Stiftung Tierärztliche Hochschule Hannover

**Institut für Tierzucht und Vererbungsforschung**

**Bachelorarbeit**

**Evaluation von Resamplingverfahren zur Analyse von Single-cell RNA-seq Daten**

Sergej Ruff

Matrikelnummer: 10024140

Studiengang: Biologie (B.Sc)

Startdatum: 01.06.2022

Abgabedatum: 02.09.2022

Erstprüfer: Prof. Dr. Klaus Jung

Zweitprüfer: Prof. Dr. Bernd Schierwater

Inhaltsverzeichnis

[Zusammenfassung iii](#_Toc112394248)

[1. Einleitung 1](#_Toc112394249)

[1.1. Einzelzell-Sequenzierung (scRNA-seq) im Vergleich zu Bulk-Ansätzen 1](#_Toc112394250)

[1.2. Einzelzell-Sequenzierung (scRNA-seq): Durchführung 2](#_Toc112394251)

[1.2.1 Einzelzell-Sequenzierung (scRNA-seq): Zellpräparation 2](#_Toc112394252)

[1.2.2 Einzelzell-Sequenzierung (scRNA-seq): Datenanalyse 4](#_Toc112394253)

[1.2.2.1 Einzelzell-Sequenzierung (scRNA-seq): Pre-Processing 5](#_Toc112394254)

[1.2.2.2 Einzelzell-Sequenzierung (scRNA-seq): Qualitätskontrolle 5](#_Toc112394255)

[1.2.2.3 Einzelzell-Sequenzierung (scRNA-seq): Normalisierung 6](#_Toc112394256)

[1.2.2.4 Einzelzell-Sequenzierung (scRNA-seq): Dimensionsreduktion 7](#_Toc112394257)

[1.2.2.5 Single-cell RNA sequencing: Downstreamanalysen 7](#_Toc112394258)

[1.3 Bootstrapping 8](#_Toc112394259)

[1.4 Ziel der Bachelorarbeit 9](#_Toc112394260)

[2. Methoden und Daten 10](#_Toc112394261)

[2.1 Beispieldatensatz: scRNA-seq-Daten aus einer COVID-19-Studie 10](#_Toc112394262)

[2.2 Analyse mit Seurat 11](#_Toc112394263)

[2.3 Qualitätskontrolle mit SingleR 11](#_Toc112394264)

[2.4 Bootstrapping 12](#_Toc112394265)

[2.5 Data.table 13](#_Toc112394266)

[3. Ergebnisse 15](#_Toc112394267)

[3.1. Ergebnisse des Gesamtdatensatzes ohne Bootstrap-Analyse 15](#_Toc112394268)

[3.2. Ergebnisse der Bootstrap-Analyse. 18](#_Toc112394269)

[3.3. Ergebnisse nach Abgleich der Bootstrap-Daten mit dem Referenzdatensatz 19](#_Toc112394270)

[3.3.1. Prozentsatz an richtigen Zuordnungen über alle Bootstrap-Durchläufe 20](#_Toc112394271)

[3.3.2. Prozentsatz an richtigen Zuordnungen pro Cluster 22](#_Toc112394272)

[3.3.3. Anteil an richtigen Zuweisungen auf Gen-Ebene 24](#_Toc112394273)

[3.3.4. Anteil an Markergenen pro Bootstrap-Durchlauf 26](#_Toc112394274)

[4. Diskussion 27](#_Toc112394275)

[5. Literaturverzeichnis 30](#_Toc112394276)

[6.Anhang 35](#_Toc112394277)

[6.1. R-Code für die Bootstrapping-Analyse 35](#_Toc112394278)

[6.2. R-Code für den Gesamtdatensatz und für den Vergleich mit der Bootstrap-Analyse 38](#_Toc112394279)

[Danksagung 47](#_Toc112394280)

# Zusammenfassung

Einzelzell-RNA-Sequenzierung erlaubt die Analyse von Transkripten auf der Ebene einzelner Zellen. Das ermöglicht die Charakterisierung und Identifizierung heterogener Zellpopulation, sowie die Untersuchung von Krankheiten, die die Genexpression einzelner Zellen beeinflussen. Des Weiteren ermöglicht die Einzelzell-RNA-Sequenzierung die Analyse von Expressionsunterschieden zwischen einzelnen Zelltypen. Eine technische Replikation von Einzelzell-RNA-Sequenzierungen, um mehr Daten zu generieren, wird aber oft nicht durchgeführt, weil die Kosten für technische Replikate oft zu hoch sind. Aufgrund des Mangels an technischen Replikaten sind Schlussfolgerung über die Robustheit und Reproduzierbarkeit der Einzelzell-RNA-Sequenzierungsverfahren schwer zu treffen. Das Problem könnte aber gelöst werden, indem man über Bootstrap-Verfahren künstliche Wiederholungen des Experiments generiert. Bootstrap-Verfahren können benutzt werden, um experimentelle Daten und Schlussfolgerungen auf ihre Robustheit und Reproduzierbarkeit zu testen, ohne weitere Experimente ausführen zu müssen. Dazu werden künstliche Wiederholungen des Experiments simuliert, indem man Teilstichproben aus einem bereits vorliegenden Datensatz zieht. Diese künstlichen Wiederholungen sollten damit in der Lage sein, durch das Rekombinieren von Stichproben und Herausziehen von Daten aus einer Stichprobe technische und biologische Replikationen durch Experimente zu ersetzen und damit eine kosteneffiziente Alternative darstellen.

In dieser Arbeit werden Einzelzell-RNA-Sequenzierungsdaten aus einer öffentlichen Datenbank einem Bootstrapping unterzogen, um die Robustheit und Reproduzierbarkeit von Einzelzell-RNA-Sequenzierungsverfahren auf Ebene der Datenauswertung zu testen.

Zu diesem Zweck wurde ein Datensatz mit B-Zellen von COVID 19-Patienten genutzt, um eine Einzelzellanalyse an diesen Datensatz durchzuführen. Die signifikanten und nur in einem Cluster auftretenden Markergene wurden nach der Analyse als eine Referenz genutzt, um ein Vergleich von Markergenen, die aus der Referenz kommen, mit Markergenen aus Bootstrapping-Ansätzen zu ermöglichen. Derselbe Datensatz wurde genutzt, um 100 Bootstrap-Durchläufe durchzuführen und es wurde überprüft, wie oft die Markergene jedes Clusters eines Durchlaufes im Bootstrap mit den Markergenen in jedem Cluster der Referenz übereinstimmen.

Diese Arbeit verdeutlicht, dass Bootstrapping-Analysen geeignet sind, um die Reproduzierbarkeit und Robustheit der bioinformatischen Auswertung von Einzelzell-RNA-Sequenzierungsdaten abschätzen zu können, aber der große Anteil an falschen Clusterzuweisungen in den Bootstrapläufen stellt derzeit noch eine Fehlerquelle dar, die behoben werden muss, bevor man Bootstrapping für weitere Analysen dieser Art anwendet.

# 1. Einleitung

Die Technik der Einzelzell-RNA-Sequenzierung (engl. single-cell RNA sequencing; Abk: scRNA-seq) erlaubt die Detektion und quantitative Analyse von Transkripten auf der Ebene einzelner Zellen. Damit ist eine Analyse der Genexpression nicht mehr auf Zellpopulationen beschränkt und man ist in der Lage Genexpressionsunterschiede zwischen einzelnen Zelltypen zu detektieren, was mit den bisherigen Bulk-RNA-Ansätzen nicht möglich gewesen ist (Macaulay und Voet 2014). Diese Arbeit untersucht auf Ebene der Datenauswertung die Stabilität des scRNA-seq Verfahrens, indem betrachtet wird, wie oft ein Marker-Gen nach Bootstrapping einem bestimmten Cluster zugeordnet wird.

In den folgenden Teilkapiteln werden zunächst die Grundlagen für scRNA-seq dargestellt. Zum einem werden hier die biologischen Aspekte betrachtet und erklärt, welche Vorteile scRNA-seq im Vergleich zu Bulk-RNA-Verfahren hat; und zum anderen werden die technischen Fragen geklärt: wie gewinnt man einzelne Zellen aus einem Gewebe, und wie analysiert man die scRNA-seq Daten in R-Studio? Anschließend wird das Prinzip der Bootstrapping-Verfahren beschrieben.

## 1.1. Einzelzell-Sequenzierung (scRNA-seq) im Vergleich zu Bulk-Ansätzen

Das Transkriptom beschreibt die Gesamtheit aller RNA-Moleküle, die eine Zelle herstellt, und anders als das Genom ist das Transkriptom ein dynamisches System, welches sich je nach Zelltyp und Zellzustand unterscheidet. Das bedeutet, dass sich die Genexpression von Zelle zur Zelle unterscheiden wird und selbst Zellpopulationen, die phänotypisch homogen zu seien scheinen, können Unterschiede aufweisen (Huang 2009). Die herkömmlichen Verfahren, die sich mit der Erhebung von Transkriptomdaten befasst haben, haben sich aber bei ihrer Analyse auf ganze Zellpopulationen bezogen. Unser jetziges Wissen zum Transkriptom bezieht sich also auf Arbeiten, die Tausende Zellen und ihre Transkriptome gleichzeitig analysiert haben. Diese sog. “Bulk”-Ansätze, die ganze Zellpopulationen analysieren, können die Heterogenität der Transkriptome nicht einfangen.

Erst nachdem der erste wissenschaftliche Artikel, welches sich mit dem Transkriptom auf Einzel-Zell-Ebene befasst hat, erschien ist (Tang et al. 2009), stieg die Anzahl an Arbeiten, die sich mit scRNA-seq befassen, rasant an. Single-cell RNA sequencing ermöglicht die Analyse des Transkriptoms auf Einzel-Zell-Ebene. Bulk-Ansätze können nur die Expression von Genen über die ganze Zellpopulationen detektieren, während scRNA-seq die Analyse der Genexpression einzelner Zellen ermöglicht. Anders als Bulk-Ansätze kann scRNA-seq auch Co-Expressionsmuster zwischen zwei Genen kenntlich machen (Macaulay und Voet 2014). Das periphere Blut enthält mehrere verschiedene Zelltypen. Dazu gehören die Lymphozyten, die Monozyten, Die Neutrophilen Zellen und weitere Zellen. Das periphere Blut und viele andere Gewebe des Körpers bestehen also aus heterogenen Zellpopulationen. Die Identifikation und Charakterisierung der einzelnen Zellen innerhalb einer heterogenen Zellpopulation wird durch scRNA-seq ermöglicht. Genauso kann man auch seltene Zellpopulationen innerhalb eines Gewebes entdecken und die Genexpressionsunterschiede zwischen den Zellen innerhalb einer heterogenen Zellpopulation untersuchen. Die Untersuchung der Genexpressionsunterschiede auf Einzel-Zell-Ebene ist wichtig für die Aufdeckung vieler Krankheiten und ihrer Mechanismen, weil Krankheiten wie Malaria (Rivera-Correa et al. 2017) und COVID-19 (Biasi et al. 2020) gezielt die Genexpression bestimmter Zelltypen verändern, um sich vor dem Immunsystem des Wirtes zu schützen.

## 1.2. Einzelzell-Sequenzierung (scRNA-seq): Durchführung

### 1.2.1 Einzelzell-Sequenzierung (scRNA-seq): Zellpräparation

Die Isolation von Zellen für scRNA-seq erfolgt aus festen Geweben wie zum Beispiel Darmzotten oder aus Zellsuspensionen wie dem peripheren Blut. Die Isolation erfolgt über Droplet basierte Technologien. Oft genutzte Technologien sind das 10X Genomics Chromium, Dropseq und inDrop (Kulkarni et al. 2019). Sie alle funktionieren nach demselben Grundprinzip und können deshalb in folgende Schritte eingeteilt werden: Vorbereitung der Zellsuspension, eine optionale Zellsortierung, Einkapseln einzelner Zellen in Droplets, cDNA-Synthese und Amplifikation, Erstellen einer Bibliothek und Sequenzierung sowie Vorprozessierung der gewonnenen Daten **(Abbildung 1,Bildquelle modifiziert nach** Salomon et al. 2019**)**.



***Abbildung 1****: Schematische Darstellung einer Droplet basierten Zellpräparation für scRNA-seq. Bestehend aus Vorbereitung der Zellsuspension, eine optionale Zellsortierung, Einkapseln einzelner Zellen in Droplets, cDNA-Synthese und Amplifikation, Erstellen einer Bibliothek und Sequenzierung sowie Vorprozessierung der gewonnenen Daten (Bildquelle modifiziert nach Salomon et al. 2019).*

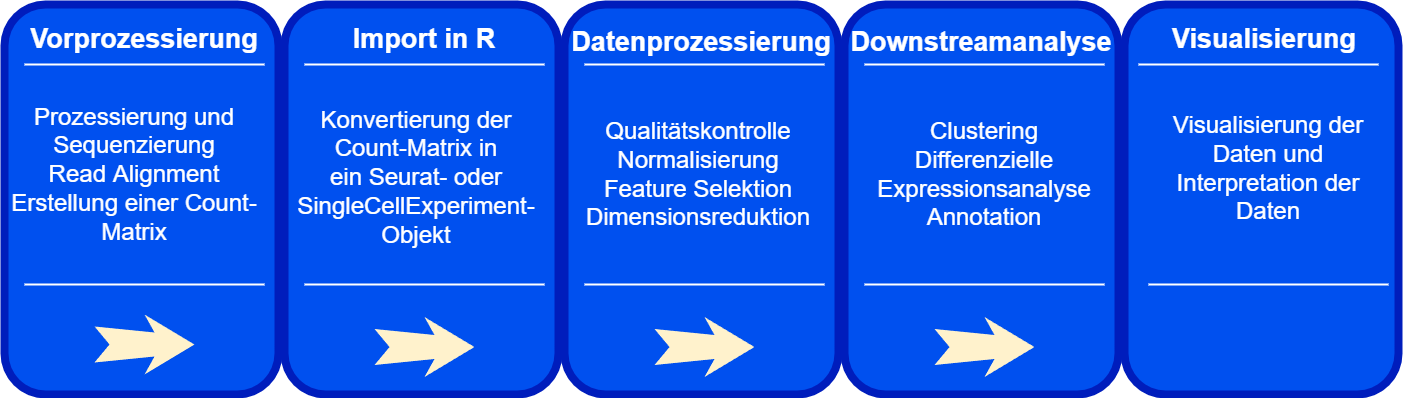
Zu Beginn der Zellpräparation müssen feste Gewebe in eine Zellsuspension dissoziiert werden. Durchflusszytometrie kann genutzt werden, um mögliche tote Zellen zu entfernen, wodurch die gewünschten Zellen angereichert werden. Ziel der Droplet basierten Verfahren ist es, einen abgeschlossenen Reaktionsraum zu erschaffen. Ein Bead und einzelne Zellen aus der Suspension treffen hier in einem Mikrofluid aufeinander. Ein Öl trennt die Flüssigkeit in einzelne Droplets auf. Im Droplet befindet sich ein Lysispuffer. Sollte sich im Droplet eine Zelle befinden, dann wird diese Zelle durch den Lysispuffer lysiert und die RNA aus der Zelle wird im Droplet freigesetzt.

Die eingeführten Beads besitzen einen Oligo-d(T)-Primer auf ihrer Oberfläche, wodurch sie komplementär an den Poly-A-Schwanz der RNA binden können. Des Weiteren besitzen die Beads Cell Barcodes, Unique Molecular Identifiers (UMI) und TSOs (template switching oligonucleotide). Cell Barcodes sind identische, mehrmals auftauchende Sequenzen auf den Beads und Sie werden genutzt für die Identifikation einzelner Zellen. UMI sind einzigartige Sequenzen auf den Beads, die als Tags genutzt werden, um die einzelnen Transkripte zu markieren (Salomon et al. 2019). So ist es möglich die einzelnen Transkripte einer bestimmten Zelle zuzuordnen.

