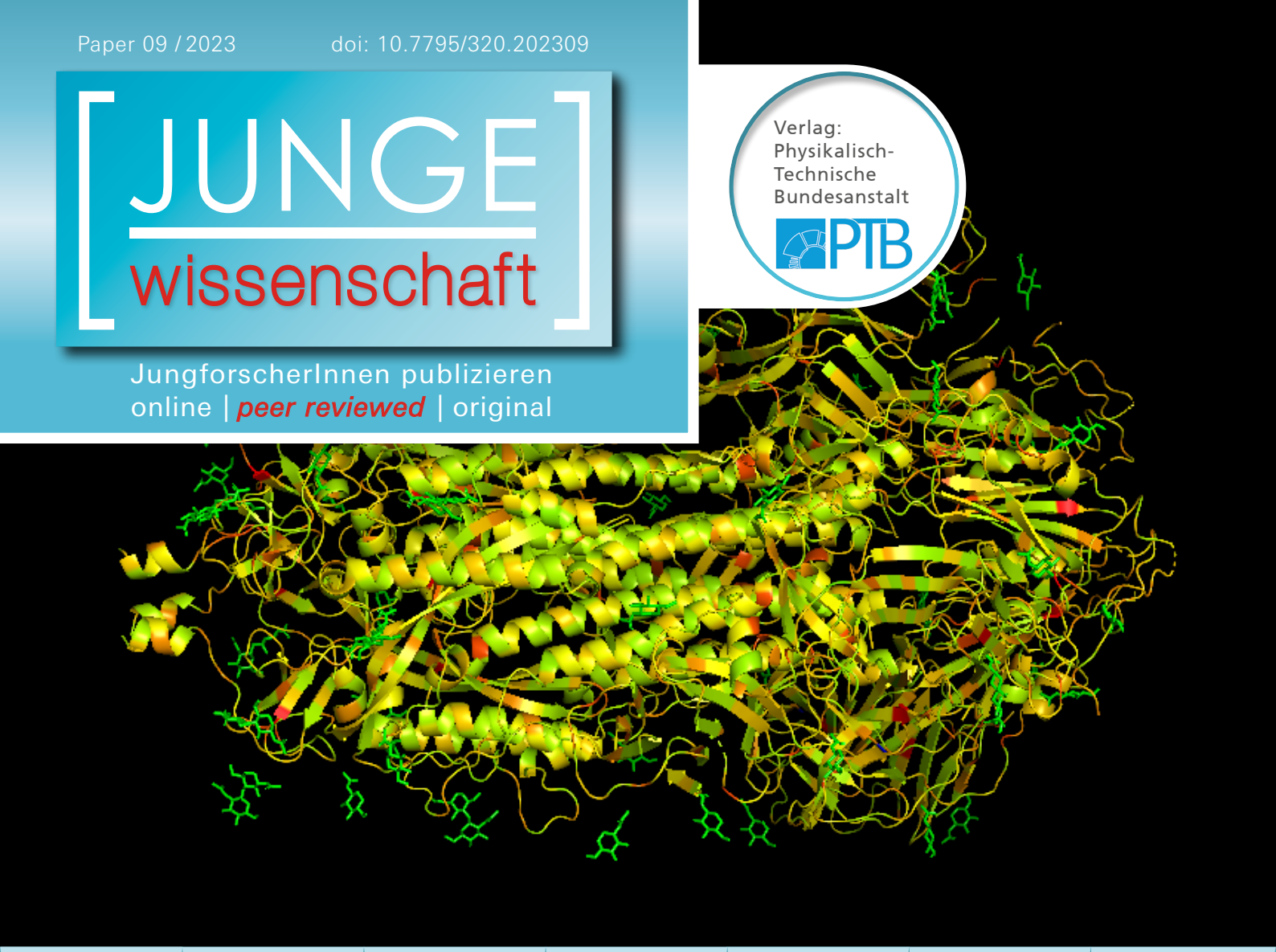
Paper 09 / 2023 doi: 10.7795/320.202309

JUNGE wissenschaft

JungforscherInnen publizieren
online | *peer reviewed* | original

Verlag:
Physikalisch-
Technische
Bundesanstalt
PTB



Mathematik & Informatik

Mutationen des SARS-CoV-2 Virus auf der Spur

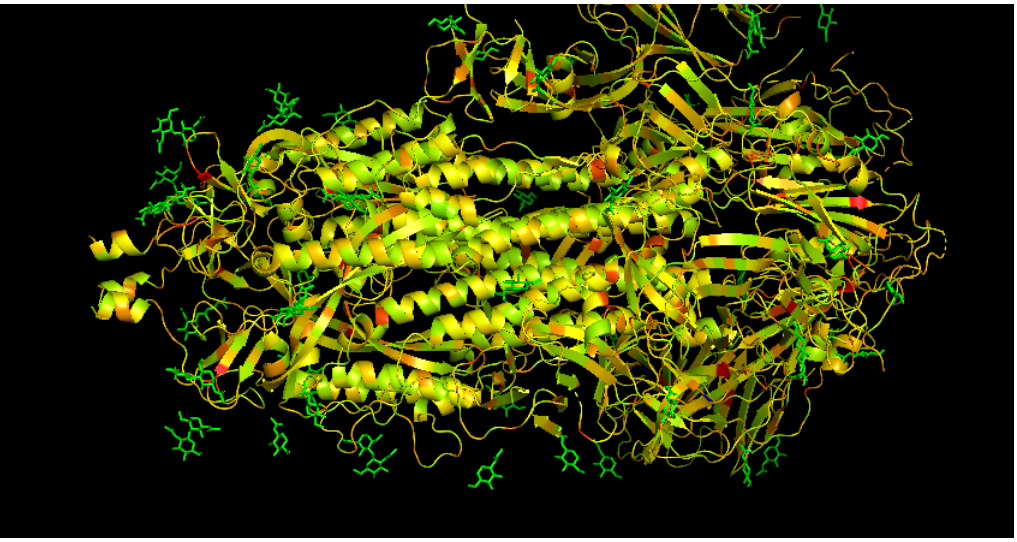
Visualisierung der Ähnlichkeit zwischen Sequenzen des Spike-Proteins aus benachbarten Ländern mithilfe der 3D-Struktur

Die Arbeit beschäftigt sich mit der Varianz der Virussequenz des Coronavirus anhand des Spike-Proteins in Deutschland und Frankreich. Dazu wurden die beiden Konservierungswerte der beiden Sequenzen verglichen und grafisch dargestellt. Es ließ sich feststellen, dass die Varianz der Sequenzen in beiden Ländern ähnlich hoch ist, ähnliche Stellen gut konserviert und somit für das Virus essentiell sind.

DIE JUNGFORSCHERIN

Lara Hombrecher (2004)
Geschwister-Scholl-Gymnasium, Lebach

Eingang der Arbeit:
17.01.2023
Arbeit angenommen:
21.03.2023



Mutationen des SARS-CoV-2 Virus auf der Spur

Visualisierung der Ähnlichkeit zwischen Sequenzen des Spike-Proteins aus benachbarten Ländern mithilfe der 3D-Struktur

1. Einleitung

Seit 2020 hat das Coronavirus die Welt im Griff. Diese neuartige Virusvariante wurde schnellstmöglich erforscht, um beispielsweise einen Impfstoff oder ein Medikament gegen den Krankheitserreger zu entwickeln. Dabei fielen in verschiedenen Ländern viele Daten zur Analyse an.

