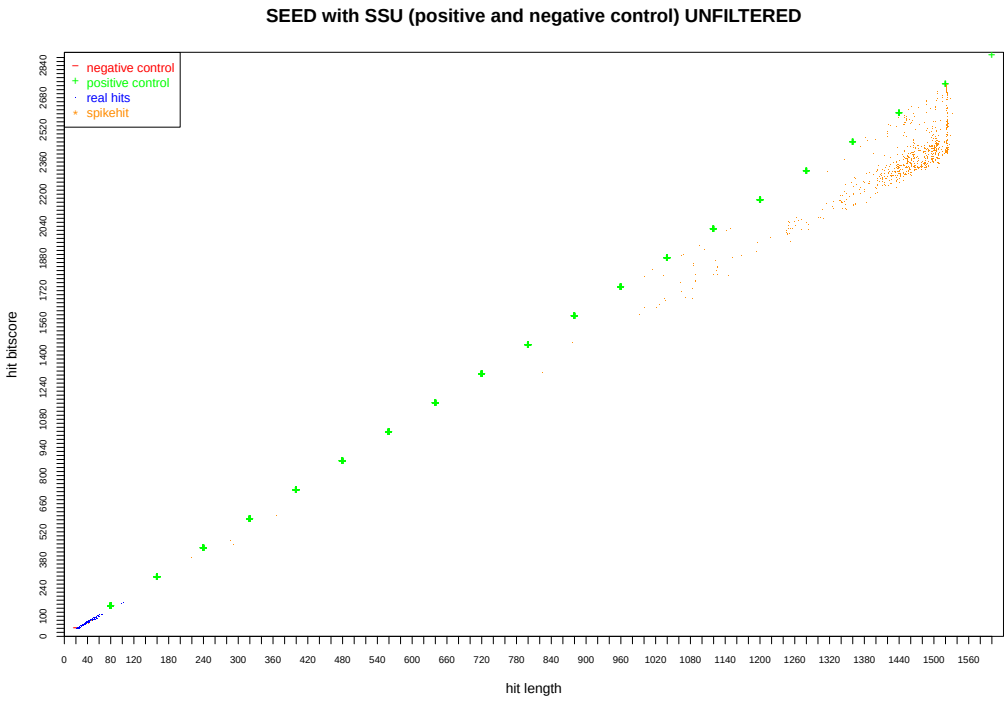

Abbildung 4.2 – Punktdiagramm mit gefilterten BLAST Treffern erstellt mit dem **SILVAVecImporter**. Auf der x-Achse ist die Alignmentlänge und auf der y-Achse der BLAST *bitscore* aufgetragen. Die Positivkontrolle ist **grün**, die Negativkontrolle **rot**, die *spike*-Sequenz **orange** und die Treffer des SILVA SEEDS mit den Vektoren **blau** dargestellt.

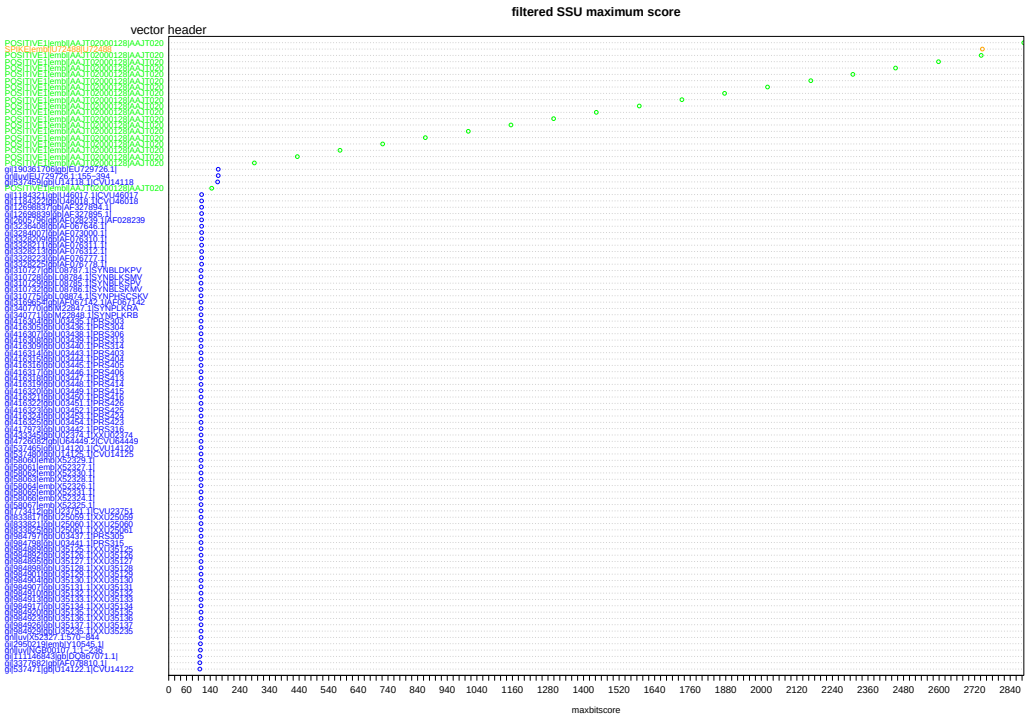

Abbildung 4.3 – Listenansicht der BLAST Treffer absteigend sortiert nach dem BLAST *bitscore* erstellt mit dem **SILVAVecImporter**. Die Positivkontrolle ist **grün**, die Negativkontrolle **rot**, die *spike*-Sequenz **orange** und die Treffer des SILVA SEEDS mit den Vektoren **blau** dargestellt.

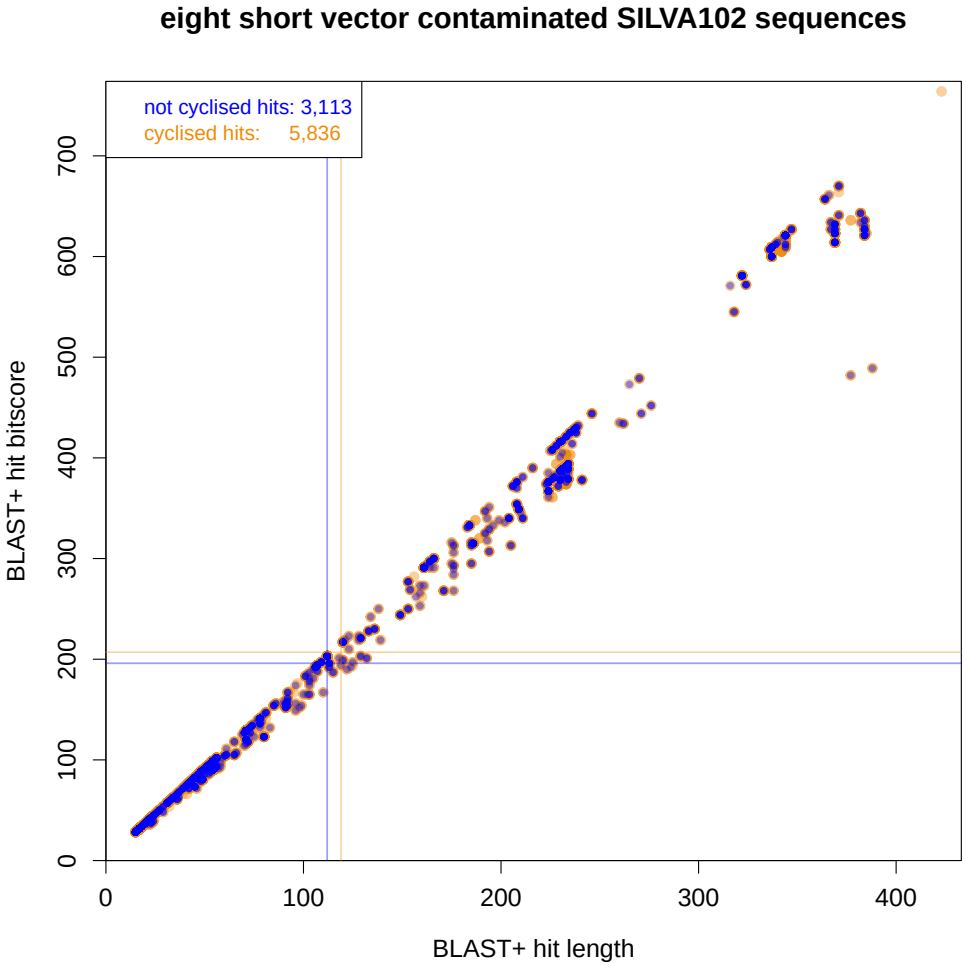

Abbildung 4.4 – Ergebnis des BLAST+ Alignments von acht als vektorkontaminiert markierte Sequenzen aus der SILVA 102 Datenbank gegen die Vektordatenbank. Auf der x-Achse ist die Alignmentlänge und auf der y-Achse der BLAST *bitscore* aufgetragen. Bei den **orangen** Treffern wurde eine zyklisierte Datenbank verwendet, bei den blauen wurde keine Zyklisierung durchgeführt. Die **Linien** markieren den jeweiligen Durchschnittswert.