Nachdem die einzelnen RNAs an den Bead gebunden haben, werden über Reverse Transkriptase-Polymerase-Kettenreaktionen cDNAs mit Barcode und UMIs generiert. Diese cDNAs beinhalten nicht nur die Sequenz des Transkripts, sondern Sie enthalten auch die Cell Barcodes und die UMI. Für die reverse Transkription benutzt man die M-MLV reverse Transkriptase (Moloney murine leukemia virus reverse transcriptase). Die M-MLV reverse Transkriptase fügt am Ende der cDNA drei zusätzliche Cytosine (C) hinzu. Über diese zusätzlichen Cs kann ein zweiter TSO-Primer an die Enden der cDNAs angehängt werden, der mit der RT-Reagenz hinzugegeben wird. Die TSOs werden eingesetzt, um doppelsträngige cDNAs zu generieren, die dann über PCR amplifiziert werden (Zhu et al. 2001)**.** Die UMI ermöglichen eine Einordnung derAmplifikate in Gruppen mit gleichen UMI. So können Amplifikationsartefakte normalisiert werden (Salomon et al. 2019). Im nächsten Schritt werden cDNA-Bibliotheken erstellt. Diese cDNAs werden über Next-Generation Sequenzer sequenziert und vor-prozessiert (De-multiplexing und Data Alignment). Die vorprozessierten Daten werden in Form von FASTQ-Dateien gespeichert.

### 1.2.2 Einzelzell-Sequenzierung (scRNA-seq): Datenanalyse

Eine Besonderheit von Single-cell RNA sequencing ist die Menge an Zellen und Features (Genen), die analysiert werden. Denn die Analyse von scRNA-seq-Daten behandelt hochdimensionale Daten, weil Tausende Gene beobachtet werden, und es behandelt Tausende bis Millionen von Zellen (Amezquita et al. 2020). Gleichzeitig ist zu bedenken, dass die hohe Anzahl an Zellen meist von wenigen biologisch unabhängigen Proben stammt. Die Menge an beobachteten Daten und die Komplexität des Systems verlangen neue bioinformatische Ansätze, um scRNA-seq Daten zu lagern, zu verwalten und zu bearbeiten. “*SingleCellExperiment” Class* von *Bioconductor* (Amezquita et al. 2020) und *“Seurat”* (Hao et al. 2021) sind oft genutzte Container, um Single Cell *Assay* Daten und Metadaten der Statistik- und Bioinformatiksoftware R **(version 4.1.3, www.r-project.org)** zu lagern. Count-Matrices können in der Assay-Komponente von “*SingleCellExperiment”* und “*Seurat”* gelagert werden. Die Assay-Komponente ist selbst wie eine Matrix strukturiert, wobei die Reihen die Features (Gene) repräsentieren und die Spalten des Assays repräsentieren die Zellen. Somit können alle wichtigen scRNA-seq-Daten, die für eine Analyse benötigt werden, gelagert und bearbeitet werden. Die Analyse von scRNA-seq-Daten nach der Zellisolation folgt dem Arbeitsablauf dargestellt auf Abbildung 2 (Bildquelle modifiziert nach Amezquita et al. 2020).

***Abbildung 2****: Arbeitsablauf einer Analyse von scRNA-seq Daten (Bildquelle modifiziert nach Amezquita et al. 2020).*

#### 1.2.2.1 Einzelzell-Sequenzierung (scRNA-seq): Pre-Processing

Am Anfang werden die amplifizierten cDNA-Bibliotheken sequenziert. Die sequenzierten cDNA-Bibliotheken werden daraufhin vorprozessiert. Zur Vorprozessierung gehört das *Demultiplexing* der cDNA-Bibliotheken und das *Alignment* von Reads an ein Referenz-Transkriptom. Die cDNA aus verschiedenen Zellen werden aus verschiedenen Zellen zusammengeführt, um die cDNA-Sequenzen gleichzeitig zu sequenzieren. *Demultiplexing* beschreibt einen Prozessierung-Schritt, bei dem die Barcodes, die an die Transkripte gebunden wurden, genutzt werden, um die cDNA-Sequenzen den richtigen Zellen zuzuordnen. Alignment beschreibt hier die Anordnung der Transkripte an ähnliche Regionen eines Referenz-Transkriptoms. Die Expressionswerte werden in einer Count-Matrix gespeichert. Die Anzahl an Reads, die pro Gen in einer Zelle auftauchen, wird in der Count-Matrix festgehalten. Die Count-Matrix kann daraufhin in R importiert und in ein *“SingleCellExperiment”* Objekt oder “*Seurat*” Objekt umgewandelt werden.

#### 1.2.2.2 Einzelzell-Sequenzierung (scRNA-seq): Qualitätskontrolle

Im Laufe der Zellpräparation und Gewinnung der Transkripte aus den einzelnen Zellen gibt es mehrere Fehlerquellen, die die Qualität der cDNA-Bibliothek negativ beeinträchtigen können. Solche Fehlerquellen sind oft auf Zellschäden während der Herstellung der Zellsuspension oder auf Fehler während der cDNA-Bibliothek-Vorbereitung zurückzuführen. Bibliotheken, wo sich solche Fehler anhäufen, haben weniger exprimierte Gene, geringe Gesamt-Counts und einen hohen Anteil an mitochondrialen Reads(Amezquita et al. 2020). Damit erschweren diese Fehler auch die spätere Analyse und Interpretation der Daten. Die Qualitätskontrolle soll fehlerhafte Reads und beschädigte Zelle rausfiltern, damit die weitere Analyse nicht durch sie beeinflusst wird. Wenn Schäden in der Zellmembran während der Zellpräparation auftreten, dann kommt es dazu, dass zytoplasmatische RNAs aus der Zelle verloren gehen, während sich die mitochondriale RNA anreichert, weil die Mitochondrien zu groß sind, um die Zelle zu verlassen (Amezquita et al. 2020). Zellen mit einem hohen Anteil an mitochondrialer RNA werden aus diesem Grund während der Qualitätskontrolle rausgefiltert. Weitere Kriterien, die genutzt werden können, um geschädigte Zellen raus zu filtern, sind die *library size* und die Anzahl an Genen mit Counts pro Zelle (Amezquita et al. 2020). Die *library size* ist die Gesamtanzahl an Counts aller Gene pro Zelle. Eine Zelle mit einer geringen *library size* hat eine schlechte Qualität, weil eine geringe *libary size* auf den Verlust von RNAs hindeutet. Alternativ kann man die Zellen mit wenigen exprimierten Genen anschauen. Zellen, die viele Features ohne Reads (Nullzählungen) besitzen, deuten auf eine schlechte Qualität hin.

#### 1.2.2.3 Einzelzell-Sequenzierung (scRNA-seq): Normalisierung

Nach der Qualitätskontrolle werden die Daten normalisiert, um systematische Unterschiede zu entfernen. Durch die Normalisierung der Daten beruhen die beobachteten Unterschiede nur auf biologische Unterschiede und nicht systematische Unterschiede, die auf Fehler während der cDNA-Synthese und PCR-Amplifikation zurückzuführen sind. Oft wird dafür Variationen der “Counts per million” oder CPM-Normalisierung benutzt (Luecken und Theis 2019). Das heißt, es werden oft modifizierte Faktoren benutzt, um die Daten zu modifizieren. Eine spezielle Normalisierungsmethode für scRNA-seq-Daten bietet das scran-Package (Lun et al. 2016). Scran berechnet einen *size factor*, indem es alle Zellen zusammenfasst und eine lineare Regression über alle Gene durchführt (Luecken und Theis 2019). Nach der Normalisierung werden die Daten log(x+1) -transformiert. Das hat den Vorteil, dass die transformierten Daten *log Fold Changes* darstellen und damit die Expressionsunterschiede zwischen den Genen repräsentieren**.** Weiterhin setzen weitere Downstreamanalysen voraus, dass die Daten einer Normalverteilung folgen(Luecken und Theis 2019). Vieth et al. (2017) haben gezeigt, dass scRNA-seq-Daten keiner lognormalisierten Verteilung folgen, aber die Transformation ist eine Ausreichende Annäherung für weitere Downstreamanalysen (Luecken und Theis 2019).

#### 1.2.2.4 Einzelzell-Sequenzierung (scRNA-seq): Dimensionsreduktion

Dimensionsreduktionen und Clustering-Verfahren vergleichen Zellen basierend auf ihren Expressionsprofilen (Amezquita et al. 2020). Bevor Dimensionsreduktionen und Clustering durchgeführt werden kann, müssen erstmal die biologisch relevanten Gene selektiert werden. Dieser Prozessierungsschritt heißt auf Englisch “Feature Selection”. Für die Selektion wird die Annahme gemacht, dass eine erhöhte Varianz auf biologische Unterschiede hindeutet (Amezquita et al. 2020). Aus diesem Grund werden variable Gene selektiert. Daraufhin erfolgt eine Dimensionsreduktion und Clustering. Eine Dimensionsreduktion reduziert die hohe Anzahl an Dimensionen, und damit auch die große Anzahl an Geninformationen, auf wenige Dimensionen. Das hat den Vorteil, dass man Features (Gene), die vom selben biologischen Prozess beeinflusst werden, auf dieselben Dimensionen reduzieren können (Amezquita et al. 2020). Damit wird die Effizienz der folgenden Downstreamanalysen verbessert und die Visualisierung der Daten ermöglicht. Die Hauptkomponentenanalyse (Englisch: principal component analysis (PCA); (Pearson 1901)) ist eine weitverbreitete Methode für Dimensionsreduktionen. ScRNA-seq benutzt PCA aber nur zum Reduzieren der Daten auf wenige Dimensionen und nicht für die Visualisierung der Daten. Für die Visualisierung der Daten benutzt man oft t-SNE (t-distributed stochastic neighbour embedding; (van der Maaten und Hinton 2008)) oder UMAP (uniform manifold approximation and projection for dimension reduction; (McInnes et al. 2018)). Die Auftrennung zwischen linearen Dimensionsreduktionsverfahren für das Zusammenfassen der Daten und Nicht-linearen Dimensionsreduktionen für die Visualisierung der Daten ist auf die Struktur der scRNA-seq-Daten zurückzuführen. ScRNA-seq-Daten verhalten sich nicht linear. Deshalb eignen sich lineare Verfahren wie PCA nicht für die Visualisierung von Daten. Amir et al. (2013) etablierten aus diesem Grund nicht-lineare Ansätze für die Visualisierung von scRNA-seq-Daten, wobei lineare Ansätze weiterhin für die Downstreamanalyse verwendet werden und die ausgewählten Dimensionen aus PCA werden als Imput für die nicht-linearen Ansätze wie t-SNE und UMAP verwendet, um die Rechenleistung zu senken.

#### 1.2.2.5 Single-cell RNA sequencing: Downstreamanalysen

Clustering gruppiert Zellen in Gruppen, indem Zellen mit ähnlichen Expressionsprofilen in gleiche Gruppen sortiert werden. Die Zellgruppierung verlangt typischerweise eine graphische Repräsentation der Daten (Luecken und Theis 2019). “S*eura*t” (Hao et al. 2021)nutzt die reduzierten Dimensionen aus der linearen Dimensionsreduktion (Siehe 1.2.2.4) als Imput, um einen KNN (k-nearest neighbor )-Graphen basierend auf den euklidischen Abstand aufzubauen. Standartgemäß wird nach der graphischen Darstellung der Daten der Louvain Algorithmus angewendet, um die Zellen zu clustern (Luecken und Theis 2019). Die Analyse von differenziert exprimierten Genen kann genutzt werden, um Markergene, die die Cluster auszeichnen, zu identifizieren. Durch einen Vergleich mit Gene Ontology (GO) oder Kyoto Encyclopedia of Genes and Genomes (KEGG) Sammlungen können die Zellen zusätzlich annotiert werden. Für die Annotation der Zellen werden Markergene genutzt, um die Identität der Zellen in den einzelnen Clustern zu bestimmen.

## 1.3 Bootstrapping

Reproduzierbarkeit ist einer der großen Prinzipien der Wissenschaftstheorie. Es ist die erfolgreiche Reproduktion der Daten und Resultate, die einem Experiment Gültigkeit verleiht. Eine technische Replikation von Hochdurchsatzsequenzierungen wird aber oft nicht durchgeführt, weil die Kosten für technische Replikationen oft zu hoch sind (Saremi et al. 2022). Bootstrap-Verfahren wurden ursprünglich von Bradley Efron entwickelt, um mittels einer Stichprobe die Verteilung einer Test-Statistik zu schätzen (Efron 1979). Das Verfahren ist nach einer Geschichte des „Lügenbarons“ Münchhausen benannt, der sich selbst am Stiefelriemen (engl. Bootstrap) aus dem Matsch gezogen hat. Dementsprechend werden bei Boostrap-Verfahren mehrfach hintereinander Teilstichproben aus einer Stichprobe gezogen. Bootstrap-Verfahren können auch genutzt werden, um experimentelle Daten und Schlussfolgerungen auf ihre Robustheit und Reproduzierbarkeit zu testen, ohne weitere Experimente ausführen zu müssen. Diese Ansätze sind kosteneffizient, weil Sie keine weiteren Experimente voraussetzen. In den Bereichen der Metagenomik(Saremi et al. 2021) und der Bulk-RNA-Sequenzierung (Saremi et al. 2022) gibt es bereits Arbeiten, die Bootstrap-Ansätze nutzen, um Daten auf ihre Robustheit und Reproduzierbarkeit zu überprüfen. Die Daten, die man aus einem Experiment erlangt, sind Stichproben aus einer größeren Population (Population an Genen, Zellen oder Beobachtungen). Bootstrapping rekombiniert diese Stichprobe und zieht einen Anteil an Daten aus der Stichprobe (Meyer et al. 1986). Es werden also wiederholt Stichproben aus der eigentlichen Stichprobe gezogen. Dieses wiederholte und zufällige Resampling (d.h. Rekombinieren und Herausziehen von Daten aus der Stichprobe) repräsentiert ein Resampling der gesamten Population und soll damit die technische Replikation durch Experimente ersetzen.

## 1.4 Ziel der Bachelorarbeit

Wie in Teilkapitel 1.3 bereits erwähnt, gibt es bereits Arbeiten im Bereich der Metagenomik und der RNA-Sequenzierung, die durch Bootstrapping die Reproduzierbarkeit von Daten analysiert haben. Diese Arbeiten zeigen, dass man im Transkriptomik-Bereich Bootstrapping nutzen kann, um Daten und Ergebnisse auf Unsicherheiten zu überprüfen. Gleichzeitig beschäftigten sich diese Arbeiten aber nur mit RNA-Proben, die ganze Zellpopulationen betrachten. Das heißt, es existieren viele Bootstrapping-Experimente, die sich mit Bulk-RNA-Ansätzen auseinandersetzen, aber es gibt nicht viele Ansätze, die ein Bootstrapping auf Einzelzell-Ebene durchgeführt haben.

Das Ziel dieser Arbeit ist es die Stabilität des scRNA-Seq-Verfahrens zu untersuchen. Die Unsicherheit der Daten wird untersucht, indem überprüft wird, wie oft die Markergene jedes Clusters eines Durchlaufes im Bootstrap mit den Markergenen in jedem Cluster eines Gesamtdatensatzes, welches als Referenz dienen soll, übereinstimmen. In dieser Arbeit wird ein Datensatz mit B-Zellen von COVID 19-Patienten genutzt und die signifikanten Markergene, die ein einziges Mal im Gesamtdatensatz auftauchen, mit den Markergenen aus 100 durchgeführten Bootstrapping-Durchläufen verglichen.