Besonders interessant an einem Virus ist der Aufbau seines Spike-Proteins an der Oberfläche, das auch für die Erkennungssysteme unseres Immunsystems eine wichtige Rolle spielt, und auf dessen Information beispielsweise auch die Wirkungsweise des mRNA-Impfstoffes von BioNTech-Pfizer basiert. In vielen Ländern sind Forschungen diesbezüglich angestellt worden, im Rahmen derer auch eine genetische Aufschlüsselung des Spike-Proteins stattfand. Es existieren also eine Menge an Sequenzen, die an unterschiedlichen Standorten in unterschiedlichen Ländern entstanden sind.

Ich habe mir die Frage gestellt, ob man Unterschiede im Spike-Protein zwischen den einzelnen Ländern feststellen kann, also ob die Virusvarianten von Land zu Land stark variieren. Um dies herauszufinden, ermittelte ich die Ähnlichkeiten und die Erhaltung mithilfe von Methoden der Bioinformatik und stellte diese grafisch anschaubar und vergleichbar dar. Im Idealfall sind hierbei die Ergebnisse benachbarter oder nah beieinander liegender Länder vergleichbar, jedoch niemals identisch, da die Mutationsrate des Coronavirus sehr hoch ist und Veränderungen sehr schnell auftreten.

2. Vorgehensweise

Zur Veranschaulichung wird die Ähnlichkeit zwischen den einzelnen Sequenzen mithilfe der Konservierung visualisiert. In einem Visualisierungs-Programm wie PyMOL [12], eine Open-Source-Software, lässt sich ein Attribut der Struktur, der B-Faktor, farblich darstellen, weshalb ich für meine Analyse diesen durch die Konservierungswerte ersetze und dann farbig markiere.

PyMOL benötigt hierzu eine sogenannte PDB-Datei, die alle relevanten Daten zur grafischen Darstellung enthält.

Um zu einem anschaulichen Ergebnis zu gelangen, müssen die folgenden Schritte durchlaufen werden:

- Einlesen der SARS-CoV-2-Sequenzen
- Multiples Sequenzalignment
- Berechnen der Konservierung
- Konsensussequenz
- Einlesen der PDB-Datei
- Änderung des B-Faktors
- Visualisierung

```
>Spike|hCoV-19/England/CAMB-7EE03/2020|2020-04
MFVFLVLLPLVSSQCVNLTTRTQLPPAYTNSFTRGVYYPDKVFRSS
>Spike|hCoV-19/England/CAMB-81447/2020|2020-04
MFVFLVLLPLVSSQCVNLTTRTQLPPAYTNSFTRGVYYPDKVFRSS
>Spike|hCoV-19/France/IDF-5850/2020|2020-03-26
MFVFLVLLPLVSSQCVNLTTRTQLPPAYTNSFTRGVYYPDKVFRSS
>Spike|hCoV-19/France/IDF-3359/2020|2020-03-23
MFVFLVLLPLVSSQCVNLTTRTQLPPAYTNSFTRGVYYPDKVFRSS
```

Abb. 1: Ausschnitt aus der Spike-Protein-Datei

doi: 10.7795/320.202309

2.1 Einlesen der SARS-CoV-2-Sequenzen

Um einen Vergleich zweier Länder darzustellen zu können, müssen die entsprechenden Sequenzen der jeweiligen Länder aus der Datei, welche von GISAID [4] stammt, mit den gesammelten Sequenzen sortiert werden.

Der Pythoncode, der verwendet wird, um die erste Datei mit den Daten über die Spike-Proteine einzulesen, lässt sich grob in zwei Teile gliedern: Zuerst wird eine Funktion definiert, die es ermöglicht, große Fasta-Dateien einzulesen, da immer nur eine Zeile ausgewertet wird (siehe Abb. 1). Anschließend wird mit dieser Funktion die tatsächliche Datei bearbeitet und nach dem angegebenen Land sortiert.