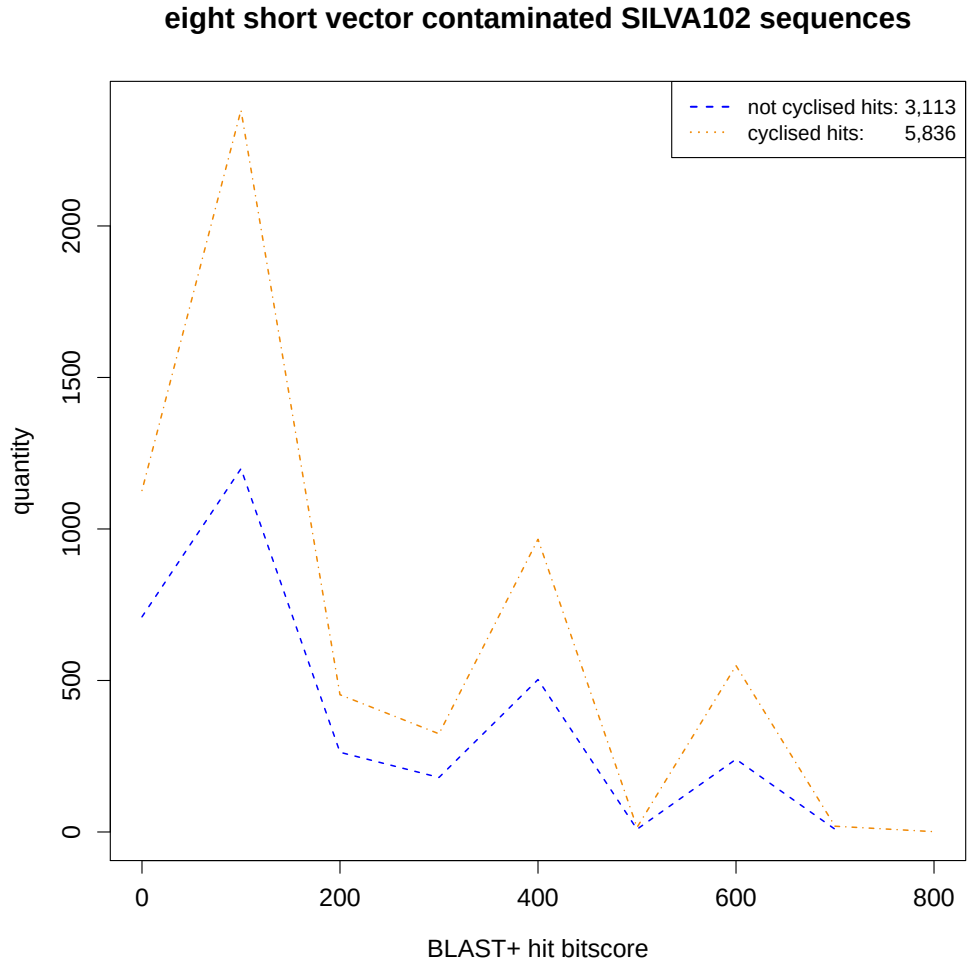

Abbildung 4.5 – Verteilung des *bitscore* des BLAST+ Alignments von acht als vektorkontaminiert markierte Sequenzen aus der SILVA 102 Datenbank gegen die Vektordatenbank. Die *bitscore* Werte wurden zur besseren Darstellung auf hunderter Potenzen gerundet und auf die x-Achse aufgetragen. Auf der y-Achse wird die jeweilige Häufigkeit dargestellt. Bei den **orangen** Treffern wurde eine zyklisierte Datenbank verwendet, bei den blauen wurde keine Zyklisierung durchgeführt.

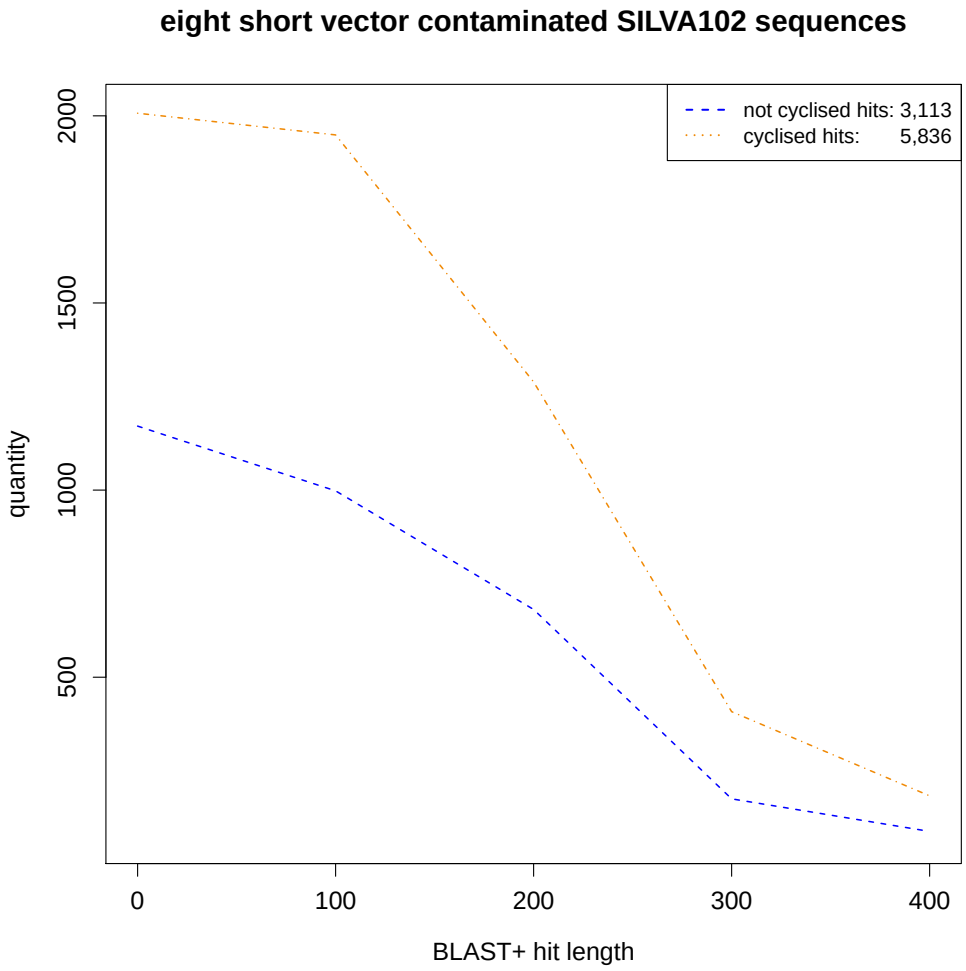

Abbildung 4.6 – Verteilung der Alignmentlänge des BLAST+ Alignments von acht als vektorkontaminiert markierte Sequenzen aus der SILVA 102 Datenbank gegen die Vektordatenbank. Die Alignmentlänge wurde zur besseren Darstellung auf hunderter Potenzen gerundet und auf die x-Achse aufgetragen. Auf der y-Achse wird die jeweilige Häufigkeit dargestellt. Bei den **orangen** Treffern wurde eine zyklisierte Datenbank verwendet, bei den blauen wurde keine Zyklisierung durchgeführt.