# 2. Methoden und Daten

Für diese Arbeit wurden Funktionen aus mehreren Packages genutzt, aber auch eigene Funktionen in RStudio (version 4.1.3) implementiert. Die genutzten Skripte befinden sich im Anhang dieser Bachelorarbeit. In den folgenden Teilkapiteln werden die genutzten Packages, ihr Nutzen in der Bachelorarbeit und die genutzten Parameter dargestellt.

Die Arbeit nutzt auch einen Datensatz, der auf der ArrayExpress-Datenbank hochgeladen wurde (Athar et al. 2019); https://www.ebi.ac.uk/arrayexpress). Der genutzte Datensatz wird in dem folgenden Teilkapitel dargestellt.

### 2.1 Beispieldatensatz: scRNA-seq-Daten aus einer COVID-19-Studie

Für die Arbeit wurde ein Datensatz mit scRNA-seq Daten benötigt. Aus diesem Grund wurde auf die Daten aus der Arbeit von Schulheiß et al. (2021) zugegriffen. Ihre Arbeit mit dem Titel “Maturation trajectories and transcriptional landscape of plasmablasts and autoreactive B cells in COVID-19" beschäftigt sich mit dem Einfluss der COVID 19 Krankheit auf die B-Zell-Differenzierung und Immunpathologie. Die prozessierten Daten ihrer Arbeit sind auf der ArrayExpress-Datenbank hinterlegt wurden. Der Datensatz besitzt die Zugriffsnummer/ “Accession-ID” E-MTAB-11011, aber man kann es auch unter folgenden Link abrufen: https://www.ebi.ac.uk/arrayexpress/experiments/E-MTAB-11011/ (Letzter Zugriff: 29.07.2022, 21:16). Schulheiß et al. (2021) benutzen scRNA-seq, um sich die peripheren B-Zellen von 4 COVID 19-Patienten, 3 genesenen Patienten und einem Patienten, der zum Zeitpunkt der Arbeit noch nicht an COVID 19 erkrankt gewesen ist, anzuschauen. Diese Arbeit beschränkt sich auf die Daten der 4 COVID 19 Patienten. Die Daten der 4 Patienten mit COVID 19 haben in dieser Arbeit den Namen “active” bekommen. Der Datensatz Active enthält 10,050 Zellen und 2000 Features (Gene). Die geringe Anzahl an Features kann dadurch erklärt werden, dass es sich bei den Daten um prozessierte Daten handelt.Schulheiß et al. (2021) haben 2000 Features selektiert und nur diese auf ArrayExpress zur Verfügung gestellt. Auf ArrayExpress findet man die Daten der Patienten als Zip-Datei unter dem Datei-Namen “E-MTAB-11011.processed.1.zip”. Nachdem man die Daten heruntergeladen und die Zip-Datei entpackt hat, findet man die Daten der 4 COVID 19-Patienten in der Datei “pbmc.active.2.5.3.8\_gex\_and\_vdj.rds”. Es handelt sich hierbei um eine rds.-Datei, die die Daten der 4 COVID 19-Patienten als ein großes “*Seurat”*-Objekt lagert. Schulheiß et al. (2021) führten in ihren prozessierten Datensatz bereits eine Qualitätskontrolle der Daten durch, indem Sie Zellen mit weniger als 200 Genen oder mehr als 2500 Genen und einem Anteil von mitochondrialen Genen über 10 % entfernt haben. Weiterhin enthält der Datensatz zwei Dimensionsreduktionen (PCA und UMAP), auf die man zugreifen kann. Mit readRDS() kann die Datei in RStudio importiert werden.

### 2.2 Analyse mit Seurat

Die Analyse der Stabilität des scRNA-seq verlangt den Vergleich zwischen einem Originaldatensatz und den künstlichen Replikaten, die man durch das Resampling der Originaldaten erhält. Aus diesem Grund wurden zwei R-Skripte erstellt: ein R-Skript für die Analyse des Gesamtdatensatzes und ein R-Skript für das Bootstrapping. Die Beide Skripte befinden sich im Anhang dieser Arbeit. Beide R-Skripte nutzen den Datensatz beschrieben in Teilkapitel 2.1 und beide Skripte nutzen das “*Seurat*”-Packet (Hao et al. 2021) für die Analyse der Daten in RStudio. Wie bereits in Teilkapitel 2.1 erwähnt, handelt es sich bei dem genutzten Datensatz um prozessierte Daten. Eine Qualitätskontrolle ist aus diesem Grund nicht notwendig gewesen. Nach dem Importieren der Daten in RStudio wurden Features über die Funktion FindVariableFeatures() mit der Methode “vst” selektiert. Über die Funktion ScaleData() wurden die Daten linear transformiert und eine Hauptkomponentenanalyse wurde mit der Funktion RunPCA() durchgeführt. Schulheiß et al. (2021) identifizierten die ersten 10 PCs (principal components) als die Stabilsten für die weitere Analyse. Um mit dem Gesamtdatensatz die Daten von Schulheiß et al. (2021) so nah wie möglich zu replizieren, wurden in dieser Arbeit 10 PCs für die weitere Analyse verwendet. Die 10 Dimensionen aus der Hauptkomponentenanalyse wurden als Imput für das Clustering mit den Funktionen FindNeighbors() und FindClusters() benutzt. Als Auflösung für FindClusters() wurde 0.5 genutzt. Nachdem die Zellen einem Cluster zugeordnet wurden, wird eine differenziale Genexpressionsanalyse über die Funktion FindMarkers() durchgeführt, um die Markergene für jedes Cluster zu identifizieren. Für das Bootstrapping wurde auf die prozessierten Daten zurückgegriffen, wodurch das Skript um mehrere Funktionen gekürzt werden konnte (Siehe Skript im Anhang).

### 2.3 Qualitätskontrolle mit SingleR

Der auf ArrayExpress zur Verfügung gestellte Datensatz von Schulheiß et al. (2021)enthält B-Zellen aus dem peripheren Blut. Um zu überprüfen, ob es sich bei den Zellen aus dem Datensatz wirklich um B-Zellen handelt, wurde eine weitere Qualitätskontrolle eingeführt. Über eine Annotation der Zellen aus dem Datensatz soll überprüft werden, ob Kontaminierungen durch andere Zelltypen existieren. Für diesen Zweck wurden das SingleR-Packet (Aran et al. 2019) und das celldex-Packet (Aran et al. 2019) verwendet. SingleR ist ein Packet, welches genutzt werden kann, um Zellen durch einen Vergleich mit einer Referenz zu annotieren. Als Referenz wurde das HumanPrimaryCellAtlasData() (Mabbott et al. 2013), welches im celldex-Packet implementiert ist, verwendet. HumanPrimaryCellAtlasData() enthält öffentlich verfügbare Microarray Datensätze mit primären, humanen Zellen. Eine direkte Annotation des Datensatzes ist nicht möglich, da SingleR nur SummarizedExperiment-Objekte oder numerische Matrizen mit Expressionswerten aus scRNA-seq-Experimenten als Imput akzeptiert. Eine direkte Konvertierung des “*Seurat”*-Objektes in ein SummarizedExperiment-Objekt ist auch nicht möglich. Stattdessen wurde das “*Seurat*”-Objekt mit den Daten der COVID 19-Patienten zuerst in ein “*SingleCellExperiment”*-Objekt und dann in ein SummarizedExperiment-Objekt konvertiert. Dann wurden über SingleR() die Daten der COVID 19-Patienten mit den Daten aus dem HumanPrimaryCellAtlas abgeglichen und die Zellen aus dem Datensatz wurden annotiert. Über die sum-Funktion werden daraufhin die Labels, die “B\_cell” in ihrem Namen enthalten, zusammenaddiert und die Prozentzahl berechnet. Damit können wir die Prozentzahl an gefundenen B-Zellen im Datensatz angeben lassen. Zellen, die nicht gegen eine B-Zelle in der Referenz mappen, werden daraufhin entfernt.

### 2.4 Bootstrapping

Ein neues R-Skript wurde für das Bootstrapping der Daten angelegt. Das Skript befindet sich im Anhang dieser Bachelorarbeit. 80% der Zellen im “active”-Datensatz wurde für das Bootstrapping genutzt. Für das Bootstrapping selbst wurde eine for-Schleife genutzt. In jedem Durchlauf der for-Schleife werden 80% der Zellen aus dem Datensatz über die *sample*-Funktion gezogen. “B” repräsentiert im R-Skript die Anzahl an Bootstrap-Läufen, die durchgeführt werden sollen. Für diese Arbeit wurden 100 Bootstrap-Läufe genutzt. Über “n” wurde festgelegt, wie viel Prozent der Zellen gesampelt werden sollen. Size=n in der *sample*-Funktion zeigt damit an, welcher Prozentsatz des Datensatzes für das Bootstrapping benutzt wird. Für das Bootstrapping nutze man die Vorgabe “1: ncol(active)”, um aus den Spalten des active-Objektes (“*Seura*t”-Objekt mit den Daten der COVID 19-Patienten beschrieben in 2.1) die Zellen für das Sampling zu nutzen. Set.seed() musste für das Bootstrapping auf NULL gesetzt werden, da “*Seurats*” FindClusters()- und FindMarkers()-Funktionen random seeds setzen. Das erschwert das Bootstrapping, da man über die random seeds nach jedem neuen Start des R-Skriptes dasselbe Ergebnis erhält. Um das zu vermeiden, setzte ich set.seed auf *NULL*, um bei jedem Bootstrap-Lauf alle bestehenden Seeds zu löschen. Mit jedem Bootstrap-Durchlauf werden die scRNA-seq-Daten aus dem Datensatz mit den Zellen der COVID 19-Patienten mit SingleR auf Kontaminierungen mit fremden Zellen überprüft (siehe Teilkapitel 2.3) und über “*Seurat*” wie in Teilkapitel 2.2 beschrieben analysiert. Die Daten aus den 100 Bootstrap-Durchläufen wurden unter “iteration” in Form von Listen abgespeichert. Das entstandene Objekt mit dem Namen “iteration” war eine Liste bestehend aus weiteren Listen, die jeweils einen Bootstrap-Durchlauf repräsentierten. Bei 100 Bootstrap-Durchläufen bestand die Liste deshalb aus 100 weiteren Listen. Für die weitere Analyse mussten diese Listen in eine große Matrix konvertiert werden. Über eine for-Schleife und die cbind()-und do.call()-Funktionen wurden die Listen auf eine große Matrix reduziert. Zwei zusätzliche Spalten, “i” und “j”, wurden eingeführt. “i” repräsentiert den Bootstrap-Durchlauf, aus dem ein Feature kommt. “j” repräsentiert das Cluster, das dem Feature in dem Bootstrap-Durchlauf i zugeordnet wurde. Die Bootstrap-Daten wurden dann über write.table() als eine csv-Datei unter dem Dateinamen “101Bootstraping220622.csv” exportiert.

### 2.5 Data.table

Der Vergleich zwischen dem Gesamtdatensatz und den Bootstrap-Durchläufen erforderte, dass die Markergene jedes Clusters eines Durchlaufs im Bootstrap mit den Markergenen in jedem Cluster des Gesamtdatensatzes verglichen werden. Dafür wurde das Objekt mit den Markergenen des Gesamtdatensatzes in eine Liste konvertiert und über eine for-Schleife wurde die Liste auf eine Matrix reduziert, wobei eine neue Spalte “i” eingeführt werden musste. Die Spalte “i” repräsentiert das Cluster, das einem Feature im Gesamtdatensatz zugeordnet wurde. Da die Datei mit dem Bootstrap-Durchläufen nicht in demselben R-Skript wie die Gesamtdatensätze erstellt wurde, musste die “101Bootstraping220622.csv”-Datei in das R-Skript mit dem Gesamtdatensatz importiert werden. Der Vergleich zwischen den Clustern der Bootstrap-Durchläufe und den Clustern aus dem Gesamtsatz erforderte die Nutzung des data.table-Pakets (Version 1.14.2). Das Objekt “ClusterMarkers”, welches die Daten aus dem Gesamtdatensatz enthält, wurde in ein data.table-Objekt mit dem Namen “GDS” (Abkürzung für Gesamtdatensatz) konvertiert. Als Nächstes selektiert man die Gene, die nur ein einziges Mal in allen Clustern auftauchen und die einen P-value haben, der kleiner als 0.05 ist. Auch die importierten Daten mit den Bootstrap-Durchläufen wurden in ein data.table-Objekt mit dem Namen “Boot” konvertiert. Danach mappte man die Gennamen aus dem Objekt “GDS” gegen die Gennamen aus dem “Boot”-Objekt und fügt eine neue Spalte “RefCluster” in “Boot” hinzu. “RefCluster” enthält neue Clusternummern. Diese Clusternummern entsprechen den Clustern aus dem Gesamtdatensatz (GDS). Wenn ein Markergen in “Boot” statt einer Zahl ein “NA” in der RefCluster-Spalte enthält, dann wurde das Markergen nicht in GDS gefunden. Das bedeutet, dass das Markergen in der Referenz (GDS) nicht auftaucht, weil das Gen entweder nicht signifikant oder einzigartig ist. Solche Markergene wurden über die Funktion na.omit() entfernt, damit sie nicht die weitere Analyse beeinflussen können. Um zu bestimmen, wie oft ein bestimmtes “RefCluster” mit einem bestimmten Cluster aus den Bootsrapping-Durchläufen pro Bootstrap-Durchlauf auftaucht, wurde eine neue Spalte “Clustercounts” in “Boot” eingeführt. “Clustercounts” zeigt an, wie oft ein Cluster in “Boot” einem bestimmten “RefCluster” pro Bootstrap-Durchlauf zugeordnet wurde. Ein neues Objekt mit dem Namen “ClusterCounts” , welches die Daten aus dem “Boot”-Objekt mit den neuen Spalten “RefCluster” und “Clustercounts” enthält, wurde angelegt. Das Objekt “ClusterCounts” enthält die Gennamen, die in den 100 Bootstrap-Durchläufen gefunden wurden, zusammen mit den p-Werten, den durchschnittlichen Log-Fold-Changes und den adjustierten p-Werten. Des Weiteren enthält es eine Spalte, die anzeigt, in welchem Bootstrap-Durchlauf das jeweilige Feature gefunden wurde (“BootstrapRun”) und welchem Cluster das Feature im jeweiligen Bootstrap-Durchlauf zugeordnet wurde (“Cluster). Die Spalte “RefCluster” entspricht dem Cluster aus der Referenz “GDS”.

# 3. Ergebnisse

Nachdem die Grundlagen für scRNA-seq im Kapitel 1. Einleitung und die genutzten Methoden und Daten in Kapitel 2 dargestellt wurden, behandelt Kapitel 3 die Ergebnisse dieser Arbeit. Wie aus dem Kapitel “Methoden und Daten” zu entnehmen ist, wurden für diese Arbeit zwei große Analysen durchgeführt:

1) Die Analyse wurde zunächst mit etablierten Methoden ohne Bootstrapping durchgeführt. Die Ergebnisse dieser Standardanalyse wurden als Referenz für die Bootstrap-Analyse genutzt.

2) In der Boostrap-Analyse wurden die Markergene jedes Clusters in jedem Bootstrap-Durchlaufes mit den Markergenen aus dem Referenzdatensatz verglichen.

Aus diesem Grund lässt sich folgendes Kapitel auch in zwei große Bereiche einteilen. Zuerst werden die Ergebnisse ohne Bootstrap-Analyse dargestellt. Hier werden die Ergebnisse des Datensatzes besprochen, die für den Vergleich von Markergenen aus dem Bootstrap-Durchläufen mit Markergenen aus einer Referenz benutzt wurden. Danach werden die Ergebnisse der Bootstrap-Analyse besprochen. Dazu gehören die Ergebnisse aus den Bootstrap-Durchläufen, aber auch der Vergleich der Markergene aus den Bootstrap-Analysen mit den Markergenen des Gesamtdatensatzes/ Referenzdatensatzes.