Dieser Vorgang besteht aus vier Arbeitsschritten:

1. Existenz der Datei prüfen, wenn nötig entpacken
2. Befreien der Zeilen von außertextlichen Zeichen
3. Aufnehmen jeder Zeile als Paar aus einem Namen und dazugehöriger Sequenz
4. Eintragen der Sequenzen aus dem entsprechenden Land

```
if line.startswith(">"):
    if len(seqs) != 0:
    # there was a sequence before
        yield seq_name, "".join(seqs)
        seq_name = line[1:]
        seqs = []
else:
        seq_name = line[1:]
else:
    seqs.append(line)
```

Abb. 2: Sequenz-Namenpaare bilden

Ist die Datei vorhanden, so kann sie als Klartext oder gzip-codiert vorliegen. Ist letzteres der Fall, wird sie entpackt. Schließlich wird die Datei im Lesemodus geöffnet.

Es werden alle Zeilen bearbeitet, die keine Leerzeilen sind. Das Programm initialisiert nun je eine Variable für die Sequenz und ihren dazugehörigen Namen. Danach wird jede zu bearbeitende Zeile von außertextlichen Zeichen wie z. B. Leerzeichen befreit.

Nun wird anhand des Kriteriums „>“, welches im Fasta-Format für den Namen einer Sequenz steht, für jede Zeile überprüft (siehe Abb. 2), ob es sich um einen Namen handelt. Ist das nicht der Fall, so handelt es sich um die Sequenz selbst, die dann der Variable `seqs` hinzugefügt wird.

Durch das Messen der Länge der Variable `seqs` wird festgestellt, ob es sich um die erste Sequenz der Datei handelt. Ist dies der Fall, so wird die letzte Sequenz mit ihrem Namen zurückgegeben, danach wird die neue Zeile, von der man bereits festgestellt hat, dass sie den Namen einer Sequenz beinhaltet, als neuer Wert der Variable `seq_name` genommen, die Variable `seqs` wird wieder „geleert“ damit sie im nächsten Schritt die neue Sequenz, die jetzt zu dem Namen passt, aufnehmen kann.

```
germany_file = open("spike_germany1.fasta", "w")
france_file = open("spike_france1.fasta", "w")
for seq_id, seq in read_fasta_gen("spikeprot0111.fasta"):
    if "France" in seq_id:
        france_file.write(seq_id + "\n")
        france_file.write(seq + "\n")
    elif "Germany" in seq_id:
        germany_file.write(seq_id + "\n")
        germany_file.write(seq + "\n")
france_file.close()
germany_file.close()
```

Abb. 3: Klassifizierung der Sequenzen nach Deutschland/Frankreich

doi: 10.7795/320.202309

Ist die Sequenz jedoch die erste, das heißt es gab keinen Eintrag vorher, so wird die Zeile einfach nur als Name angenommen.

Nun startet die `if`-Verzweigung wieder von vorne, die Zeile beginnt jedoch nicht mit einem >, sondern mit der Sequenz, was bedeutet, dass dieser Zeile der vom vorherigen Schritt noch entleerten Variable angefügt wird, bevor dann das ganze Verfahren für die nächste Zeile wiederholt wird.

`Yield` erzeugt die Ausgabe dieser Funktion, speichert anders als eine return-Anweisung jedoch die Werte der lokalen Variablen, kann also direkt bei dem gespeicherten Zustand fortfahren.

Die letzte Zeile der Datei wird eine leere sein, woraufhin die letzte `if`-Verzweigung greift, der die Sonderregelung für die letzte Sequenz beinhaltet. Diese wird mitsamt ihrem Namen ausgegeben, die Variablen werden nicht mehr neu initialisiert, die Generatorfunktion endet.

Im nächsten Teil werden zuerst zwei neue Dateien initialisiert, in die später die entsprechenden Sequenzen eingetragen werden. Nun läuft die Funktion, die im Code vorher definiert wurde, über die große Datei mit allen Spike-Protein-Dateien und filtert jene heraus, in deren Namen entweder „France“, steht, was bedeutet, dass dieses Protein in Frankreich sequenziert wurde, oder „Germany“ für Deutschland (siehe Abb. 3). Für diese Sequenzen schreibt das Programm Name und Sequenz in die jeweilige Datei und prüft schließlich die nächste Zeile.