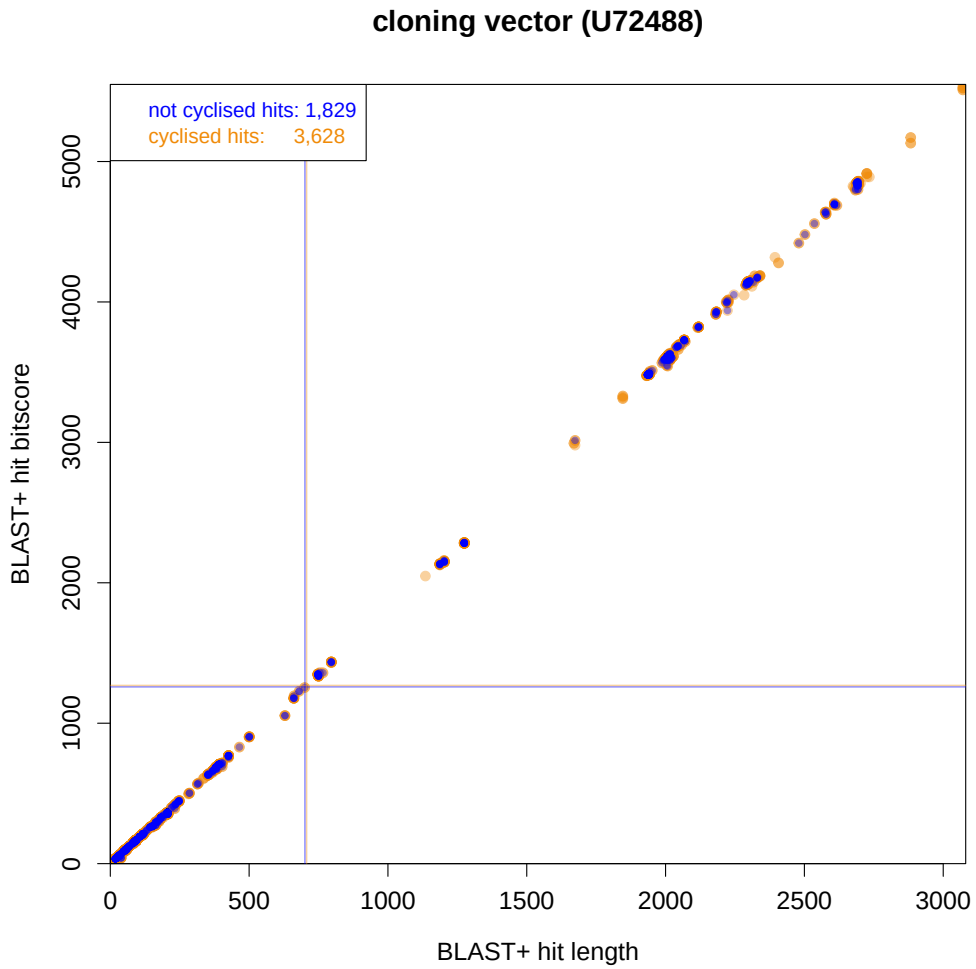

Abbildung 4.7 – Ergebnis des BLAST+ Alignments des *E.Coli* rRNA Klonierungsvektors gegen die Vektordatenbank. Auf der x-Achse ist die Alignmentlänge und auf der y-Achse der BLAST *bitscore* aufgetragen. Bei den **orangen** Treffern wurde eine zyklisierte Datenbank verwendet, bei den blauen wurde keine Zyklisierung durchgeführt. Die **Linien** markieren den jeweiligen Durchschnittswert.

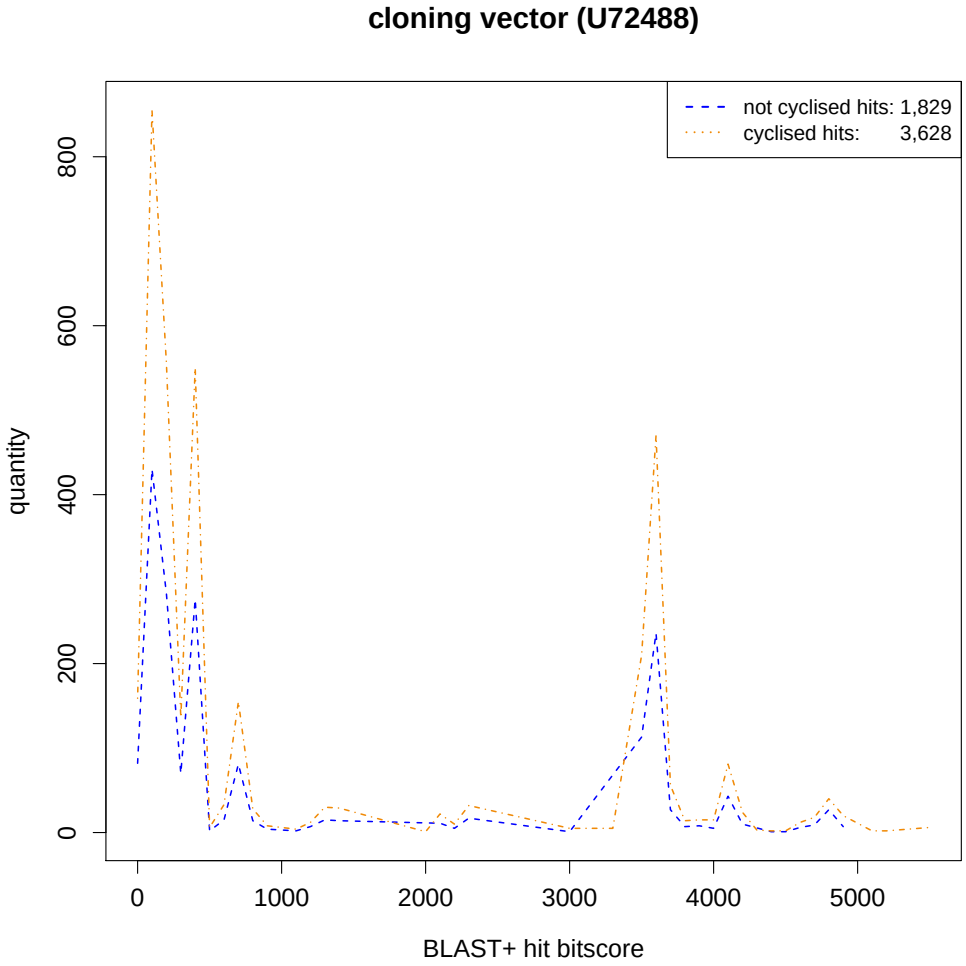

Abbildung 4.8 – Verteilung des *bitscore* des BLAST+ Alignments des *E.Coli* rRNA Klonierungsvektors gegen die Vektordatenbank. Die *bitscore* Werte wurden zur besseren Darstellung auf hunderter Potenzen gerundet und auf die x-Achse aufgetragen. Auf der y-Achse wird die jeweilige Häufigkeit dargestellt. Bei den **orangen** Treffern wurde eine zyklisierte Datenbank verwendet, bei den blauen wurde keine Zyklisierung durchgeführt.