## 3.1. Ergebnisse des Gesamtdatensatzes ohne Bootstrap-Analyse

Die Qualitätskontrolle mit SingleR (beschrieben in Teilkapitel 2.3) ergab, dass 92.22 % (9.268 Zellen) der identifizierten Zellen im Datensatz mit den Zellen der COVID-19 Patienten B-Zellen sind. 7.78 % (782 Zellen) der Zellen im Datensatz wurden als Nicht-B-Zellen identifiziert und wurden für die weitere Analyse entfernt. Die Analyse ergab, dass der Datensatz der COVID-19 Patienten 7 Cluster enthält. Insgesamt gibt es 724 Markergene. Markergene können in mehreren Clustern auftauchen. Über alle 7 Cluster verteilt findet man deshalb insgesamt 2.314 Gene. **Tabelle 1** zeigt die Verteilung der Gene über alle 7 Cluster.

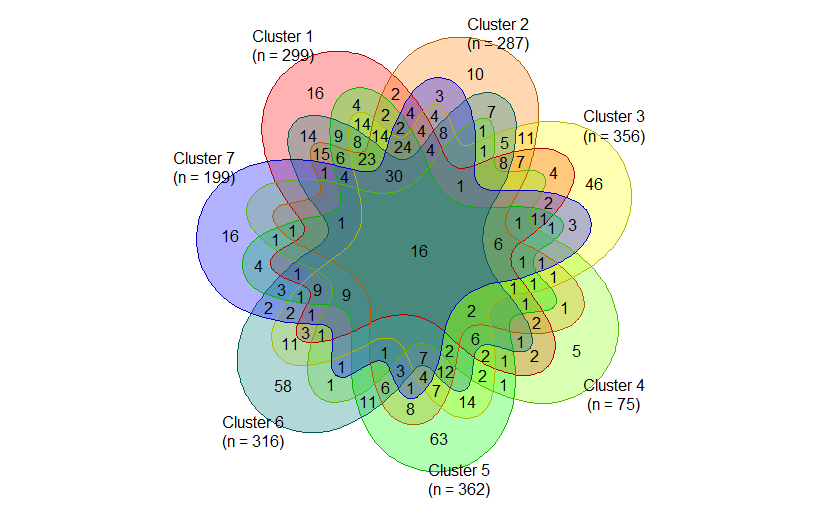
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Cluster im Gesamt-datensatz** | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **Anzahl an Markergenen im Cluster** | 322 | 324 | 381 | 102 | 404 | 410 | 371 |

***Tabelle 1****: Anzahl an Clustern im Gesamtdatensatz, der später als Referenzdatensatz für den Vergleich mit Bootstrap-Durchläufen benutzt wurde. In der zweiten Reihe steht die Anzahl an gefundenen Genen pro Cluster. Markergene können in mehreren Clustern auftauchen. CD69 taucht zum Beispiel in allen Clustern einmal auf (siebenmal insgesamt). Die Anzahl an gefundenen Markergenen pro Cluster ist deshalb höher als die Anzahl an einzigartigen Markergenen.*  
  
Von den 724 Markergenen sind 214 Markergene einzigartig (d.h. Sie tauchen nur in einem Cluster auf) und signifikant (haben einen p-Wert kleiner 0.05). Diese 214 Markergene sind ebenfalls auf die 7 Cluster verteilt. **Tabelle 2** zeigt die Verteilung der 214 Markergene auf die 7 Cluster.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Cluster** | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **Anzahl an Markergenen im Cluster** | 16 | 10 | 46 | 5 | 63 | 58 | 16 |

***Tabelle 2****: Anzahl an Clustern im Gesamtdatensatz, nachdem die einzigartigen und signifikanten (p<0.05) Markergene selektiert wurden. Die zweite Reihe zeigt die Anzahl an gefundenen Markergenen pro Cluster. Durch die Selektion tauchen die Markergene nur in einem Cluster auf. Markergene tauchen also nicht mehr in mehreren Clustern auf.*

**Abbildung 3** zeigt, wie viele signifikante Markergene (p-Wert kleiner 0.05) im Bootstrapdatensatz pro Cluster vorhanden sind. Die Cluster besitzen 75 (Cluster 4) bis 362 Markerge (Cluster 5). 16 Markergene (siehe Zentrum) kommen in jedem der sieben Cluster vor. Man sieht, dass der Anteil an einzigartigen Markergenen pro Cluster (Markergene, die nur in einem Cluster auftauchen) den Werten in Tabelle 2 entspricht. Das heißt, dass fünf (Cluster 4) bis 63 Markergene (Cluster 5) einzigartig sind (kommen nur in einem Cluster vor). Diese einzigartigen Markergene wurden als Referenz für die weitere Analyse verwendet.

***Abbildung 3:*** *Venn-Diagramme zeigen die Markergene pro Cluster im Bootstrapdatensatz. Anzahl an Markergenen pro Cluster unter Clusternummer angegeben (“n=x”). Um die Interpretation der Abbildung zu erleichtern, soll hier Cluster 1 als Beispiel dienen. Cluster 1 hat 299 Markergene, nachdem die Markergene mit einem p-Wert kleiner 0.05 selektiert wurden. Das heißt, im Vergleich zu Tabelle 1 wurden 23 Markergene mit einem p-Wert höher als 0.05 rausselektiert. Von diesen 299 Markergenen sind 16 Markergene in Cluster 1 einzigartig, weil Sie in keinem anderen Cluster vorzufinden sind. Diese 16 Markergene aus Cluster 1 wurden als Referenz für weitere Analysen verwendet.*

Als Nächstes wurden die 5 Markergene mit den geringsten p-Werten pro Cluster extrahiert. Das heißt, aus den Datensatz mit den 214 einzigartigen Markergenen wurden pro Cluster 5 Markergene mit den niedrigsten p-Werten selektiert. **Tabelle 3** zeigt die 5 Markergene, die pro Cluster selektiert wurden.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Cluster-nummer** | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **Top 5 signifikantesten Marker-gene pro Cluster** | ZFYVE16  (100) | CCR6  (100) | CCDC138  (100) | IGHV4-34  (100) | MAST4  (100) | IGHV6-1 (96) | BIN2  (97) |
| HIST1H2BF  (99) | PDIA6  (89) | GLA  (100) | IGHV3-73  (99) | SEMA3D  (100) | IGHV4-4 (76) | SERPINB1 (96) |
| JUP  (96) | PAXBP1  (89) | KLHL8  (84) | IGHV1-69  (93) | COL4A4  (100) | IGHV3-11 (65) | PMAIP1 (91) |
| CFAP298  (95) | BACE2  (69) | TTC39B  (61) | IGHV2-5 (93) | ESR2  (100) | CAMKK1 (13) | LITAF  (89) |
| GPATCH11  (92) | ZC3H6  (54) | AC087623.2 (47) | CD96  (32) | ENOSF1  (99) | HIST2H2AA4  (10) | IGHV3-64D (89) |

***Tabelle 3:*** *Die 5 Markergene mit den niedrigsten p-Werten pro Cluster von den 214 einzigartigen Markergenen. Die Zahl in der Klammer nach dem Gennamen zeigt an, in wie vielen Bootstrap-Durchläufen das Markergen gefunden wurde. Insgesamt wurden 100 Bootstrap-Durchläufe durchgeführt.*

## 3.2. Ergebnisse der Bootstrap-Analyse.

Die 100 Bootstrap-Durchläufe enthalten insgesamt 207.079 Gene über alle Bootstrap-Durchläufe und alle Cluster. Auch bei der Bootstrap-Analyse können Markergene in mehreren Clustern auftauchen. 992 Markergene wurden in allen Bootstrap-Durchläufen entdeckt. Die Anzahl an Clustern variiert je nach Bootstrap-Durchlauf. Das heißt, es werden nicht immer 7 Cluster wie beim Gesamtdatensatz (siehe Teilkapitel 3.1) gefunden. **Tabelle 4** zeigt die Verteilung der gefundenen Cluster pro Bootstrap-Durchlauf. Man sieht, dass zwischen 5 bis 8 Cluster in den 100 Bootstrap-Durchläufen gefunden wurden. 68 Durchläufe besaßen nur 6 Cluster, 26 Durchläufe hatten jeweils 7 Cluster, 5 Durchläufe 5 und 3 der 100 Bootstrap-Durchläufe hatten 8 Cluster.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cluster** | 5 | 6 | 7 | 8 |
| **Anzahl an Bootstrap-Durchläufen mit derselben Clusteranzahl** | 5 | 68 | 26 | 3 |

***Tabelle 4:*** *Anzahl an gefundenen Clustern. Die zweite Reihe gibt an, wie viele der 100 Bootstrap-Durchläufe die jeweilige Anzahl an Clustern aufwiesen.*

## 3.3. Ergebnisse nach Abgleich der Bootstrap-Daten mit dem Referenzdatensatz

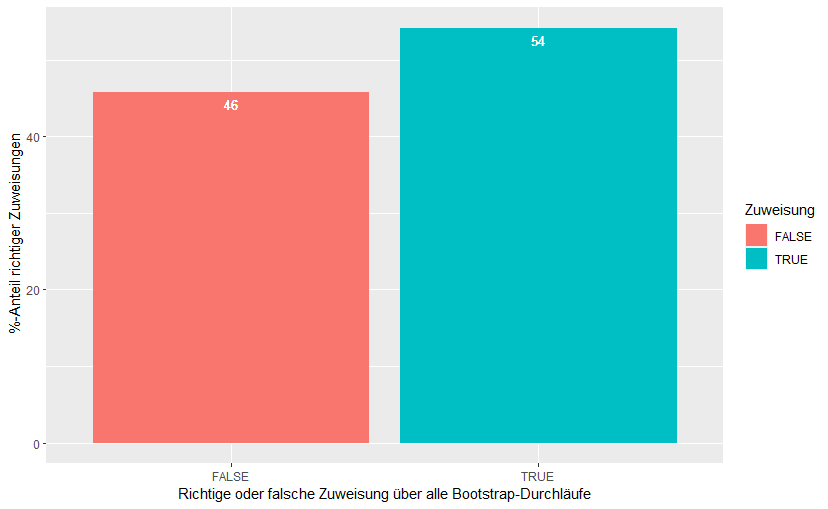
Nachdem die einzigartigen und signifikanten Markergene aus dem Gesamtdatensatz mit den Genen aus der Bootstrap-Analyse verglichen und alle nicht im Gesamtdatensatz auftauchenden Markergene aus den Daten der Bootstrap-Analyse entfernt wurden, blieben nur 23.760 Gene übrig (von den 207.079 beschrieben in Teilkapitel 3.2). Die Anzahl an Markergenen pro Cluster und pro Bootstrap-Durchlauf folgen keiner Normalverteilung. **Tabelle 5** zeigt aus diesem Grund den Medianwert an Markergenen pro Cluster mit dem Minimal- und Maximalwert an gefundenen Werten in einem der Cluster.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Cluster** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| **Median** | 21 | 25 | 48 | 8 | 71.5 | 47 | 47 | 47 |
| **Max** | 27 | 56 | 55 | 71 | 83 | 76 | 70 | 49 |
| ***Min*** | *14* | *15* | *19* | *2* | *5* | *22* | *40* | *42* |

***Tabelle 5:*** *Anzahl an Clustern im Bootstrap-Datensatz, nachdem die einzigartigen signifikanten Markergene aus dem Gesamtdatensatz mit den Genen aus der Bootstrap-Analyse verglichen und alle nicht im Gesamtdatensatz auftauchenden Gene aus den Daten der Bootstrap-Analyse entfernt wurden. Pro Cluster ist der Medianwert, der Minimal- und Maximalwert an Markergenen angegeben. Um die Interpretation der Tabelle zu erleichtern, wird Cluster 1 als Beispiel genutzt. Cluster 1 taucht in der Bootstrap-Datei in jedem Bootstrap-Durchlauf auf. Für diese Arbeit wurde berechnet, wie viele Markergene pro Bootstrap-Durchlauf einem bestimmten Cluster im Bootstrap zugeordnet werden. Im Bootstrap-Durchlauf 1 gibt es zum Beispiel 19 Markergene, die Cluster 1 zugeordnet wurden. Pro Bootstrap-Durchlauf wurde jetzt der Medianwert, Minimal- und Maximalwert an gefundenen Markergenen pro Bootstrap-Cluster bestimmt. Für Cluster 1 bedeutet das, dass Cluster 1 in 50 % der Bootstrap-Durchläufe zwischen 14 und 21 Markergene besitzt und in 50 % der Bootstrap-Durchläufe hat es zwischen 21 und 27 Markergene.*

### 3.3.1. Prozentsatz an richtigen Zuordnungen über alle Bootstrap-Durchläufe

Wie in Kapitel 3.3 beschrieben, hat der Datensatz der Bootstrap-Analyse 23.760 Gene über alle 100 Bootstrap-Durchläufe verteilt, nachdem alle nicht im Gesamtdatensatz auftauchenden Markergene aus den Daten der Bootstrap-Analyse entfernt wurden. Für die Analyse der Stabilität des scRNA-seq-Verfahrens wurde bestimmt, wie oft die Markergene jedes Clusters eines Durchlaufes im Bootstrap mit den Markergenen in jedem Cluster eines Gesamtdatensatzes, welches als Referenz dienen soll, übereinstimmen. Zuerst wurde die Anzahl an richtigen und falschen Zuweisungen über alle 100 Bootstrap-Durchläufe bestimmt. **Abbildung 4** zeigt den Prozentsatz an richtigen Zuweisungen über alle 100 Bootstrap-Durchläufe.

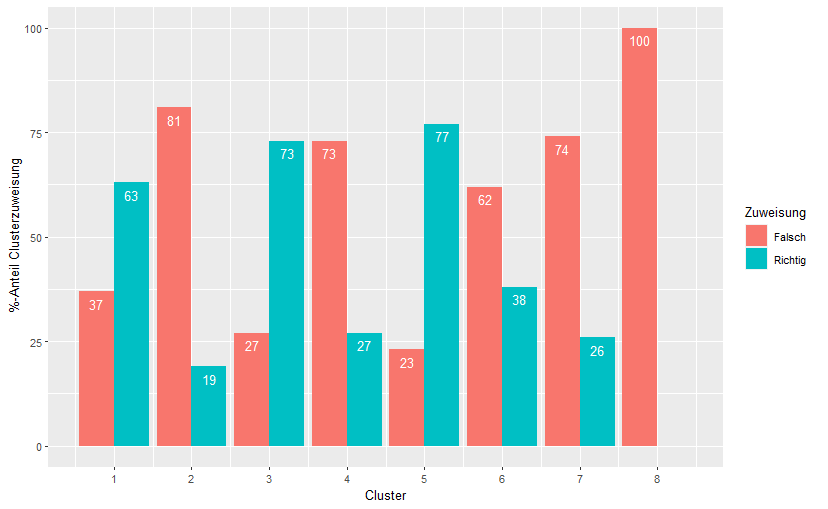


***Abbildung 4****: Prozent-Anteil an richtigen und falschen Zuweisungen für alle 23760 Markergene in den 100 Bootstrap-Durchläufen. Prozentsatz an richtigen Zuweisungen auf der Y-Achse. Markergene, dessen Cluster in der Bootstrap-Datei mit dem Cluster aus dem Referenzdatensatz übereinstimmen, werden dem Wert “TRUE” zugewiesen, um zu zeigen, dass die Zuweisung richtig ist. Markergene, dessen Cluster in der Bootstrap-Datei nicht mit dem Cluster aus der Referenz übereinstimmen, haben den Wert “FALSE” bekommen. Der Prozentsatz an richtigen (TRUE) und falschen (FALSE) Zuweisungen wurde bestimmt und in Form eines Balkendiagramms dargestellt. Ein Balken (grüner Balken, rechts) repräsentiert die Markergene mit richtigen Zuweisungen und der rote Balken (linker Balken) zeigt den Prozentsatz an falsch zugewiesenen Markergenen über alle Bootstrap-Durchläufe.*

Die 100 Bootstrap-Durchläufe enthalten insgesamt 23.760 Markergene. Von diesen 23.760 Markergenen haben etwa 54 % (12.878) die richtige Zuweisung und 45 % (10.882) die falsche Zuweisung. Ein Markergen ist richtig zugewiesen, wenn das Cluster in der Bootstrap-Datei dieselbe Clusternummer hat wie das Cluster aus dem Referenzdatensatz. Es ist falsch zugewiesen, wenn es verschiedene Clusternummern hat.