Dieser Abschnitt benötigt mit 3,6875 Sekunden 58 % der Gesamtlaufzeit des Programms.

Am Ende der Spike-Protein-Datei werden die beiden Dateien, die nun alle Sequenzen aus Frankreich bzw. Deutschland enthalten, geschlossen.

2.2 Multiples Sequenzalignment

Ein Sequenzalignment lagert zwei oder mehrere Sequenzen optimal übereinander, d. h. es lagert sie so, dass möglichst viele Positionen chemisch ähnlich oder identisch sind. [7]

Dabei wird jeder möglichen Übereinanderlagerung ein Score zugewiesen, der darauf basiert, wie viele übereinanderliegende Stellen identisch sind. Dies wird durch das Einfügen sogenannter „Gaps“ (engl. für Lücken) erreicht. Somit lässt sich die Ausrichtung der Sequenzen finden, bei der möglichst viele Stellen gleich sind, zudem erhalten alle Sequenzen durch die Gaps die gleiche Länge.

Diese Eigenschaften sind sehr wertvoll für die weitere Analyse einer Reihe von Sequenzen, die in diesem optimal aufeinander abgepassten Zustand nun z. B. im Hinblick auf ihre Konservierung untersucht werden können.

In dieser Arbeit wurde das Softwareprodukt ClustalOmega genutzt, welches multiple Sequenzalignments ausführen kann, also drei oder mehr Sequenzen miteinander aligniert.[3]

ClustalOmega kann maximal 4000 Sequenzen miteinander alignieren, die gefilterten Sequenzen sind allerdings deutlich zahlreicher. Daher musste die Größe der zu alignierenden Datei an-

```
#Dicitionary
D = {,A':0, ,R':1, ,N':2, ,D':3, ,C':4, ,Q':5, ,E':6, ,G':7, ,H':8, ,I':9,\
,L':10, ,K':11, ,M':12, ,F':13, ,P':14, ,S':15, ,T':16, ,W':17, ,Y':18, ,V':19}
D_back = {}
for x in D:
  D_back[D[x]] = x
w, h = 20, len(aln[0])
list_of_lists = [[0 for x in range (h)] for y in range (w)]
for y in aln :
  x = 0
  for B in y :
    if B in D :
      list_of_lists [D[B]] [x] = list_of_lists [D[B]] [x]+1
      x = x+1
    else :
      x = x+1
```

Abb. 4: Berechnung der `list_of_lists`

doi: 10.7795/320.202309

gepasst werden. Ich habe beschlossen, mich zunächst auf einen Testwert von 2000 Sequenzen zu beschränken.

Hierzu wurde ein Hilfsprogramm geschrieben, das nur die ersten 2000 Sequenzen einer Datei mitsamt ihrem Namen ausgibt und dem Fasta-Format angleicht, sodass ClustalOmega diese Daten verarbeiten kann.

2.3 Berechnung der Konservierung

Wenn biologische Strukturen (bspw. Aminosäure- oder Nukleotidsequenzen) im Laufe der Evolution weitestgehend erhalten und gleichgeblieben sind, so werden sie als konserviert bezeichnet.[6]

Durch die Konservierung lassen sich Stellen in der Sequenz mit vielen Abweichungen darstellen, weshalb dieser Wert für den grafischen Vergleich der Varianz wichtig ist.

Schematisch lässt sich der Code zur Berechnung in fünf Arbeitsschritte gliedern, die den groben Ablauf skizzieren.