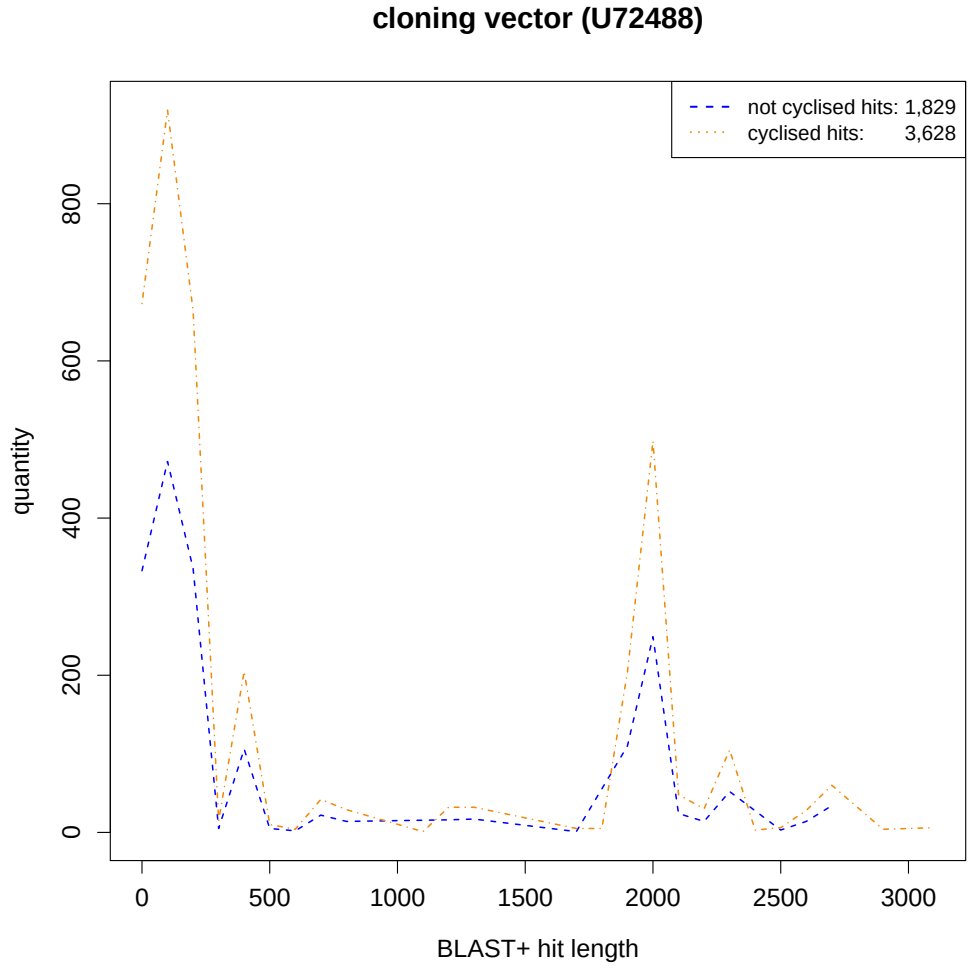

Abbildung 4.9 – Verteilung der Alignmentlänge des BLAST+ Alignments des *E.Coli* rRNA Klonierungsvektors gegen die Vektordatenbank. Die Alignmentlänge wurde zur besseren Darstellung auf hunderter Potenzen gerundet und auf die x-Achse aufgetragen. Auf der y-Achse wird die jeweilige Häufigkeit dargestellt. Bei den **orangen** Treffern wurde eine zyklisierte Datenbank verwendet, bei den blauen wurde keine Zyklisierung durchgeführt.

```
/* SeqCoverage.h
 * This class is for the SequenceCheck to determine the Vectorcoverage
 * written by: Timmy Schweer
 * Date: 08.2010
 */
#ifndef    SEQCOVERAGE_H_
# define   SEQCOVERAGE_H_
// include guard
#include <vector>
#include <iostream>
#include <map>
#include <algorithm>

namespace silva
{
typedef std::pair<unsigned int, unsigned int> t_pair_unint;
typedef std::vector<t_pair_unint> t_vec_pair_unint;

class SeqCoverage
{

  public:
    SeqCoverage(int seqLength, bool debug);
    SeqCoverage(int seqLength);
    t_pair_unint getValue(int which);
    void addValue(int,int);
    void printValues();
    void calculateCoverage();
    double getCoverage();
    unsigned int getCoverquantity();
  private:
    unsigned int seqLength;
    t_vec_pair_unint amount;
    bool debug;
    bool calculated;
    double coverage;
    unsigned int coverquantity;

};
std::ostream& operator<<(std::ostream&, SeqCoverage& _o);
//std::ostream& operator<<(std::ostream&, const silva::Region&);
//weil global GANNZZ GLOBAL

}

#endif


/* SeqCoverage.cpp
 * This class is for the SequenceCheck module to determine the Vectorcoverage
 * written by: Timmy Schweer
 * Date: 08.2010
 */
#include "SeqCoverage.h"
using namespace silva;

SeqCoverage::SeqCoverage(int seqLength, bool debug)
```

Abbildung 4.10 – C++ Klasse SeqCoverage Teil I. Verwendet für die Erkennung der Vektorkontamination

```
{
  this->seqLength=seqLength;
  this->debug=debug;
  this->calculated=false;
  this->coverage=0;
  this->coverquantity=0;
}

SeqCoverage::SeqCoverage(int seqLength)
{
  this->seqLength=seqLength;
  this->debug=true;
  this->calculated=false;
  this->coverage=0;
  this->coverquantity=0;
}

t_pair_unint SeqCoverage::getValue(int which)
{
  return this->amount[which];
}

void SeqCoverage::addValue(int from, int to)
{
  std::pair<unsigned int, unsigned int> tmp (from ,to);
  this->amount.push_back( tmp );
  this->calculated = false;
}

void SeqCoverage::calculateCoverage()
{
  if (this->amount.size()>0 and this->seqLength>0)
  {
    sort(this->amount.begin(), this->amount.end());
    t_vec_pair_unint::iterator it = this->amount.begin();
    t_vec_pair_unint::iterator it_end = this->amount.end();
    unsigned int start=1,stop=1;
    unsigned int covered = 0;
    for (; it!=it_end; ++it) {
      if ((*it).first==0 or (*it).second < (*it).first or (*it).second > this->
seqLength) {
        if (this->debug) std::cout << "SeqCoverage::get_coverage(): invalid pair value:"
            <<"first:" << (*it).first << " second:" << (*it).second << std::endl;
        continue;
      }
      if (start==1 and stop==1) { start=(*it).first; stop=(*it).second; } //start
condition
      else if(stop>=(*it).second) continue; // 1. inclusion do nothing
      else if(stop>=(*it).first) stop=(*it).second; //2. extending, hit is longer (and
starts before last)
      else { covered+=stop-start+1; start=(*it).first; stop=(*it).second;} // 3. gap,
jump to next hit
    }
    if (this->amount[this->amount.size()-1].second==stop) covered+=stop-start+1; //end
conditition
    this->coverquantity = covered;
    this->coverage = ((double)covered / (double)this->seqLength) * 100;
```

Abbildung 4.11 – C++ Klasse SeqCoverage Teil II. Verwendet für die Erkennung der Vektorkontamination

```
    if(this->coverage>100) {
      std::cout << "INVALID COVERAGE FOUND" << std::endl;
      for(unsigned int i=0; i < this->amount.size(); i++) {
        std::cout << i << ". value-start:" << this->amount[i].first << "stop:" << this
->amount[i].second << std::endl;
      }
    }
  }
  else {
    this->coverage = 0.0;
    this->coverquantity = 0;
  }
  this->calculated = true;
}

double SeqCoverage::getCoverage() {
  if (this->calculated==false) this->calculateCoverage();
  return this->coverage;
}

unsigned int SeqCoverage::getCoverquantity() {
  if (this->calculated==false) this->calculateCoverage();
  return this->coverquantity;
}


void SeqCoverage::printValues() {
  t_vec_pair_unint::iterator it = this->amount.begin();
  t_vec_pair_unint::iterator it_end = this->amount.end();
  for (; it!=it_end; ++it) {
    std::cout << "start:" << (*it).first;
    std::cout << "end:" << (*it).second << std::endl;
  }
}

std::ostream& silva::operator<<(std::ostream& _os, SeqCoverage& _o)
{
    _os << "sequence Coverage"
    << "\n pCoverage:\t" << _o.getCoverage()
    ;
    return _os;
}
```

Abbildung 4.12 – C++ Klasse SeqCoverage Teil III. Verwendet für die Erkennung der Vektorkontamination