### 3.3.2. Prozentsatz an richtigen Zuordnungen pro Cluster

In dieser Arbeit wurden nicht nur die richtig und falsch zugewiesenen Gene in allen 100 Bootstrap-Durchläufen bestimmt, sondern es wurden auch die richtigen und falschen Zuweisungen auf Cluster-Ebene betrachtet. Teilkapitel 3.3.1 behandelt den Anteil richtig und falsch zugewiesenen Markergenen über alle 100 Bootstrap-Durchläufe. Das heißt, es ignoriert die Zuordnung der Markergene in bestimmte Clustergruppen und betrachtet nur, wie hoch der Anteil an richtigen und falschen Zuordnungen ist. Um konkrete Aussagen über die Stabilität des scRNA-seq-Verfahrens machen zu können, ist eine Betrachtung auf Cluster-Ebene unerlässlich. Für diesen Zweck wurden alle Markergene, die in allen Bootstrap-Durchläufen einem bestimmten Cluster zugeordnet wurden, zusammengefasst und der Prozentsatz an richtigen und falschen Zuweisungen pro Cluster bestimmt. **Abbildung 5** zeigt, wie hoch der Prozentsatz an richtig und falsch zugeordneten Markergenen pro Cluster ist.

***Abbildung 5:*** *Balkendiagramm an richtig und falsch zugeordneten Markergenen pro Cluster im Bootstrapdatensatz. Die X-Achse zeigt die 8 Cluster, die im Bootstrapdatensatz in den 100 Bootstrap-Durchläufen gefunden wurden. Jedes Cluster hat zwei Balken. Der grüne Balken zeigt an, wie hoch der %-Anteil an richtig zugeordneten Markergenen ist und der rote Balken repräsentiert den Anteil an falsch zugeordneten Markergenen in dem Cluster. Der Prozent-Anteil an Zuweisungen auf der Y-Achse reicht von 0 bis 100 mit Prozent als Einheit. Um die Interpretation der Abbildung zu erleichtern, soll Cluster 1 als Beispiel dienen. In allen 100 Bootstrap-Durchläufen gibt es Markergene, die Cluster 1 zugeordnet werden. Unabhängig davon in welchem Bootstrap-Durchlauf das Markergen gefunden wurde, wurde die Anzahl an Markergenen pro Cluster bestimmt und der Prozentsatz an richtigen und falschen Zuweisungen bestimmt. Für Cluster 1 bedeutet das, dass 63 % der Markergene in allen Bootstrap-Durchläufen, die während der Bootstrap-Analyse dem Cluster 1 zugeordnet wurden, mit dem Cluster aus dem Referenzdatensatz übereinstimmen und 37 % der Markergene mit dem Cluster 1 stimmen nicht überein. Die Prozentsätze aus dieser Abbildung beruhen auf der* ***Tabelle 6****.*

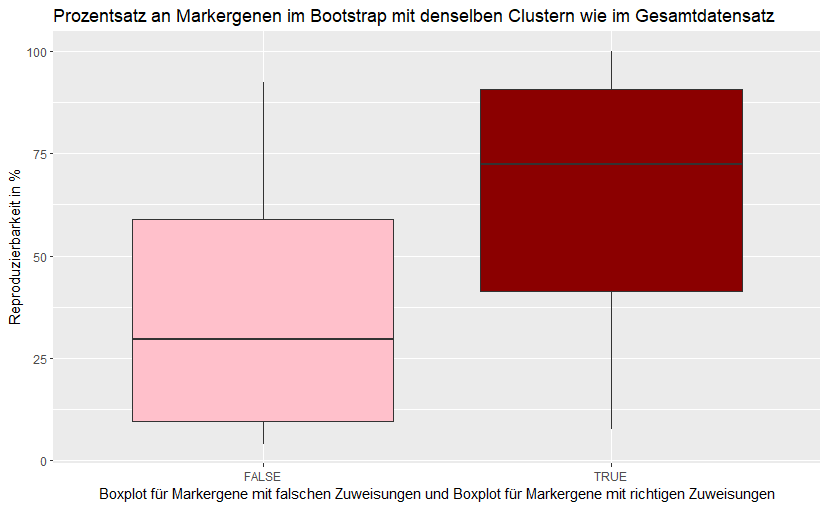
Betrachtet man den Prozentsatz und den Anteil an richtig und falsch zugewiesenen Markergenen auf Clusterebene, so fällt der hohe Anteil an Clustern mit größtenteils falsch zugewiesenen Markergenen auf. Nur 3 der 8 Cluster haben mehr richtig zugeordnete Markergene als falsch zugeordnete Markergene. Diese Cluster sind Cluster 1, 3 und 5. So wurden zum Beispiel 63 % (1.317) der Markergene in Cluster 1 dem richtigen Cluster während des Bootstrapping zugeordnet und 37 % (777) dem falschen Cluster. 73 % (3.194) der Markergene in Cluster 3 wurden dem richtigen Cluster zugeordnet und 27 % (1.169) dem Falschem. Cluster 5 hat mit 77 % (5.317) den höchsten Prozentsatz an richtig zugeordneten Markergenen, während nur 23 % (1.593) der Markergene in Cluster 5 falsch zugewiesen wurden. Die Cluster, die einen höheren Anteil an falsch als an richtig zugewiesenen Markergenen besitzen, sind Cluster 2, 4, 6, 7 und 8. Hier fällt vor allem Cluster 8 auf, weil Cluster 8 mit 100 % (138) nur falsch zugewiesene Markergene besitzt. **Tabelle 6** enthält die Anzahl an richtig und falsch zugewiesenen Markergenen pro Cluster, sowie die Anzahl an Markergenen in dem Cluster (in allen 100 Bootstrap-Durchläufen).

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Cluster** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| **Richtig zugewiesene**  **Marker-gene pro Cluster** | 1.317 | 560 | 3.194 | 273 | 5.317 | 1.853 | 364 | 0 |
| **Falsch zugewiesene Marker-gene pro Cluster** | 777 | 2.372 | 1.169 | 726 | 1.593 | 3.087 | 1.020 | 138 |
| **Anzahl an Marker-Genen pro Cluster** | 2.094 | 2.932 | 4.363 | 999 | 6.910 | 4.940 | 1.384 | 138 |

***Tabelle 6****: Clusternummer des Bootstrapdatensatz in der ersten Reihe der Tabelle. Gefolgt von Anzahl an richtig zugewiesenen Markergenen im jeweiligen Cluster in der zweiten Reihe. Die Anzahl an falsch zugewiesenen Markergenen im jeweiligen Cluster und die Anzahl an Markergenen insgesamt im jeweiligen Cluster.*

### 3.3.3. Anteil an richtigen Zuweisungen auf Gen-Ebene

Aus Teilkapitel 3.1. ist bereits bekannt, dass 214 einzigartige Markergene im Gesamtdatensatz verfügbar sind. Da wir den Bootstrapdatensatz mit dem Gesamtdatensatz verglichen haben und alle nicht im Gesamtdatensatz auftauchenden Markergene aus dem Bootstrapdatensatz entfernt haben, besitzt der Bootstrapdatensatz dieselben 214 einzigartigen Markergene. Um die Reproduzierbarkeit auf Gen-Ebene zu überprüfen, wurde pro Gen ausgerechnet, wie oft ein Markergen im Bootstrap demselben Cluster zugeordnet wird wie im Gesamtdatensatz. Umgekehrt wurde auch der Prozentsatz an Malen ausgerechnet, bei dem das Cluster für ein Markergen beim Bootstrapping einem anderen Cluster als im Gesamtdatensatz entspricht (**Abbildung 6**).

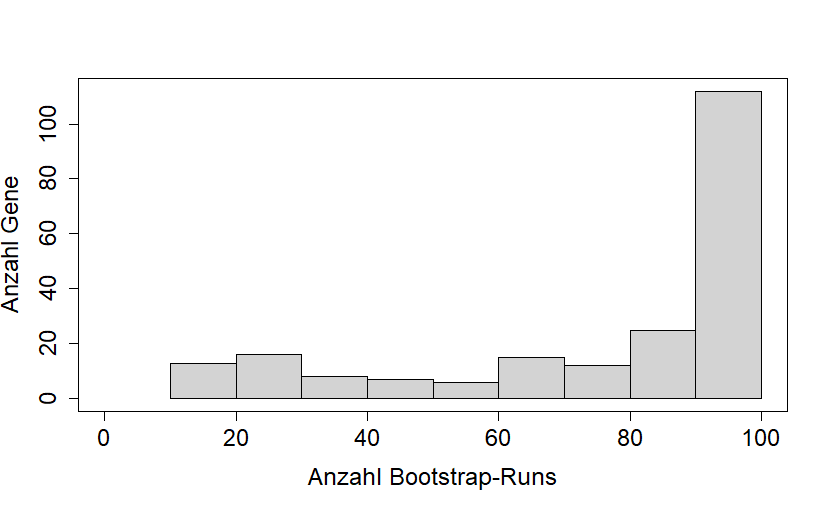


***Abbildung 6:*** *Boxplot für den Prozentsatz an Markergenen im Bootstrapdatensatz mit denselben Clustern wie im Gesamtdatensatz. Y-Achse repräsentiert die Produzierbarkeit in Prozent. Das heißt, es zeigt an, wie oft die Markergene demselben Cluster im Bootstrapdatensatz wie im Gesamtdatensatz zugeordnet wurden. Der rechte Boxplot zeigt an, wie oft ein Markergene richtig zugeordnet wurden, während der linke Boxplot in der Abbildung den %-Anteil an falschen Zuordnungen darstellt. Die Boxplots sind folgendermaßen zu interpretieren: Der Median der Markergene, die in einem bestimmten Bootstrap-Durchlauf falsch zugewiesen wurden, ist ca. 30 %. Der Minimalwert beträgt ca. 4 % und der Maximalwert ca. 92 %. Das heißt, dass 50 % der Markergene, die in dem Bootstrapdatensatz falsch zugeordnet wurden, eine Fehlzuweisung zwischen 4 bis 30 % über alle 100 Bootstrap-Durchläufe aufwiesen. Unter den anderen 50 % der Markergene waren Markergene dabei, die eine Fehlzuweisung von 30 bis 92 % aufwiesen.*

**Abbildung 6** zeigt, dass 50 % der 214 Markergene in den 100 Bootstrap-Durchläufen in ca. 8 bis 70 % der Fälle dem richtigen Cluster zugeteilt wurden. Die anderen 50 % der Markergene werden in 70 bis 100 % der Bootstrap-Läufe, in denen diese Markergene auftauchen, dem richtigen Cluster zugeordnet. Umgekehrt haben 50 % der falsch zugewiesenen Markergene eine Fehlerquote unter 30 % und die anderen 50 % haben eine Fehlerquote über 30 %.

### 3.3.4. Anteil an Markergenen pro Bootstrap-Durchlauf

Nachdem der Prozentsatz an richtigen Zuweisungen über alle Bootstrap-Durchläufe (3.3.1.), auf Cluster (3.3.2.) - und Gen-Ebene (3.3.3.) betrachtet wurde, war es wichtig herauszufinden, in wie vielen der 100 Bootstrap-Durchläufe die 214 Markergene auftauchen. Das wurde in **Abbildung 7** dargestellt. Dem Histogramm ist zu entnehmen, dass 117 Markergene in mehr als 90 Bootstrap-Durchläufen auftauchen. Das bedeutet, dass etwa 54 % der Markergene in mehr als 90 Bootstrap-Durchläufen gefunden werden. 55 Markergene tauchen in 50 bis 90 Durchläufen auf und 42 Markergene tauchen in weniger als 50 Durchläufen auf. Das heißt 46 % der Markergene tauchen in weniger als 90 Bootstrap-Durchläufen auf und etwa 20 % der Markergene tauchen in weniger als 50 % der Durchläufe auf. 13 Markergene tauchen in weniger als 20 Durchläufen auf.



***Abbildung 7****: Histogramm mit Anzahl an gefundenen Genen (X-Achse) pro Bootstrap-Run (Y-Achse). Histogramm zeigt, wie viele Markergene in 100 Bootstrap-Durchläufen gefunden werden.*

# 4. Diskussion

Einzelzell-RNA-Sequenzierungstechnologien ermöglichen die Analyse von Transkripten auf der Ebene einzelner Zellen. Damit können Genexpressionsunterschiede zwischen einzelnen Zellen genutzt werden, um Zelltypen zu identifizieren und zu charakterisieren oder um den Einfluss einer Krankheit auf die Genexpression einzelner Zellen zu analysieren. Die hohen Kosten der Einzelzell-RNA-Sequenzierungsverfahren sorgen aber dafür, dass oft auf technische Replikationen verzichtet wird. Somit gibt es wenige wissenschaftliche Arbeiten, die sich mit der Robustheit und Reproduzierbarkeit von Einzelzell-RNA-Sequenzierungsdaten auseinandergesetzt haben. In dieser Arbeit wurden Einzelzell-RNA-Sequenzierungsdaten aus der Datenbank einem Bootstrapping unterzogen, um die Robustheit und die Reproduzierbarkeit der Daten zu testen. Beim Bootstrapping werden wiederholt Teilmengen des Datensatzes analysiert und die Ergebnisse der einzelnen Durchläufe in einer Gesamtschau betrachtet. Signifikante und nur in einem Cluster auftretende Markergene aus dem Gesamtdatensatz wurden als Referenz genutzt, um zu prüfen, wie oft die Markergene jedes Clusters eines Durchlaufes im Bootstrap mit den Markergenen in jedem Cluster der Referenz übereinstimmen.

Aus Teilkapitel 3.3.1. ist bekannt, dass nur 54 % (12.878 Gene) der 23.760 untersuchten Markergene den korrekten Clustern im Bootstrapping zugewiesen wurden. Auf Cluster-Ebene haben nur Cluster 1, 3 und 5 mehr richtige als falsche Zuweisungen (**siehe Abbildung 5**). Ein Grund für den geringen Prozentsatz an richtigen Zuweisungen auf Cluster-Ebene und für alle Markergene ist die Anzahl an gefunden Cluster pro Bootstrap-Durchlauf. Im Gesamtdatensatz, welcher als Referenz genutzt wurde, wurden 7 Cluster gefunden (in Teilkapitel 3.1 behandelt). Der Anteil an Clustern und der Anteil an Markergenen pro Cluster im Referenzdatensatz ist in **Tabelle 2** angegeben. Unter optimalen Bedingungen würde das Bootstrapping eine ähnliche Anzahl an Markergenen pro Cluster und die gleiche Clusteranzahl finden, aber **Tabelle 4** zeigt, dass nur 26 der 100 durchgeführten Bootstrap-Durchläufe 7 Cluster aufwiesen. Das heißt, nur 26 % der Bootstrap-Durchläufe wiesen die korrekte Anzahl an Clustern auf. Die Mehrheit (68 %) der Bootstrap-Durchläufe weisen nur 6 Cluster auf und einige Durchläufe haben 5 (5%) oder 8 Cluster (3%). Das hat zur Folge, dass einige Markergene, die zum Beispiel Cluster 7 zugeordnet werden sollten, einem anderen Cluster zugewiesen werden, weil nicht jeder Bootstrap-Durchlauf 7 Cluster aufweist.