- Einlesen der Fasta-Datei
- Berechnen der Häufigkeit
- Einlesen der Substitutionsmatrix
- Berechnen der Konservierung
- Ausgabe der Daten

Wichtige Komponenten sind hierbei die Alignmentdatei, eine BLOSUM-Substitutionsmatrix, die die Ähnlichkeit der einzelnen Aminosäuren und ihre Wahrscheinlichkeit, im Laufe der Zeit ineinander zu mutieren, festhält, und die folgende Formel zur Berechnung der Konservierung [14]:

$$V = \sum_{a}^{K} \sum_{b}^{K} p_a p_b M(a,b) \qquad (1)$$

Die Formel besteht aus der Summe über alle möglichen Kombinationen der Aminosäuren, deren Produkt aus den beiden Häufigkeiten und dem zugehörigen Eintrag in der Substitutionsmatrix.

1. A B Q R — l_o_l [A] [0] = 1, l_o_l [B] [1] = 1, l_o_l [Q] [2] = 1, l_o_l [R] [3] = 1
2. A B N L — l_o_l [A] [0] = 2, l_o_l [B] [1] = 2, l_o_l [N] [2] = 1, l_o_l [L] [3] = 1
3. B N Q T — l_o_l [B] [0] = 1, l_o_l [N] [1] = 1, l_o_l [Q] [2] = 2, l_o_l [T] [3] = 1
4. A K L M — l_o_l [A] [0] = 3, l_o_ [K] [1] = 1, l_o_l [L] [2] = 1, l_o_l [M] [3] = 1

Abb. 5: Beispiel für Aminosäurelisten

Die Häufigkeit einer Aminosäure wird berechnet über ihre Anzahl, welche durch eine Matrix `list_of_lists` erfasst wird. In dieser befinden sich 20 Listen, die jeweils für eine Aminosäure stehen und die Länge der Alignments haben (siehe Abb. 4).

Für jede Sequenz in dem Alignment wird nun jede Position erfasst und die entsprechende Stelle in der entsprechenden Aminosäure-Liste um eins nach oben gezählt.

In dem Beispiel in Abb. 5 wird zuerst die erste Sequenz Position für Position untersucht, die `list_of_lists` für A wird an der 1. Stelle (0) um eins nach oben gezählt, B an der 2. Stelle, Q an der 3. und R an der 4. Danach wird die zweite Sequenz untersucht, die A-Liste wird an der 1. Stelle wieder um eins nach oben gezählt und alle weiteren Positionen der Sequenz ebenfalls erfasst. In der dritten Sequenz steht B an erster Stelle, daher wird für die Position 0 nun die B-Liste um eins nach oben gezählt.

Nach diesem Schema verfährt das Programm und erhält zum Schluss für jede Position die genaue Anzahl der Aminosäuren, die an dieser Stelle in allen Sequenzen stehen. Die Häufigkeit für eine Aminosäure wird dann berechnet, indem man die Anzahl durch die Anzahl aller betrachteten Sequenzen teilt.

In der BLOSUM-Substitutionsmatrix wird aus der chemischen Ähnlichkeit der Aminosäuren und auf Basis der Häufigkeit ihres Vorkommens ein Wert ermittelt, mit dessen Rate die Aminosäuren im Laufe der Zeit ineinander übergehen.

Aus diesen Informationen kann man nun den Konservierungswert berechnen, der sich zusammensetzt aus dem Vorkommen der Aminosäuren und deren Einträgen in der Matrix, welche Ersetzungen bewerten.

Dieser Konservierungswert wird nun grafisch dargestellt, um ihn durch die 3D-Struktur des Spike-Proteins vergleichbar zu machen.

Die Berechnung der Konservierung benötigt 1,125 Sekunden, also 18 % der Gesamtlaufzeit.