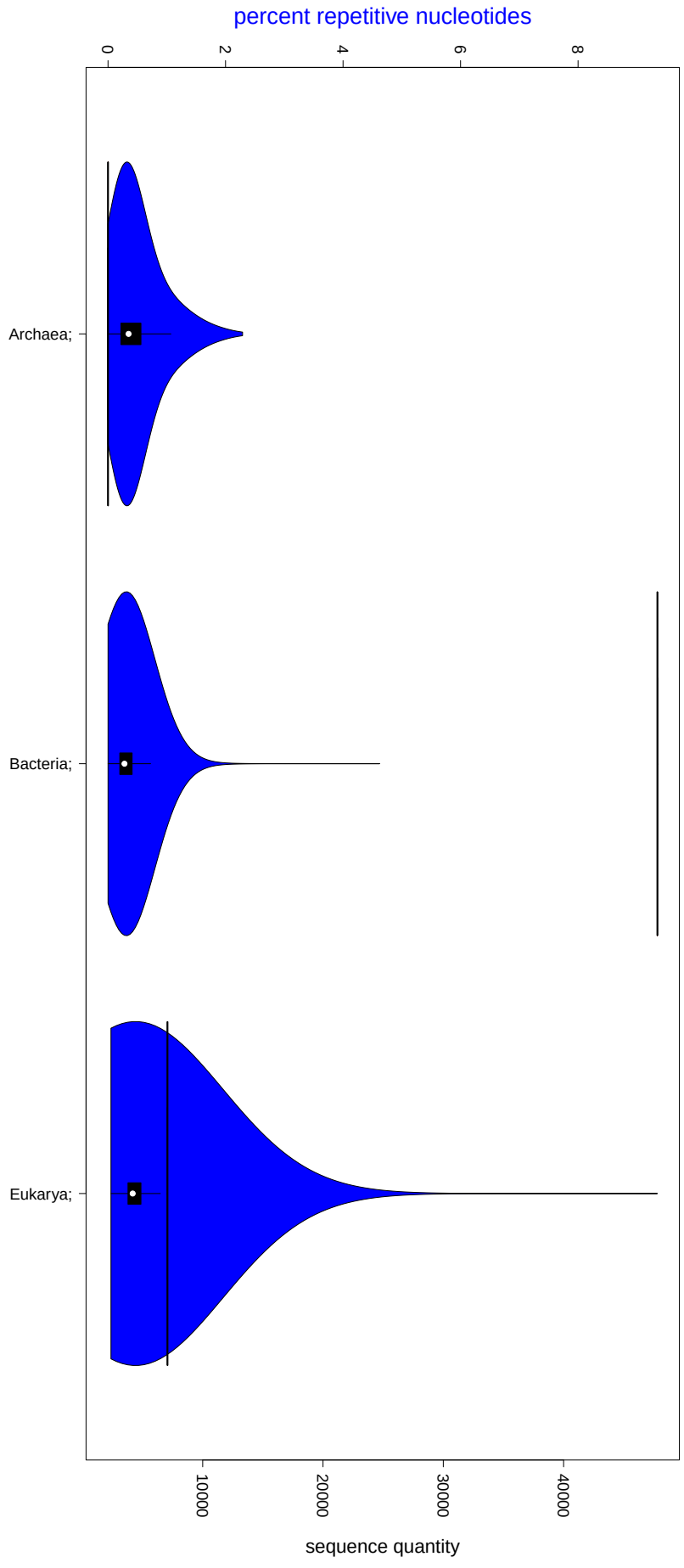

Abbildung 4.13 – Violinplot [HN98] des prozentualen Anteil an repetitiven Nukleotiden im SILVA SEED aufgeteilt nach der taxonomischen Zuordnung in der ersten Taxa *Reich*. Die linke x-Achse zeigt den prozentualen Anteil an repetitiven Nukleotiden und auf der y-Achse die taxonomische Klassifikation. Die Höhe des Violinplots zeigt die Varianz des repetitiven Anteils an, die Breite zeigt die Verteilung. Der breite schwarze Strich in den Violinplots zeigt die Sequenzanzahl auf der rechten x-Achse an. Der weisse Punkt in den blauen Violinplots markiert den Mittelwert an repetitiven Anteilen.

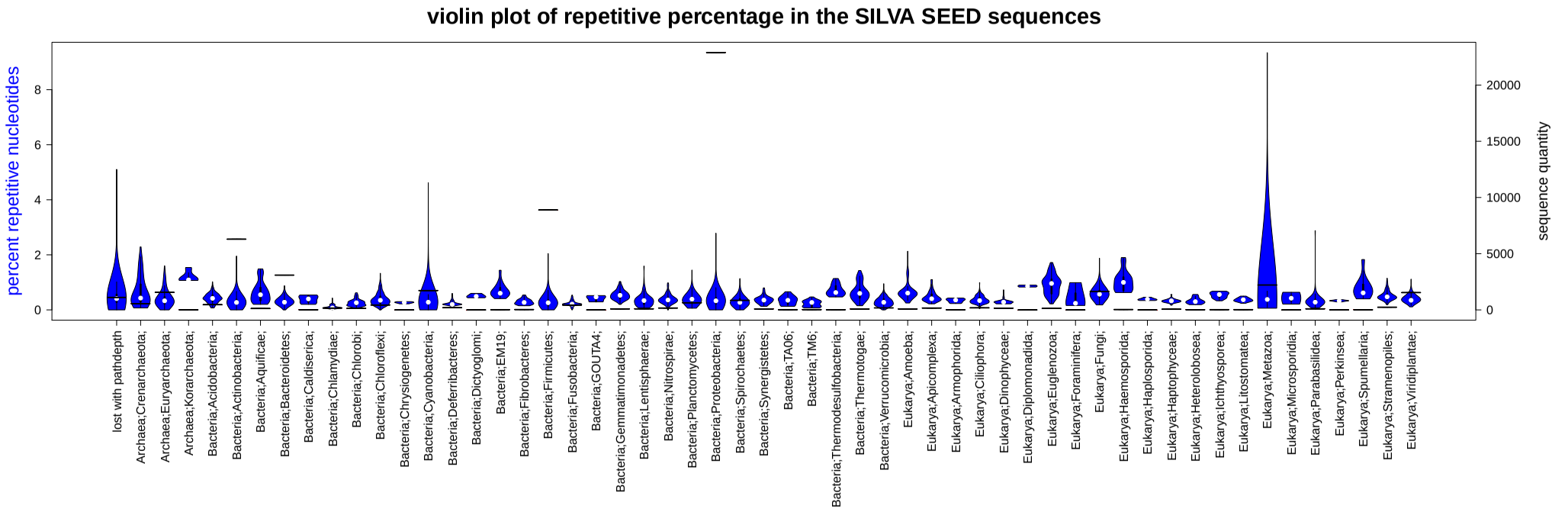

Abbildung 4.14 – Violinplot [HN98] des prozentualen Anteil an repetitiven Nukleotiden im SILVA SEED aufgeteilt nach der taxonomischen Zuordnung in der zweiten Taxa *Stamm*. Die linke x-Achse zeigt den prozentualen Anteil an repetitiven Nukleotiden und auf der y-Achse die taxonomische Klassifikation. Die Höhe des Violinplots zeigt die Varianz des repetitiven Anteils an, die Breite zeigt die Verteilung. Der breite schwarze Strich in den Violinplots zeigt die Sequenzanzahl auf der rechten x-Achse an. Der weisse Punkt in den blauen Violinplots markiert den Mittelwert an repetitiven Anteilen. Der Violinplot mit der Beschriftung ″lost with pathdepth″stellt Sequenzen dar, deren taxonomischer Pfad nicht ausreicht um sie in der Taxa *Stamm* darzustellen.

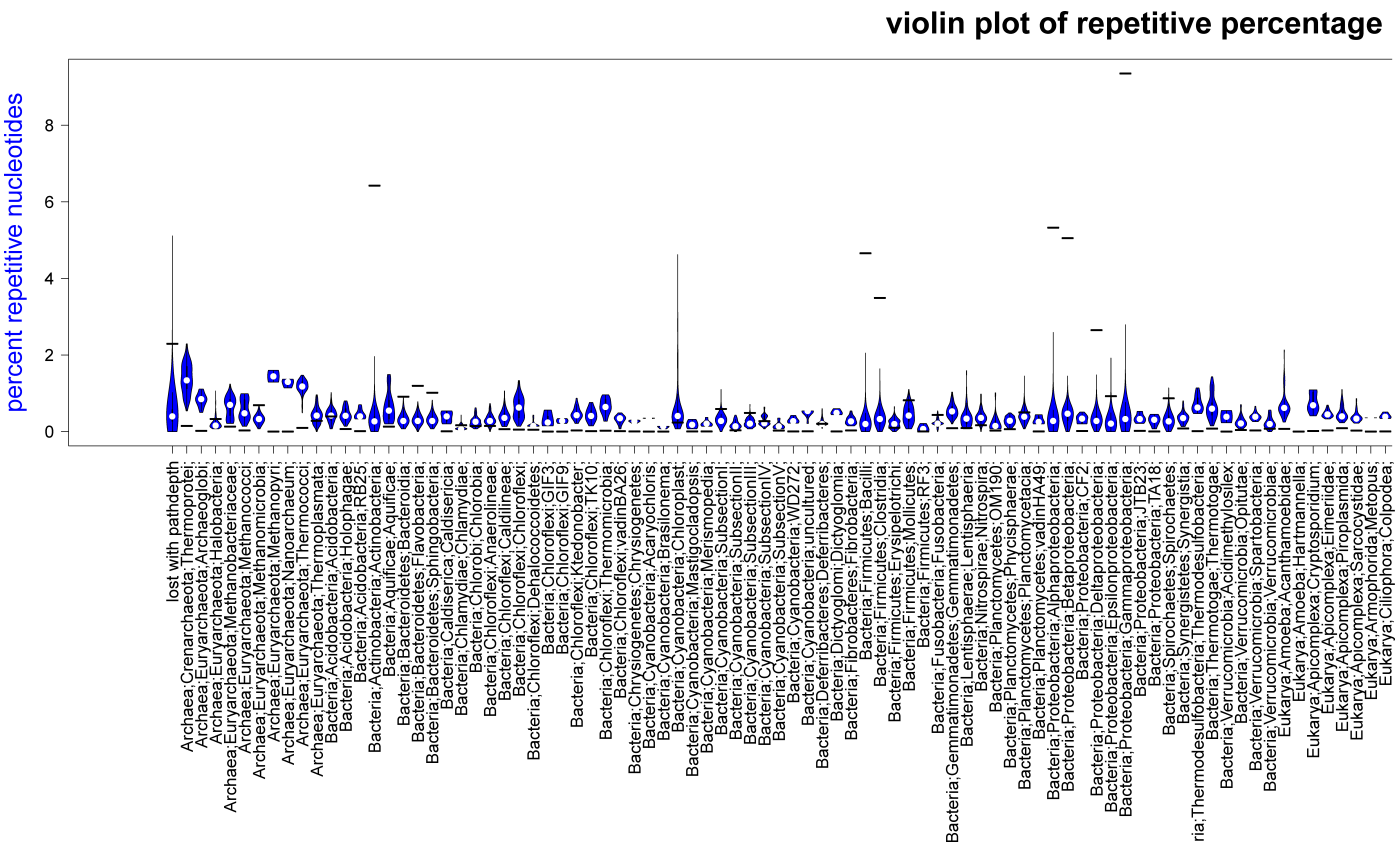

Abbildung 4.15 – Violinplot [HN98] des prozentualen Anteil an repetitiven Nukleotiden im SILVA SEED aufgeteilt nach der taxonomischen Zuordnung in der dritten Taxa *Unterstamm*. Die linke x-Achse zeigt den prozentualen Anteil an repetitiven Nukleotiden und auf der y-Achse die taxonomische Klassifikation. Die Höhe des Violinplots zeigt die Varianz des repetitiven Anteils an, die Breite zeigt die Verteilung.