Das erklärt auch, warum Cluster 8 in **Abbildung 5** eine Fehlerquote von 100 % aufweist: es existieren nur 7 Cluster im Referenzdatensatz und damit können die Markergene, die dem Cluster 8 in der Bootstrapping-Datei zugewiesen wurden, nie mit den Clustern im Referenzdatensatz übereinstimmen. Das Beheben dieses Problems könnte auch den Anteil an richtig zugewiesenen Markergenen erhöhen.

Aus **Abbildung 6** ist bereits bekannt, dass 50 % der Markergene in 70 bis 100 % der Bootstrap-Läufe, in denen diese Markergene auftauchen, dem richtigen Cluster zugeordnet werden. Darunter sind auch Markergene, die (Beispiel) Cluster 7 zugeordnet werden sollten, aber in den Bootstrap-Ansätzen, die weniger oder mehr als 7 Cluster aufweisen, einem anderen Cluster zugeordnet werden.

Ein weiteres Problem ist die Anzahl an gefundenen Markergenen pro Bootstrap-Durchlauf. Nur 117 Markergene von den 214 einzigartigen Markergenen tauchen in mehr als 90 Bootstrap-Durchläufen auf. Das heißt, nur 54 % der Markergene im Datensatz können für biologische Interpretationen weiterverwendet werden. 42 Markergene (20%) tauchen in weniger als 50 Bootstrap-Durchläufen auf und sind damit nicht für weitere Interpretationen verwendbar. Um das zu verdeutlichen, sollen hier ein paar Markergene als Beispiel aufgeführt werden.

CAMKK1 (Calcium/calmodulin-dependent protein kinase kinase 1) ist das Gen für eine Calcium/ Calmodulin-abhängige Serin/Threonin-Kinase, die eine große Rolle in den Calcium/ Calmodulin-abhängigen Kinase-Kaskadenwegen spielt (Yano et al. 1998). Dieses Gen wurde als ein signifikantes Gen in Cluster 6 im Gesamtdatensatz gefunden (siehe **Tabelle 3**). Yano et al. (1998) beschrieben die Funktion von CAMKKs als ein Schutzmittel gegen Apoptose. Vireninfektionen wie COVID 19 nutzen Apoptose, um aus der Wirtszelle freigesetzt zu werden (Smatti et al. 2019). Demnach könnte die Expression von CAMKK1 als eine Antwort auf die Virus-Infektion interpretiert werden. In der Bootstrapdatei kommt das Gen aber nur in 13 von 100 Bootstrap-Durchläufen vor. Dieses Gen sollte also nur mit Vorsicht für biologische Interpretationen benutzt werden. Ohne Bootstrapping würde dieses Gen mit einer Standardanalyse zu einer möglichen Fehlinterpretation führen. Weitere Markergene, die in unter 20 Bootstrap-Durchläufen aufgetaucht sind, sind Thioredoxin Domain Containing 5 (TXNDC5) und PR/SET Domain 1 (PRDM1) (nicht in **Tabelle 3** angegeben). TXNDC5 ist eine Protein-Disulfid-Isomerase. Das Produkt dieses Gens fördert die Ausbildung und Neuordnung von Disulfidbrücken und ist damit auch für die korrekte Proteinfaltung wichtig (Wang et al. 2022). Huang et al. (2021) beschrieben die Rolle von TXNDC5 als ein Marker für die Ausbildung von spezifischen Antikörpern gegen die COVID-19-Antigene. TXNDC5 wurde im Gesamtdatensatz Cluster 6 zugewiesen. Die Anwesenheit von TXNDC5 in Cluster 6 deutet daraufhin, dass die Zellen in diesem Cluster TXNDC5 als eine Antwort auf eine Infektion mit COVID 19 exprimieren. Im Bootstrapdatensatz wurde TXNDC5 aber nur in 15 Bootstrap-Durchläufen gefunden. Das kann zu fehlerhaften Schlussfolgerungen über die Anwesenheit und biologische Funktionen von TXNDC5 in Cluster 6 führen. PRDM1 (alternativer Name: BLIMP1) ist ein Repressor, der die Differenzierung von Effektor-B-Zellen kontrolliert (Nutt et al. 2007). Im Gesamtdatensatz wurde PRDM1 Cluster 6 zugewiesen. Daraus könnte man schlussfolgern, dass die Zellen in Cluster 6 terminal differenzierte B-Zellen sind. Im Bootstrapdatensatz taucht PRDM1 nur in 19 Bootstrap-Durchläufen auf. PRDM1 ist demnach nicht stabil im Bootstrapdatensatz und kann zu einer fehlerhaften Schlussfolgerung führen. Umgekehrt gibt es aber auch Gene, die in mehr als 90 Bootstrap-Durchläufen auftauchten und deshalb für eine biologische Interpretation genutzt werden können. Als Beispiel soll hier der C-C Motif Chemokine Receptor 6 (CCR6) dienen. Suan et al. (2017) identifizierten CCR6 als einen Marker für B-Gedächtniszellen. Im Gesamtdatensatz wurde CCR6 Cluster 2 zugwiesen. Daraus kann man schlussfolgern, dass die Zellen in Cluster 2 B-Gedächtniszellen sind. CCR6 taucht im Bootstrapdatensatz in allen Bootstrap-Durchläufen auf und damit ist das Gen stabil. Laut Schulheiß et al. (2021) handelt es sich bei den Zellen in Cluster 2 um B-Gedächtniszellen. Das heißt, die Schlussfolgerung, dass CCR6 auf die Anwesenheit von B-Gedächtniszellen deutet, wird durch die Bootstrap-Analyse unterstützt. …

Diese Arbeit zeigt, dass Bootstrapping Schlussfolgerung über die Stabilität der Analyseergebnisse von Einzelzell-Sequenzierungsdaten ermöglicht (siehe Beispiel CCR6) und damit eine kostengünstige Alternative zu technischen Replikationen. Nachteil von Bootstrapping ist der hohe Rechenaufwand und der Anteil an falschen Clusterzuweisungen, der in dieser Arbeit dokumentiert wurde. Der Anteil an falschen Clusterzuweisungen entsteht durch den Anteil an Clustern pro Bootstrap-Durchlauf. Wie bereits weiter oben beschrieben, werden pro Bootstrap-Durchlauf nicht dieselbe Anzahl an Clustern wie im Gesamtdatensatz gefunden. Weil die Cluster Zellpopulationen definieren sollen, erschwert eine ungleiche Clusteranzahl pro Bootstrap-Durchlauf die Interpretation der Daten. Zukünftige Arbeiten sollten sich deshalb mit der Frage befassen, wie man das Problem mit der ungleichen Anzahl an Clustern pro Durchlauf beheben kann. Auch der Anteil an richtigen Zuweisungen auf Cluster-Ebene sollte in zukünftigen Arbeiten berücksichtigt werden. Auf Cluster-Ebene wurde in dieser Arbeit ein hoher Prozentsatz an falsch zugewiesenen Markergenen gefunden, der nicht allein auf die falsche Anzahl an Clustern pro Durchlauf zurückzuführen ist. Es ist möglich, dass eine Kombination von Markergenen in einem Bootstrap-Durchlauf dasselbe Cluster wiederfindet, aber dem Cluster wird eine andere Nummer zugewiesen. Alternativ kann es sein, dass ein Markergen in einem Bootstrap-Durchlauf mehrmals gefunden wird und mehreren Clustern zugewiesen wird. Diese Annahmen konnten in dieser Arbeit nicht untersucht werden. Es ist empfohlen in weiteren Arbeiten zu bestimmen, wie oft Markergennamen in einem bestimmten Bootstrap-Durchlauf wiedergefunden werden. Eine Anwendung von Bootstrapping auf Einzelzell-Sequenzierungsdaten in zukünftigen Studien kann von Nutzen sein. Bevor man aber Bootstrapping auf große Einzelzell-Sequenzierungsdaten anwendet, sollten die hier dokumentierten Fehlerquellen behoben werden.

Fazit

# 5. Literaturverzeichnis

Macaulay, I. C. & Voet, T. (2014). Single cell genomics: advances and future perspectives. *PLoS genetics 10* (1), e1004126. doi:10.1371/journal.pgen.1004126

Huang, S. (2009). Non-genetic heterogeneity of cells in development: more than just noise. *Development (Cambridge, England) 136* (23), 3853–3862. doi:10.1242/dev.035139

Tang, F., Barbacioru, C., Wang, Y., Nordman, E., Lee, C., Xu, N., Wang, X., Bodeau, J., Tuch, B. B., Siddiqui, A., Lao, K. & Surani, M. A. (2009). mRNA-Seq whole-transcriptome analysis of a single cell. *Nature methods 6* (5), 377–382. doi:10.1038/nmeth.1315

Rivera-Correa, J., Guthmiller, J. J., Vijay, R., Fernandez-Arias, C., Pardo-Ruge, M. A., Gonzalez, S., Butler, N. S. & Rodriguez, A. (2017). Plasmodium DNA-mediated TLR9 activation of T-bet+ B cells contributes to autoimmune anaemia during malaria. *Nature communications 8* (1), 1282. doi:10.1038/s41467-017-01476-6

Biasi, S. de, Meschiari, M., Gibellini, L., Bellinazzi, C., Borella, R., Fidanza, L., Gozzi, L., Iannone, A., Lo Tartaro, D., Mattioli, M., Paolini, A., Menozzi, M., Milić, J., Franceschi, G., Fantini, R., Tonelli, R., Sita, M., Sarti, M., Trenti, T., Brugioni, L., Cicchetti, L., Facchinetti, F., Pietrangelo, A., Clini, E., Girardis, M., Guaraldi, G., Mussini, C. & Cossarizza, A. (2020). Marked T cell activation, senescence, exhaustion and skewing towards TH17 in patients with COVID-19 pneumonia. Nature communications 11 (1), 3434. doi:10.1038/s41467-020-17292-4

Kulkarni, A., Anderson, A. G., Merullo, D. P. & Konopka, G. (2019). Beyond bulk: a review of single cell transcriptomics methodologies and applications. *Current opinion in biotechnology 58,* 129–136. doi:10.1016/j.copbio.2019.03.001

Salomon, R., Kaczorowski, D., Valdes-Mora, F., Nordon, R. E., Neild, A., Farbehi, N., Bartonicek, N. & Gallego-Ortega, D. (2019). Droplet-based single cell RNAseq tools: a practical guide. Lab on a chip 19 (10), 1706-1727. doi:10.1039/c8lc01239c

Zhu, Y. Y., Machleder, E. M., Chenchik, A., Li, R. & Siebert, P. D. (2001). Reverse transcriptase template switching: a SMART approach for full-length cDNA library construction. BioTechniques 30 (4), 892-897. doi:10.2144/01304pf02

Amezquita, R. A., Lun, A. T. L., Becht, E., Carey, V. J., Carpp, L. N., Geistlinger, L., Marini, F., Rue-Albrecht, K., Risso, D., Soneson, C., Waldron, L., Pagès, H., Smith, M. L., Huber, W., Morgan, M., Gottardo, R. & Hicks, S. C. (2020). Orchestrating single-cell analysis with Bioconductor. Nature methods 17 (2), 137-145. doi:10.1038/s41592-019-0654-x

Hao, Y., Hao, S., Andersen-Nissen, E., Mauck, W. M., Zheng, S., Butler, A., Lee, M. J., Wilk, A. J., Darby, C., Zager, M., Hoffman, P., Stoeckius, M., Papalexi, E., Mimitou, E. P., Jain, J., Srivastava, A., Stuart, T., Fleming, L. M., Yeung, B., Rogers, A. J., McElrath, J. M., Blish, C. A., Gottardo, R., Smibert, P. & Satija, R. (2021). Integrated analysis of multimodal single-cell data. Cell 184 (13), 3573-3587.e29. doi:10.1016/j.cell.2021.04.048

R Core Team (2022). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Luecken, M. D. & Theis, F. J. (2019). Current best practices in single-cell RNA-seq analysis: a tutorial. Molecular systems biology 15 (6), e8746. doi:10.15252/msb.20188746

Lun, A. T. L., Bach, K. & Marioni, J. C. (2016). Pooling across cells to normalize single-cell RNA sequencing data with many zero counts. Genome biology 17, 75. doi:10.1186/s13059-016-0947-7

Vieth, B., Ziegenhain, C., Parekh, S., Enard, W. & Hellmann, I. (2017). powsimR: power analysis for bulk and single cell RNA-seq experiments. Bioinformatics (Oxford, England) 33 (21), 3486-3488. doi:10.1093/bioinformatics/btx435

Pearson, K. (1901). LIII. On lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 2 (11), 559-572. doi:10.1080/14786440109462720

van der Maaten, L. & Hinton, G. (2008). Visualizing data using t-SNE.

McInnes, L., Healy, J. & Melville, J. (2018, 09. Februar). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. https://arxiv.org/pdf/1802.03426

Amir, E. D., Davis, K. L., Tadmor, M. D., Simonds, E. F., Levine, J. H., Bendall, S. C., Shenfeld, D. K., Krishnaswamy, S., Nolan, G. P. & Pe'er, D. (2013). viSNE enables visualization of high dimensional single-cell data and reveals phenotypic heterogeneity of leukemia. Nature biotechnology 31 (6), 545-552. doi:10.1038/nbt.2594

Saremi, B., Gusmag, F., Distl, O., Schaarschmidt, F., Metzger, J., Becker, S. & Jung, K. (2022). A comparison of strategies for generating artificial replicates in RNA-seq experiments. Scientific reports 12 (1), 7170. doi:10.1038/s41598-022-11302-9

Efron, B. (1979). Bootstrap Methods: Another Look at the Jackknife. The Annals of Statistics 7 (1). doi:10.1214/aos/1176344552

Saremi, B., Kohls, M., Liebig, P., Siebert, U. & Jung, K. (2021). Measuring reproducibility of virus metagenomics analyses using bootstrap samples from FASTQ-files. Bioinformatics (Oxford, England) 37 (8), 1068-1075. doi:10.1093/bioinformatics/btaa926

Meyer, J. S., Ingersoll, C. G., McDonald, L. L. & Boyce, M. S. (1986). Estimating Uncertainty in Population Growth Rates: Jackknife vs. Bootstrap Techniques. Ecology 67 (5), 1156-1166. doi:10.2307/1938671

Athar, A., Füllgrabe, A., George, N., Iqbal, H., Huerta, L., Ali, A., Snow, C., Fonseca, N. A., Petryszak, R., Papatheodorou, I., Sarkans, U. & Brazma, A. (2019). ArrayExpress update - from bulk to single-cell expression data. Nucleic acids research 47 (D1), D711-D715. doi:10.1093/nar/gky964

Schultheiß, C., Paschold, L., Willscher, E., Simnica, D., Wöstemeier, A., Muscate, F., Wass, M., Eisenmann, S., Dutzmann, J., Keyßer, G., Gagliani, N. & Binder, M. (2021). Maturation trajectories and transcriptional landscape of plasmablasts and autoreactive B cells in COVID-19. iScience 24 (11), 103325. doi:10.1016/j.isci.2021.103325

Aran, D., Looney, A. P., Liu, L., Wu, E., Fong, V., Hsu, A., Chak, S., Naikawadi, R. P., Wolters, P. J., Abate, A. R., Butte, A. J. & Bhattacharya, M. (2019). Reference-based analysis of lung single-cell sequencing reveals a transitional profibrotic macrophage. Nature immunology 20 (2), 163-172. doi:10.1038/s41590-018-0276-y

Mabbott, N. A., Baillie, J. K., Brown, H., Freeman, T. C. & Hume, D. A. (2013). An expression atlas of human primary cells: inference of gene function from coexpression networks. BMC genomics 14, 632. doi:10.1186/1471-2164-14-632

Yano, S.; Tokumitsu, H.; Soderling, T. R. (1998): Calcium promotes cell survival through CaM-K kinase activation of the protein-kinase-B pathway. In: Nature 396 (6711), S. 584–587. DOI: 10.1038/25147.

Smatti, Maria K.; Cyprian, Farhan S.; Nasrallah, Gheyath K.; Al Thani, Asmaa A.; Almishal, Ruba O.; Yassine, Hadi M. (2019): Viruses and Autoimmunity: A Review on the Potential Interaction and Molecular Mechanisms. In: Viruses 11 (8). DOI: 10.3390/v11080762.

Wang, Xueling; Li, Haoran; Chang, Xiaotian (2022): The role and mechanism of TXNDC5 in diseases. In: European journal of medical research 27 (1), S. 145. DOI: 10.1186/s40001-022-00770-4.

Nutt, Stephen L.; Fairfax, Kirsten A.; Kallies, Axel (2007): BLIMP1 guides the fate of effector B and T cells. In: Nature reviews. Immunology 7 (12), S. 923–927. DOI: 10.1038/nri2204.

# 6.Anhang

## 6.1. R-Code für die Bootstrapping-Analyse

001### This code follows a basic pipeline for Single Cell Sequencing Data Analysis   
002### based on the example dataset E-MTAB-11011   
003   
004setwd**(**"A:/RFILES/setwd"**)**   
005   
006   
007#################   
008### Libraries ###   
009#################   
010   
011library**(**"Seurat"**)**   
012library**(**"celldex"**)**   
013library**(**"SingleR"**)**   
014   
015   
016   
017#################   
018### Functions ###   
019#################   
020   
021# function to compute all steps in a basic single cell experiment pipeline   
022scsPipeline **=** **function(**SeuratObj**){**   
023   
024 #########################   
025 ### Feature Selection ###   
026 #########################   
027   
028 SeuratObj **=** FindVariableFeatures**(**SeuratObj, selection.method **=** "vst", nfeatures **=** 2000**)**   
029   
030 ####################   
031 ### Data Scaling ###   
032 ####################   
033   
034 all.genes\_SeuratObj **=** rownames**(**SeuratObj**)**   
035 SeuratObj **=** ScaleData**(**SeuratObj, features **=** all.genes\_SeuratObj**)**   
036   
037 #################################   
038 ### Dimension Reduction (PCA) ###   
039 #################################   
040   
041 SeuratObj **=** RunPCA**(**SeuratObj, features **=** VariableFeatures**(**object **=** SeuratObj**))**   
042   
043 ##################   
044 ### Clustering ###   
045 ##################   
046   
047 SeuratObj **=** FindNeighbors**(**SeuratObj, dims **=** 1**:**10**)**#setzt keinen random.seed.   
048 SeuratObj **=** FindClusters**(**SeuratObj, resolution **=** 0.5**)**#setzt random.seed=0   
049   
050   
051 #########################################   
052 ### Differentially Expressed Features ###   
053 #########################################   
054   
055 clusterList **=** vector**(**mode **=** "list", length **=** length**(**levels**(**SeuratObj**$**seurat\_clusters**)))**   
056   
057 #Loop that finds markers, then stores them in an additional cluster list   
058 **for(**i **in** 1**:**length**(**clusterList**))** clusterList**[[**i**]]** **=** FindMarkers**(**SeuratObj, ident.1 **=** as.numeric**(**levels**(**SeuratObj**$**seurat\_clusters**)[**i**])**, min.pct **=** 0.25**)**   
059 #FindMarkers setzt random.seed=1 als Default   
060   
061   
062   
063 #To return multiple objects, wrap them in a list first   
064 return**(**clusterList**)**   
065**}**   
066   
067#function to remove wrongly assigned cells from a single cell experiment stored as Seurat Object   
068remWC **=** **function(**SeuratObj, regex **=** ".\*B\_cell.\*"**){**   
069   
070 ############################   
071 ### Cell Type Annotation ###   
072 ############################   
073   
074 #active   
075 sce **=** as.SingleCellExperiment**(**SeuratObj**)** #convert Seurat to SingleCellExperiment   
076 se **=** as**(**sce, "SummarizedExperiment"**)** #convert SingleCellExperiment to SummarizedExperiment   
077 rownames**(**se**)** **=** rownames**(**sce**)** #re-add the rownames   
078   
079 # predict cell type   
080 pred **=** SingleR**(**test **=** se, ref **=** hpca.se, assay.type.test**=**1,   
081 labels **=** hpca.se**$**label.main**)**   
082   
083 pred**$**first.labels #show results   
084   
085 #percentage b cells   
086 pB **=** sum**(**grepl**(**".\*B\_cell.\*", pred**$**first.labels**))** **/** length**(**pred**$**first.labels**)** **\*** 100   
087 print**(**paste0**(**"Percentage B-cells: ", round**(**pB, 2**)**, " %"**))**   
088   
089 SeuratObj**$**CellAnnotation **=** pred**$**first.labels   
090   
091 ###################################   
092 ### Removal of wrong cell types ###   
093 ###################################   
094   
095 ca **=** as.vector**(**SeuratObj**$**CellAnnotation**)** #vector with cell annotations 096 indices **=** which**(**grepl**(**regex, ca**))** #indices of cells to keep   
097 SeuratObj **=** SeuratObj**[**,indices**]** #retain cells according to indices   
098   
099 return**(**SeuratObj**)** #return Seurat Object after removal of wrongly assigned cells   
100**}**   
101   
102#function to export the top200 genes for each cluster   
103exT200 **=** **function(**clusterList, name, path **=** getwd**()){**   
104 **for(**i **in** 1**:**length**(**clusterList**)){**   
105 #take the row names from the first 200 rows in clusters unless there's fewer rows available   
106 num **=** ifelse**(**nrow**(**clusterList**[[**i**]])** **<** 200, nrow**(**clusterList**[[**i**]])**, 200**)** 107 top200 **=** rownames**(**clusterList**[[**i**]])[**1**:**num**]**   
108 write.table**(**top200, file **=** paste0**(**path,"/", name, "\_", i, "\_top200.txt"**)**, quote **=** **FALSE**, row.names **=** **FALSE**, col.names **=** **FALSE)**   
109 **}**   
110**}**   
111   
112   
113##################   
114### Data Input ###   
115##################   
116   
117# read files   
118active **=** readRDS**(**"A:/RFILES/E-MTAB-11011 - Single Cell Analysis of B cells in COVID-19 comprising active and recovered disease/.rds-files der Patienten/active-Covid-Kranke/pbmc.active.2.5.3.8\_gex\_and\_vdj.rds"**)**   
119   
120   
121########################################   
122### Remove wrongfully assigned cells ### 123########################################   
124   
125hpca.se **=** celldex**::**HumanPrimaryCellAtlasData**()** #reference data   
126active **=** remWC**(**active**)**   
127   
128######################################## 129###########Bootstrap#################### 130######################################## 131#as.matrix(GetAssayData(active))   
132#n=round(ncol(A)/100\*80)   
133B**=**100   
134iteration**=**vector**(**mode**=**"list",length**=**B**)**   
135n**=**round**(**ncol**(**active**)/**100**\***80**)**   
136start**=**Sys.time**()**   
137**for(**i **in** 1**:**B**){**   
138 set.seed**(NULL)**   
139 s**=**sample**(**1**:**ncol**(**active**)**,size**=**n**)**   
140 D**=**active**[**,s**]**   
141 iteration**[[**i**]]=**scsPipeline**(**D**)**   
142**}**   
143end**=**Sys.time**()**   
144print**(**end**-**start**)**   
145#################################################   
146#########Bootstrap-Daten als csv exportieren##### 147#################################################   
148   
149setwd**(**"A:/RFILES/bootstrap/22062022"**)**   
150   
151**for** **(**i **in** 1**:**length**(**iteration**)){**   
152 **for(**j **in** 1**:**length**(**iteration**[[**i**]])){**   
153 iteration**[[**i**]][[**j**]]=**cbind**(**iteration**[[**i**]][[**j**]]**,i,j**)**   
154 iteration**[[**i**]][[**j**]]=**cbind**(**iteration**[[**i**]][[**j**]]**,rownames**(**iteration**[[**i**]][[**j**]]))**   
155 **}**   
156**}**   
157   
158   
159**for(**i **in** 1**:**length**(**iteration**))**iteration**[[**i**]]=** do.call**(**rbind,iteration**[[**i**]])**   
160result**=**do.call**(**rbind,iteration**)** 161write.table**(**result,"101Bootstraping150822.csv",quote**=FALSE**,sep**=**";"**)**