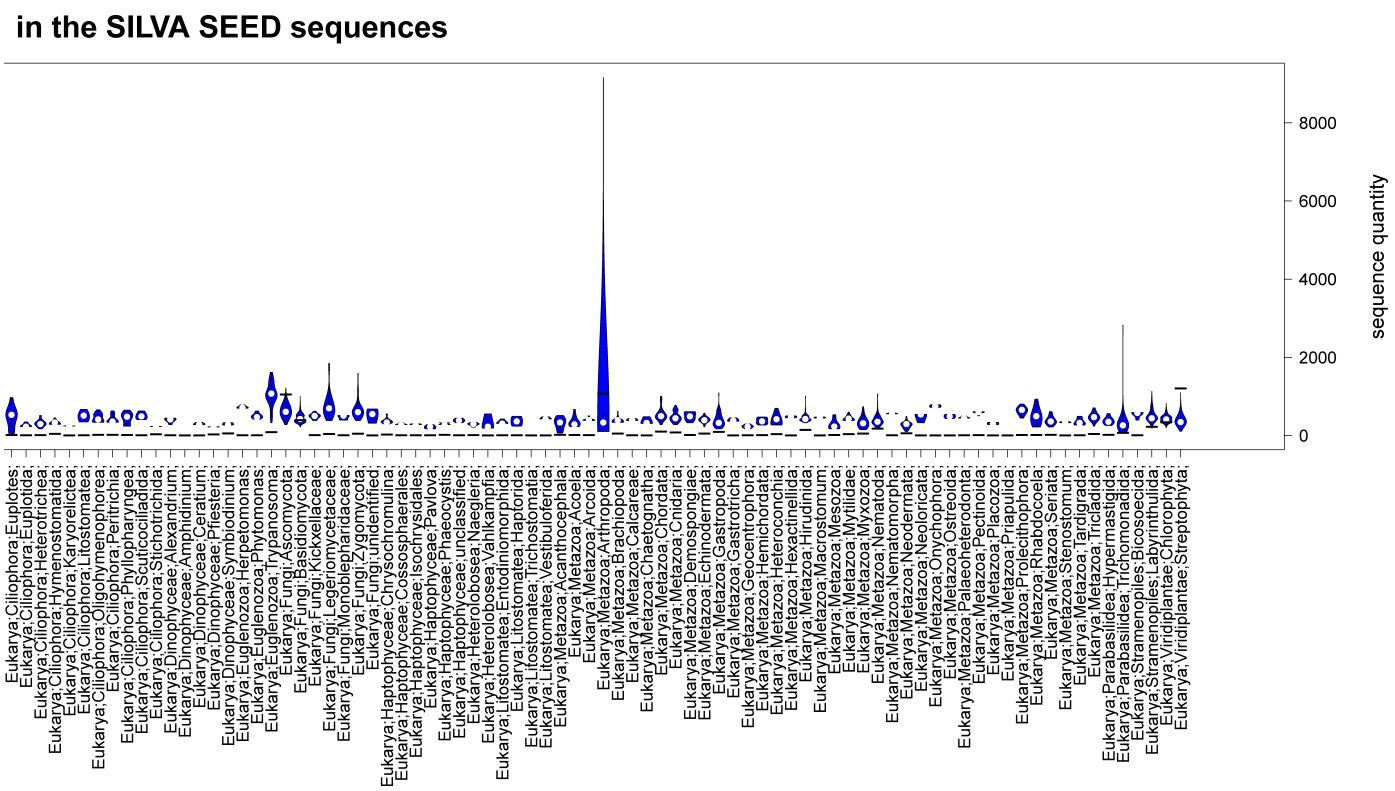

Abbildung 4.16 – Der breite schwarze Strich in den Violinplots zeigt die Sequenzanzahl auf der rechten x-Achse an. Der weisse Punkt in den blauen Violinplots markiert den Mittelwert an repetitiven Anteilen. Der Violinplot mit der Beschriftung ″lost with pathdepth″stellt Sequenzen dar, deren taxonomischer Pfad nicht ausreicht um sie in der Taxa *Unterstamm* darzustellen.

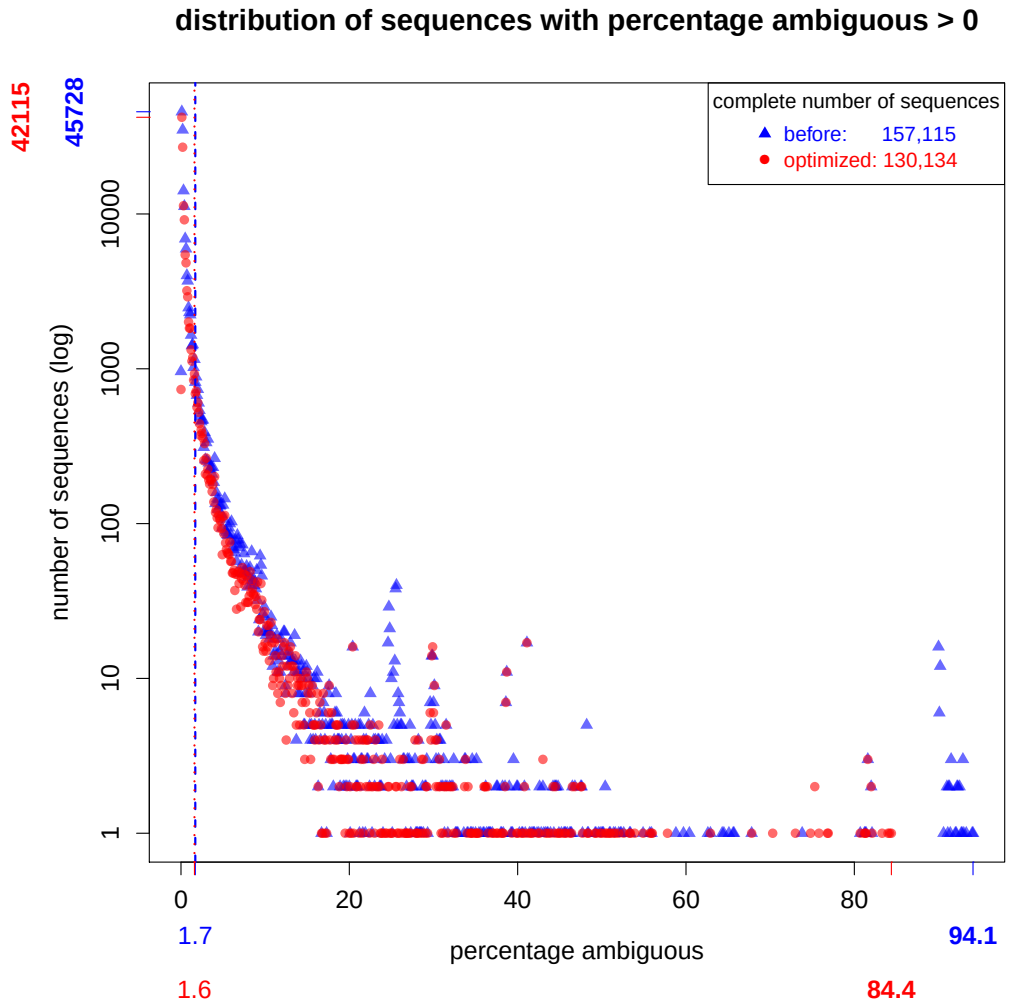

Abbildung 4.17 – Vergleich der Sequenzen die mehrdeutige Basen enthalten in der vorherigen und der optimierten Version. Die x-Achse zeigt in logarithmischer Skalierung die Sequenzanzahl, auf der y-Achse befindet sich der entsprechende Anteil an mehrdeutigen Basen. Die **blauen** Dreiecke stehen für die bisherige und die **roten** Punkte für die optimierte Version. Die roten und blauen Striche an den Achsen zeigen die maximalen Werte, die gestrichelte und gepunktete vertikale Linie markiert den Grenzwert, der verwendet werden müsste um 10% der betroffenen Sequenzen herauszufiltern.

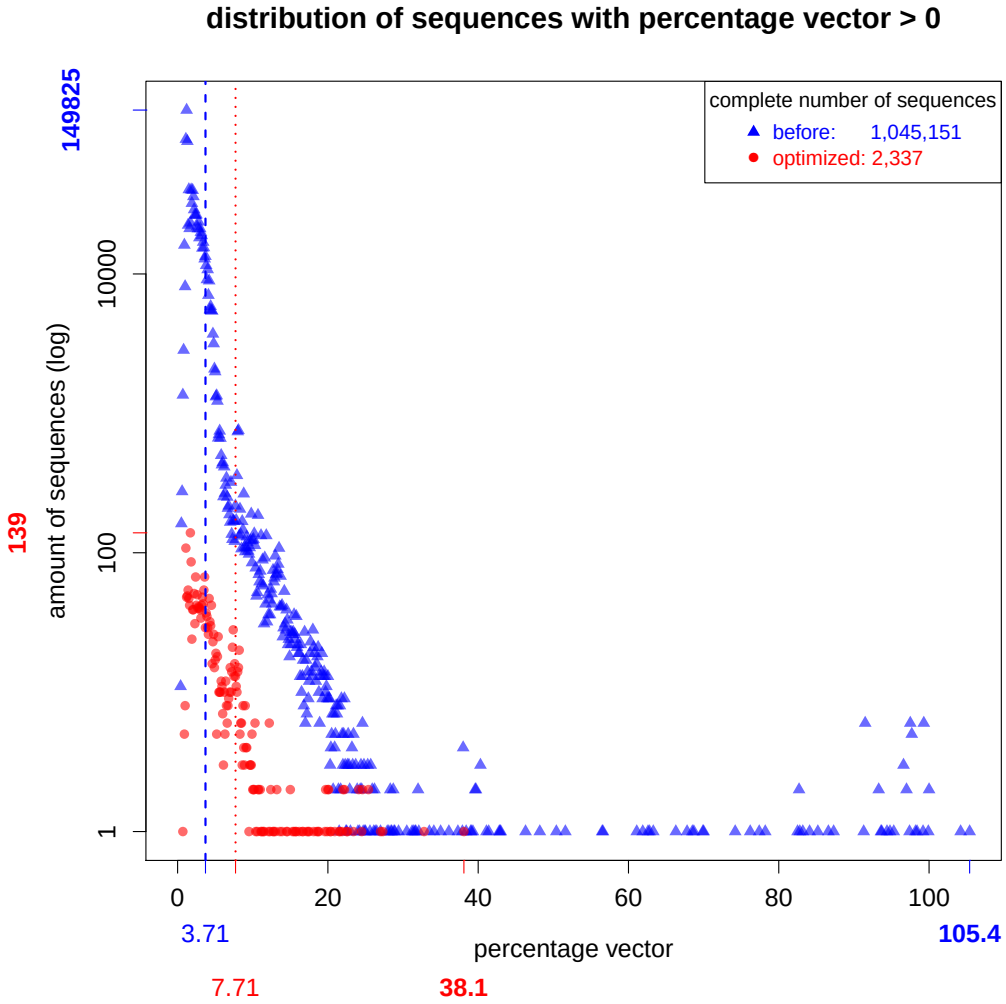

Abbildung 4.18 – Vergleich der Sequenzen die eine Vektorkontamination enthalten in der vorherigen und der optimierten Version. Die x-Achse zeigt in logarithmischer Skalierung die Sequenzanzahl, auf der y-Achse befindet sich der entsprechende Anteil an Vektorkontamination. Die **blauen** Dreiecke stehen für die bisherige und die **roten** Punkte für die optimierte Version. Die roten und blauen Striche an den Achsen zeigen die maximalen Werte, die gestrichelte und gepunktete vertikale Linie markiert den Grenzwert, der verwendet werden müsste um 10% der betroffenen Sequenzen herauszufiltern.

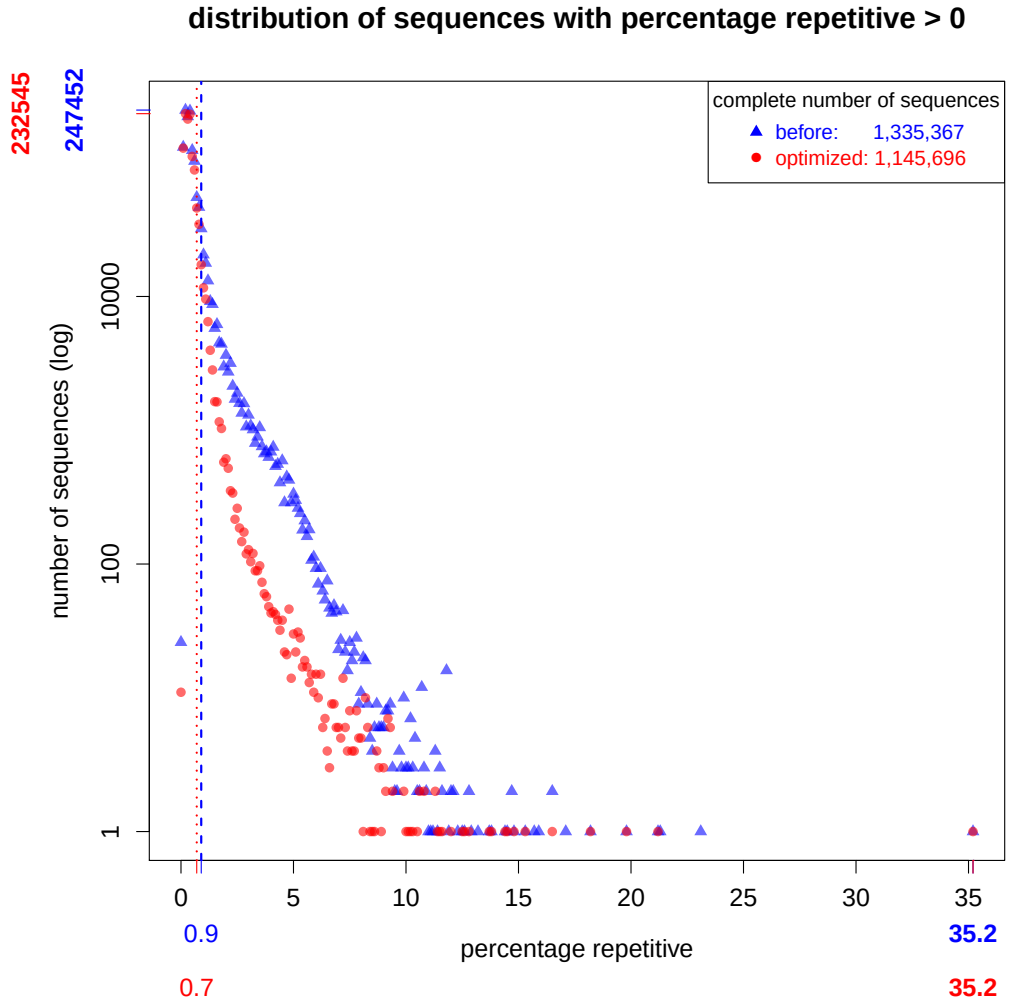

Abbildung 4.19 – Vergleich der Sequenzen die repetitive Nukleotide enthalten in der vorherigen und der optimierten Version. Die x-Achse zeigt in logarithmischer Skalierung die Sequenzanzahl, auf der y-Achse befindet sich der entsprechende Anteil an repetitiven Nukleotiden. Die **blauen** Dreiecke stehen für die bisherige und die **roten** Punkte für die optimierte Version. Die roten und blauen Striche an den Achsen zeigen die maximalen Werte, die gestrichelte und gepunktete vertikale Linie markiert den Grenzwert, der verwendet werden müsste um 10% der betroffenen Sequenzen herauszufiltern.