## 6.2. R-Code für den Gesamtdatensatz und für den Vergleich mit der Bootstrap-Analyse

001### This code follows a basic pipeline for Single Cell Sequencing Data Analysis 002### based on the example dataset E-MTAB-11011 003 004setwd**(**"A:/RFILES/für 25.05.22"**)** 005 006 007################# 008### Libraries ### 009################# 010 011library**(**"Seurat"**)** 012library**(**"celldex"**)** 013library**(**"SingleR"**)** 014library**(**"data.table"**)** 015 016################# 017### Functions ### 018################# 019 020# function to compute all steps in a basic single cell experiment pipeline 021scsPipeline **=** **function(**SeuratObj**){** 022 023 ######################### 024 ### Feature Selection ### 025 #########################   
026   
027 SeuratObj **=** FindVariableFeatures**(**SeuratObj, selection.method **=** "vst", nfeatures **=** 2000**)**   
028   
029 ####################   
030 ### Data Scaling ###   
031 ####################   
032   
033 all.genes\_SeuratObj **=** rownames**(**SeuratObj**)**   
034 SeuratObj **=** ScaleData**(**SeuratObj, features **=** all.genes\_SeuratObj**)**   
035   
036 #################################   
037 ### Dimension Reduction (PCA) ###   
038 #################################   
039   
040 SeuratObj **=** RunPCA**(**SeuratObj, features **=** VariableFeatures**(**object **=** SeuratObj**))**   
041   
042 ##################   
043 ### Clustering ###   
044 ##################   
045   
046 SeuratObj **=** FindNeighbors**(**SeuratObj, dims **=** 1**:**10**)**   
047 SeuratObj **=** FindClusters**(**SeuratObj, resolution **=** 0.5**)**   
048   
049   
050 #########################################   
051 ### Differentially Expressed Features ###   
052 #########################################   
053   
054 clusterList **=** vector**(**mode **=** "list", length **=** length**(**levels**(**SeuratObj**$**seurat\_clusters**)))**   
055   
056 #Loop that finds markers, then stores them in an additional cluster list   
057 **for(**i **in** 1**:**length**(**clusterList**))** clusterList**[[**i**]]** **=** FindMarkers**(**SeuratObj, ident.1 **=** as.numeric**(**levels**(**SeuratObj**$**seurat\_clusters**)[**i**])**, min.pct **=** 0.25**)**   
058   
059   
060 #To return multiple objects, wrap them in a list first   
061 return**(**list**(**SeuratObj, clusterList**))**   
062**}**   
063   
064#function to remove wrongly assigned cells from a single cell experiment stored as Seurat Object   
065remWC **=** **function(**SeuratObj, regex **=** ".\*B\_cell.\*"**){**   
066   
067 ############################   
068 ### Cell Type Annotation ###   
069 ############################   
070   
071 #active   
072 sce **=** as.SingleCellExperiment**(**SeuratObj**)** #convert Seurat to SingleCellExperiment   
073 se **=** as**(**sce, "SummarizedExperiment"**)** #convert SingleCellExperiment to SummarizedExperiment   
074 rownames**(**se**)** **=** rownames**(**sce**)** #re-add the rownames   
075   
076 # predict cell type   
077 pred **=** SingleR**(**test **=** se, ref **=** hpca.se, assay.type.test**=**1,   
078 labels **=** hpca.se**$**label.main**)**   
079   
080 pred**$**first.labels #show results   
081   
082 #percentage b cells   
083 pB **=** sum**(**grepl**(**".\*B\_cell.\*", pred**$**first.labels**))** **/** length**(**pred**$**first.labels**)** **\*** 100   
084 print**(**paste0**(**"Percentage B-cells: ", round**(**pB, 2**)**, " %"**))**   
085   
086 SeuratObj**$**CellAnnotation **=** pred**$**first.labels   
087   
088 ###################################   
089 ### Removal of wrong cell types ###   
090 ###################################   
091   
092 ca **=** as.vector**(**SeuratObj**$**CellAnnotation**)** #vector with cell annotations 093 indices **=** which**(**grepl**(**regex, ca**))** #indices of cells to keep   
094 SeuratObj **=** SeuratObj**[**,indices**]** #retain cells according to indices   
095   
096 return**(**SeuratObj**)** #return Seurat Object after removal of wrongly assigned cells   
097**}**   
098   
099#function to export the top200 genes for each cluster   
100exT200 **=** **function(**clusterList, name, path **=** getwd**()){**   
101 **for(**i **in** 1**:**length**(**clusterList**)){**   
102 #take the row names from the first 200 rows in clusters unless there's fewer rows available   
103 num **=** ifelse**(**nrow**(**clusterList**[[**i**]])** **<** 200, nrow**(**clusterList**[[**i**]])**, 200**)** 104 top200 **=** rownames**(**clusterList**[[**i**]])[**1**:**num**]**   
105 write.table**(**top200, file **=** paste0**(**path,"/", name, "\_", i, "\_top200.txt"**)**, quote **=** **FALSE**, row.names **=** **FALSE**, col.names **=** **FALSE)**   
106 **}**   
107**}**   
108   
109   
110##################   
111### Data Input ###   
112##################   
113   
114# read files   
115active **=** readRDS**(**"A:/RFILES/E-MTAB-11011 - Single Cell Analysis of B cells in COVID-19 comprising active and recovered disease/.rds-files der Patienten/active-Covid-Kranke/pbmc.active.2.5.3.8\_gex\_and\_vdj.rds"**)**   
116   
117   
118########################################   
119### Remove wrongfully assigned cells ### 120########################################   
121   
122hpca.se **=** celldex**::**HumanPrimaryCellAtlasData**()** #reference data   
123active **=** remWC**(**active**)**   
124   
125   
126########################   
127### Feature Analysis ###   
128########################   
129   
130   
131active\_data **=** scsPipeline**(**active**)**   
132active **=** active\_data**[[**1**]]**   
133active\_clusters **=** active\_data**[[**2**]]** 134########################################################################################################   
135#####################################   
136####Gesamtdatensatz filtern########## 137#####################################   
138   
139ClusterMarkers**=**as.vector**(**active\_clusters,mode**=**"list"**)**   
140   
141**for** **(**i **in** 1**:**length**(**ClusterMarkers**)){**   
142 ClusterMarkers**[[**i**]]=**cbind**(**ClusterMarkers**[[**i**]]**,i**)**   
143 ClusterMarkers**[[**i**]]=**cbind**(**ClusterMarkers**[[**i**]]**,rownames**(**ClusterMarkers**[[**i**]]))**   
144**}**   
145   
146ClusterMarkers**=**do.call**(**rbind,ClusterMarkers**)**   
147#neue Spaltennamen   
148colnames**(**ClusterMarkers**)[**7**]<-**"Genname"   
149colnames**(**ClusterMarkers**)[**6**]<-**"Cluster"   
150   
151#Gesamtdatensatz=GDS   
152GDS**<-**as.data.table**(**ClusterMarkers**)**   
153   
154#Filter fur p<0.05   
155GDS**<-**GDS**[**p\_val**<**0.05**]**   
156   
157#wie oft taucht Gen in GDS auf?   
158GDS**<-**GDS**[**,GDSGenanzahl **:=**.N,by**=**.**(**Genname**)]**   
159#"." in 172 ist Abkuerzung fur Liste   
160#.N gibt die Anzahl (Counts) raus. Sortiere hier nach Genname und Gebe Anzahl raus.   
161#"by" in 172: nach welchen Gruppen soll sortiert werden?   
162   
163#Behalte nur die Gene, die 1 einziges Mal auftauchen (Ueber alle Cluster).   
164GDS**<-**GDS**[**GDSGenanzahl**==**1**]**   
165#Soll man es nur bei GDS machen oder auch alle Gene in 100 Bootstraps auch?   
166#Hier will man nur die unique MArkergene haben. Macht es Sinn Bootstrap zu filtern?   
167#Nur im GDS alle Gene rausschmeissen,die nicht unique sind oder auch Bootstrap?   
168   
169################################   
170####Mit Boot\_Strap abgleichen###   
171################################   
172   
173#Bootstrap aufrufen   
174wd **=** "A:/RFILES/bootstrap/22062022"   
175setwd**(**wd**)**   
176   
177rawfiles **=** list.files**(**wd**)**   
178rawfiles#ueberpruefe welchen Index die Datei hat fuer Zeile 123 179Bootraw**=** read.csv**(**rawfiles**[**1**]**, row.names **=** 1, sep **=**";"**)**   
180#Spalte unbennen fuer Zeile 204, damit GDS und Boot gleichen Spaltennamen haben.   
181colnames**(**Bootraw**)[**8**]<-**"Genname"   
182colnames**(**Bootraw**)[**7**]<-**"Cluster"   
183Boot**<-**as.data.table**(**Bootraw**)**   
184 185##################################################################################   
186#Habe jetzt zwei Data.Tables (GDS und Row). Moechte beide miteinander vergleichen.# 187##################################################################################   
188#Will sehen, welches Gen in meinem GDS finde ich auch in Boot wieder und in welchem Cluster ist es im GDS.   
189#Schreibe die Clusternummer als neue Spalte in Boot.   
190   
191Boot**<-**Boot**[**GDS,RefCluster **:=**i.Cluster,on**=**.**(**Genname**)]**   
192#uebergebe hier Bootstraps meinen Gesamtdatensatz (GDS)   
193#Mappe beide gegeneinander basierend auf Gennamen.   
194#Es sucht also Gennamen raus und mappt gegen Gennamen.   
195#NA, wenn Gene nicht in GDS vorzufinden sind, aber in Boot schon. 196#Fuegt neue Spalte RefCluster hinzu. Zeigt an, welchem Cluster es im GDS enspricht.   
197 198##########################################################################################################   
199   
200   
201#Anzahl der Gene pro Cluster fuer jeden Bootstraprun   
202ClusterCounts**<-**Boot**[**,Clustercounts **:=**.N,by**=**.**(**i,Cluster,RefCluster**)]** 203#Gruppiere nach Bootstrap,Clusterzahl und nach Anzahl an gefundenen Counts.   
204#Clustercounts Spalte sagt, wie oft etwas vorkommt.   
205#Zeigt mir an, wie oft ein Cluster gefunden wurde fuer jeden Durchlauf verglichen mit selben RefCluster und Cluster.   
206#Es ist nicht pro Gen, sondern fuer alle Gene im selben Durchlauf, mit dem selben Cluster und dem selben RefCluster   
207   
208#entferne NA   
209ClusterCounts**<-**na.omit**(**ClusterCounts**)**   
210#muss NAs entfernen, da ansonsten nur noch NAs uebrig bleiben.   
211   
212#enstpricht Cluster RefCluster?   
213ClusterCounts**$**ClusterMatch**<-**ClusterCounts**$**Cluster**==**ClusterCounts**$**RefCluster   
214#Wie oft kommt ein Gen vor   
215ClusterCounts**<-**ClusterCounts**[**,Genanzahl **:=**.N,by**=**.**(**Genname**)]**   
216#Wie viele sind im jeweiligen Cluster pro bootstrap-Durchlauf vorzufinden   
217ClusterCounts**<-**ClusterCounts**[**,CountsfurCluster **:=**.N,by**=**.**(**Cluster,i**)]** 218#Bsp. Bootstraprun 1, Cluster =4,   
CountsfurCluster=8 bedeutet, dass es im Bootstrap-Run 1, Cluster 4 8 Markergene gibt   
219#Wie viele Markergene pro Cluster (ueber alle B-Runs)   
220ClusterCounts**<-**ClusterCounts**[**,FeaturesproCluster **:=**.N,by**=**.**(**Cluster**)]** 221#Wie viele Markergene pro BootstrapRun   
222#Bootanzahl: Wie viele Rows sind pro Bootstrap enthalten 223ClusterCounts**<-**ClusterCounts**[**,BootAnzahl **:=**.N,by**=**.**(**i**)]**   
224   
225#Gesamte Anzahl an Features ueber alle BootstrapDurchlaeufe 226ClusterCounts**<-**ClusterCounts**[**,Featuresanzahl **:=**.N**]**   
227   
228#Gebe i einen neuen Namen.   
229colnames**(**ClusterCounts**)[**6**]<-**"BootstrapRun"   
230colnames**(**Boot**)[**6**]<-**"BootstrapRun"   
231 232################################################################################### 233#####################Matching###################################################### 234##################################################################################   
235   
236# Was ist die Proportion der Anzahl der Markergene mit demselben Cluster,refCluster und Bootstraprun im Verhaeltnis zur Anzahl an Markergenen in dem Cluster und Bootstrap   
237ClusterCounts**<-**ClusterCounts**[**,Proportion **:=(**Clustercounts**/**CountsfurCluster**)\***100**]**   
238   
239#Wie viele cluster und RefCluster stimmen pro Bootstrap ueberein?   
240ClusterCounts**<-**ClusterCounts**[**,MatchproBootstrapCluster **:=**.N,by**=**.**(**BootstrapRun,ClusterMatch**)]**   
241ClusterCounts**<-**ClusterCounts**[**,ProzentanrichtigenClusternproBootstrap **:=(**MatchproBootstrapCluster**/**BootAnzahl**)\***100**]**   
242   
243ClusterCounts**<-**ClusterCounts**[**,MatchproCluster **:=**.N,by**=**.**(**Cluster,ClusterMatch**)]**   
244ClusterCounts**<-**ClusterCounts**[**,ProzentanrichtigenClustern **:=(**MatchproCluster**/**FeaturesproCluster**)\***100**]**   
245   
246#Auf Ebene des einzelnen Gens. Wie viel vom gen wurde richtig gemappt udn wie viele falsch   
247#Bsp.Gen Jup kommt 176 mal vor und er wurde 93 richtig und 83 mal falsch gemappt   
248ClusterCounts**<-**ClusterCounts**[**,MatchGenEbene **:=**.N,by**=**.**(**Genname,ClusterMatch**)]**   
249#Wie oft in Prozent wurde das Gen falsch oder richtig gemappt. 250ClusterCounts**<-**ClusterCounts**[**,ProzentRichtigeGenMatches **:=(**MatchGenEbene**/**Genanzahl**)\***100**]**   
251   
252ProzentRichtigeCluster**<-**ClusterCounts**[**ClusterCounts**[**,.I**[**unique**(**ProzentanrichtigenClustern**)]**,by **=**.**(**Cluster,ClusterMatch**)]$**V1**]**   
253 254############################################################################## 255####################Visualisuerung############################################ 256##############################################################################   
257   
258########################################   
259#Abbildung 1############################ 260########################################   
261   
262df **=** ProzentRichtigeCluster**[**,c**(**7,22,11**)]**   
263   
264#add a dummy row with 0% TRUE matches for cluster 8   
265dummyRow **=** data.frame**(**8, 0, **TRUE)**   
266colnames**(**dummyRow**)** **=** colnames**(**df**)**   
267df **=** rbind**(**df, dummyRow**)**   
268df**$**ProzentanrichtigenClustern **=** round**(**df**$**ProzentanrichtigenClustern**)**   
269   
270ggplot**(**data **=** df,   
271 aes**(**x **=** Cluster, y **=** ProzentanrichtigenClustern, fill **=** ClusterMatch**))+** 272 geom\_bar**(**stat **=** "identity",   
273 position **=** position\_dodge**())** **+**   
274 geom\_text**(**aes**(**label **=** ifelse**(**ProzentanrichtigenClustern **==** 0, "",ProzentanrichtigenClustern**))**,   
275 vjust **=** 1.6,   
276 color **=** "white",   
277 position **=** position\_dodge**(**0.9**)**,   
278 size **=** 3.5**)** **+**   
279 labs**(**x **=** "Cluster",   
280 y **=** " %-Anteil Clusterzuweisung",   
281 fill **=** "Zuweisung"**)** **+**   
282 scale\_x\_continuous**(**breaks**=**1**:**8,   
283 labels**=**as.character**(**1**:**8**))** **+**   
284 scale\_fill\_discrete**(**labels **=** c**(**"Falsch", "Richtig"**))** **+**   
285 # theme\_minimal() + 286 theme**(**text **=** element\_text**(**size **=** 10**))** 287 288 289######################################### 290#Abbildung 2############################# 291######################################### 292#Wie viele Features sind richtig oder falsch gemappt wurden ueber alle verfuegbaren Features. 293ggplot**(**ClusterCounts,aes**(**ClusterMatch,WahrFalsch,fill**=**ClusterMatch**))+** 294 geom\_bar**(**stat**=**"identity",position**=**position\_dodge**())+**geom\_text**(**aes**(**label**=**round**(**WahrFalsch**))**, vjust**=**1.6, color**=**"white",   
295 position **=** position\_dodge**(**0.9**)**, size**=**3.5**)+**   
296 labs**(**x **=** "Richtige oder falsche Zuweisung ueber alle Bootstrap-Durchlaeufe",   
297 y **=** " %-Anteil richtiger Zuweisungen",   
298 fill **=** "Zuweisung"**)**   
299#Problem: Warum sind die Werte so schlecht: Liegt es daran, dass ich einen Fehlergemacht habe oder liegt es an Single Cell und Bootstrapping   
300   
301#########################################   
302#Abbildung 3############################# 303#########################################   
304   
305#Wie viele Markergene tauchen pro Bootstrap-Run auf   
306T1 **=** table**(**ClusterCounts**$**BootstrapRun, ClusterCounts**$**Genname**)**   
307   
308T2 **=** **(**T1**>**0**)**   
309   
310hist**(**apply**(**T2, 2, sum**)**, xlab**=**"Anzahl Bootstrap-Runs", ylab**=**"Anzahl Gene", main**=**"", xlim**=**c**(**0, 100**)**, cex.lab**=**1.5, cex.axis**=**1.5**)**   
311   
312box**()**   
313   
314#########################################   
315#Abbildung 4############################# 316#########################################   
317   
318#Boxplot: Wahrscheinlichkeit, dass ein Gen richtig oder falsch gemappt wird auf Y.   
319ProzentRichtigeGenMatches**<-**ClusterCounts**[**ClusterCounts**[**,.I**[**which.max**(**ProzentRichtigeGenMatches**)]**,by **=**.**(**ClusterMatch,Genname**)]$**V1**]**   
320ProzentRichtigeGenMatches**<-**ProzentRichtigeGenMatches**[**,Median **:=**median**(**ProzentRichtigeGenMatches**)**,by**=**.**(**ClusterMatch**)]** 321ProzentRichtigeGenMatches**<-**ProzentRichtigeGenMatches**[**,max **:=**max**(**ProzentRichtigeGenMatches**)**,by**=**.**(**ClusterMatch**)]** 322ProzentRichtigeGenMatches**<-**ProzentRichtigeGenMatches**[**,min **:=**min**(**ProzentRichtigeGenMatches**)**,by**=**.**(**ClusterMatch**)]** 323#ProzentRichtigeGenMatches<-na.omit(ProzentRichtigeGenMatches) 324ggplot**(**ProzentRichtigeGenMatches,aes**(**y**=**ProzentRichtigeGenMatches,group**=**ClusterMatch**))+**geom\_boxplot**(**fill**=**c**(**"pink","darkred"**)**,aes**(**x**=**ClusterMatch**))+** 325 labs**(**x**=**"Boxplot fuer Markergene mit falschen Zuweisungen und Boxplot fuer Markergene mit richtigen Zuweisungen",y**=**"Reproduzierbarkeit in %",title**=**"Prozentsatz an Markergenen im Bootstrap mit denselben Clustern wie im Gesamtdatensatz"**)**   
326   
327   
328######################################   
329#Abbildung 5########################## 330######################################   
331   
332   
333df**<-**ClusterMarkers   
334   
335df**<-** subset**(**df,p\_val**<**0.05**)**   
336   
337C1**<-**df**[**df**$**Cluster **==** 1,7**]**   
338   
339C2**<-**df**[**df**$**Cluster **==** 2,7**]**   
340   
341C3**<-**df**[**df**$**Cluster **==** 3,7**]**   
342   
343C4**<-**df**[**df**$**Cluster **==** 4,7**]**   
344   
345C5**<-**df**[**df**$**Cluster **==** 5,7**]**   
346   
347C6**<-**df**[**df**$**Cluster **==** 6,7**]**   
348   
349C7**<-**df**[**df**$**Cluster **==** 7,7**]**   
350   
351C.all **=** list**(**C1, C2, C3, C4, C5, C6, C7**)**   
352   
353   
354Genenames **=** table**(**df**$**Genname**)**   
355   
356length**(**which**(**Genenames **==** 1**))** # -> 214 Gene names show up   
357   
358ClusterNames **=** paste0**(**"Cluster ", 1**:**7, "\n", "(n = ", sapply**(**C.all, **function(**x**)** length**(**x**))**, ")"**)**   
359   
360venn**(**C.all, zcolor**=**"style", snames **=** ClusterNames, box **=** **FALSE**, ilcs **=** 1, sncs **=** 1**)**

# Danksagung

Ich möchte hier Herrn Prof. Dr. Jung danken. Ich danke Ihnen für die Bereitstellung des Themas, für die Betreuung und für die Hilfe, die ich von Ihnen während der Bachelorarbeit bekommen habe. Bei Ihnen konnte ich nicht nur meine Bachelorarbeit schreiben, sondern auch viel Wissen im Bereich Bioinformatik aneignen. Herrn Prof. Dr. Bernd Schierwater möchte ich hier dafür danken, dass er sich als Zweitprüfer für meine Bachelorarbeit bereitgestellt hat. Darüber hinaus möchte ich mich bei allen Mietgliedern der Arbeitsgruppe “ Genomics and Bioinformatics of Infectious Diseases” aus dem Institut für Tierzucht und Vererbungsforschung für die freundliche Aufnahme in ihr Team und die tolle Zeit, die ich mit Ihnen verbringen konnte, bedanken. Ein besonderer Dank gilt hier auch an Michael Selle für die Betreuung während der Bachelorarbeit, für die Ratschläge, die er mir gegeben hat und für das Wissen, das er mir vermittelt hat.

Die Erfahrungen, die ich während meiner Arbeit gesammelt habe, und das Wissen, dass ich vermittelt bekommen habe, inspirierten mich, mich für den Master medizinische Datenwissenschaften zu entscheiden und eine zukünftige Karriere im Bereich Bioinformatik und Datenwissenschaften anzustreben. Dafür werde ich für immer dankbar sein.

Vielen Dank.

Ehrenwörtliche Erklärung

Hiermit versichere ich, dass die vorliegende Bachelorarbeit selbstständig von mir verfasst wurde und ich dabei keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Des Weiteren versichere ich, dass alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht sind. Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Ort, Datum) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 Sergej Ruff