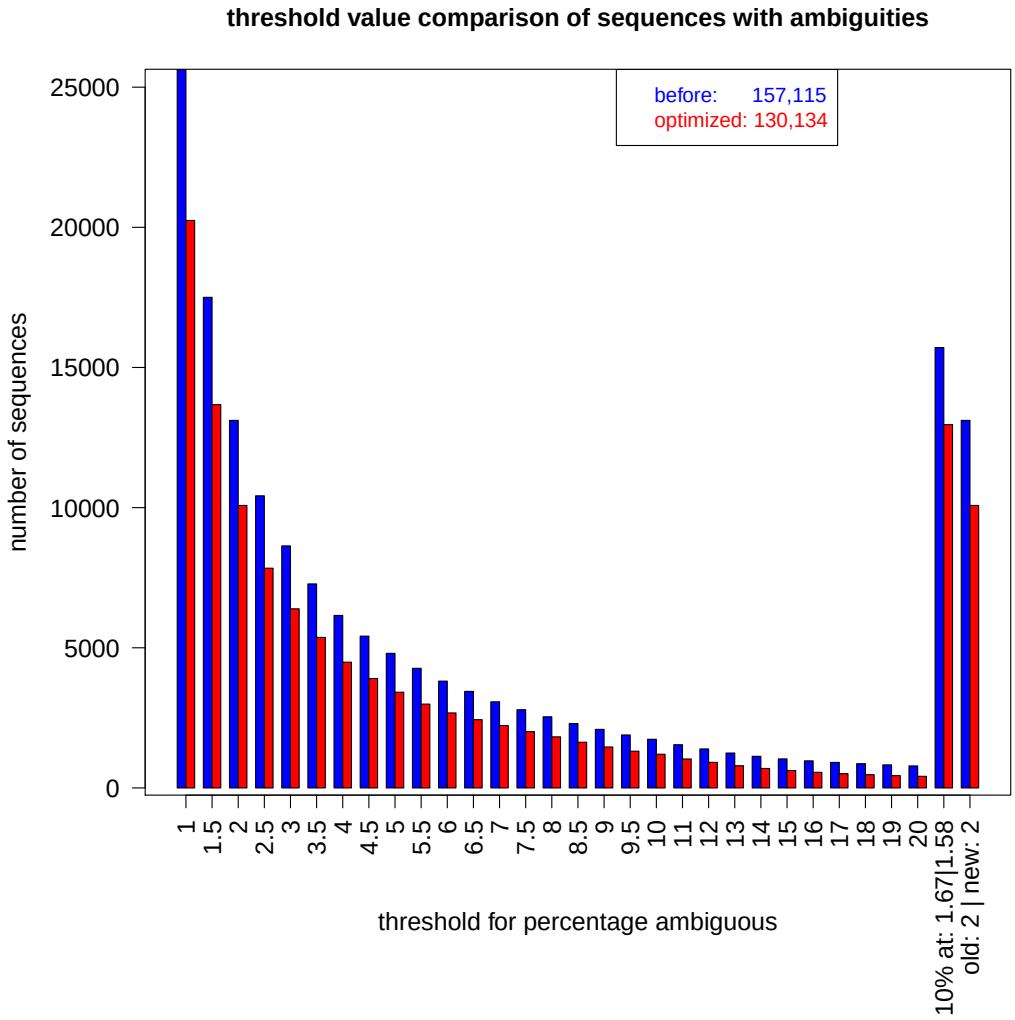

Abbildung 4.20 – Balkendiagramm zur Bestimmung des Grenzwertes für mehrdeutige Basen. Die **blauen** Balken stehen für die bisherige Pipeline, die **roten** Balken für die optimierte Version. Die x-Achse zeigt unterschiedliche Grenzwerte für den prozentualen Anteil an mehrdeutigen Basen an. Die y-Achse zeigt die Sequenzanzahl, welche von dem entsprechenden Grenzwert betroffen wäre. In der Legende steht die Gesamtanzahl an Sequenzen die einen repetitiven Anteil größer null besitzen. Die zweite Balkengruppe von rechts stellt den Grenzwert dar, der gewählt werden müßte, um 10% der betroffenen Sequenzen herauszufiltern.

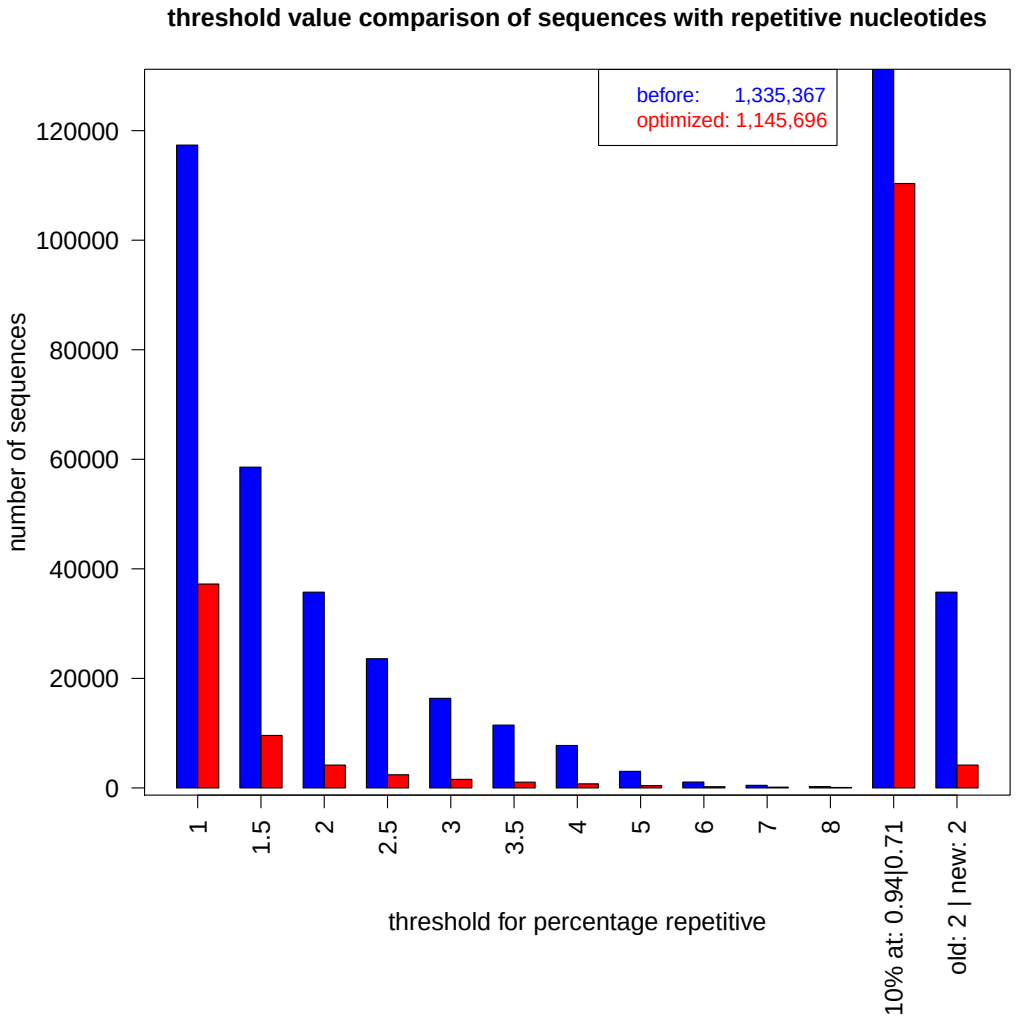

Abbildung 4.21 – Balkendiagramm zur Bestimmung des Grenzwertes für repetitive Nukleotide. Die **blauen** Balken stehen für die bisherige Pipeline, die **roten** Balken für die optimierte Version. Die x-Achse zeigt unterschiedliche Grenzwerte für den prozentualen Anteil an repetitiven Nukleotiden an. Die y-Achse zeigt die Sequenzanzahl, welche von dem entsprechenden Grenzwert betroffen wäre. In der Legende steht die Gesamtanzahl an Sequenzen die einen repetitiven Anteil größer null besitzen. Die zweite Balkengruppe von rechts stellt den Grenzwert dar, der gewählt werden müßte, um 10% der betroffenen Sequenzen herauszufiltern.

Eidesstattliche Erklärung Hiermit erkläre ich an Eides Statt, dass ich die vorliegende Abschlussarbeit
″Qualitätsmanagement ribosomaler RNA Sequenzen in der SILVA Datenbank″

selbständig und ohne fremde Hilfe angefertigt habe. Ich habe dabei nur die in der Arbeit angegebenen Quellen und Hilfsmittel benutzt. Die Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfung vorgelegt und auch noch nicht veröffentlicht.

....
(Ort, (Abgabe-) Datum) (Unterschrift)

Erklärung zu Eigentum und Urheberrecht Hiermit erkläre ich mein Einverständnis, dass die Fachhochschule Bingen die vorliegende Abschlussarbeit den Studierenden und interessierten Dritten zur Einsichtnahme zur Verfügung stellt und unter Nennung meines Namens (Urheber) veröffentlichen darf.

....
(Ort, (Abgabe-) Datum) (Unterschrift)