2.4 Konsensussequenz

Aus den maximalen Vorkommen an jeder Position wird eine Sequenz konstruiert, die pro Position die am häufigsten auftretende Aminosäure enthält. Diese nennt man Konsensussequenz.[5] In dieser Konsensussequenz entspricht jede Position einem Konservierungswert, der im vorherigen Schritt berechnet wurde. Somit eignet sich diese Sequenz im weiteren Verlauf als Hilfsmittel für das korrekte Zuordnen der Konservierungswerte.

Um die Konsensussequenz zu bilden, muss zuerst die Alignment-Datei erneut eingelesen werden. Das Fasta-Format-typische „>“ entscheidet zwischen der eigentlichen Sequenz und ihrem Namen. Beide Komponenten werden in separate Listen sortiert.

In einem Dictionary wird zunächst jeder Aminosäure ein Index zugeordnet, damit diese als Listen weiterbearbei-

doi: 10.7795/320.202309

tet werden können. Zur späteren Rückübersetzung der Indizes in Aminosäuren, wird auch noch die Invertierung dieser Zuordnung als ein Dictionary `D_back` angelegt (siehe Abb. 4). Um die einzelnen Aminosäurepositionen in den Sequenzen zu erfassen, wird zuerst eine Art Matrix erzeugt. Diese besteht aus 20 verschiedenen Listen, welche jeweils für eine mögliche Aminosäure stehen. Die Indizes, die im Dictionary Aminosäuren zugewiesen wurden, erlauben eine Erstellung von 20 Listen. Jede dieser 20 Listen in der allgemeinen `list_of_lists` ist eine Liste der gleichen Länge wie die Alignments, die aufgrund des Algorithmus alle die gleiche Länge besitzen. So kann für jede Position in den alignierten Sequenzen die Anzahl der Aminosäure nach dem gleichen Verfahren wie bei der Berechnung der Konservierung bestimmt werden.

Bei der Konsensussequenz soll nun an jeder Stelle die Aminosäure stehen, die am häufigsten vorkommt. Dazu prüft der Algorithmus in Abb. 6 an jeder Position der Alignments, welche Aminosäure an die zugehörige Stelle der Konsensussequenz gehört.

Es wird ein Counter `c` initialisiert, der die aktuelle Position speichert, für die die häufigste Aminosäure geprüft wird. `Y` durchläuft hierbei alle Positionen der Alignments und zählt `c` immer eine Position weiter, nachdem die häufigste Aminosäure für die vorherige Position ermittelt wurde.

Die Variable `Häufigste` speichert den Index der Liste, die die Aminosäure repräsentiert, die momentan an dieser Stelle das höchste Vorkommen hat. Sie startet bei 0, was in diesem Fall der Aminosäure „A“ entspricht. `x` durchläuft als Index alle 20 Aminosäurelisten. Für jede Liste wird geschaut, ob der Wert an der aktuellen Position `c` größer ist als der Wert der bisherigen „häufigsten“ Liste. Ist dies der Fall, wird der neue Index als häufigster Index angenommen.

Ist die Anzahl von Aminosäuren der ersten Liste, also „R“, größer als die Anzahl der „A“, so wird der neue Wert der Variable `Häufigste = 1` betragen. Kommt „A“ öfter vor, bleibt `Häufigste = 0` bestehen.

Sind alle 20 Listen durchlaufen und abgeglichen, steht die Liste mit dem größten Vorkommen fest und wird durch das umgekehrte Dictionary `D_back` wieder von einer Indexzahl in eine Aminosäure umgeschrieben, die dann in der Konsensussequenz-Datei gespeichert wird.

Schließlich erhöht sich der Counter `c` um eins und das Verfahren prüft die nächste Position.

Das Erstellen der Konsensussequenz benötigt 0,875 Sekunden, also 14 % der Gesamtlaufzeit.

2.5 Einlesen der PDB-Datei

Um nun weiterzuarbeiten, muss man zunächst die PDB-Datei, die die strukturellen Daten des Spike-Proteins zur Visualisierung erhält, einlesen. PDB steht für *Protein Data Bank*.[10] Hier werden die Koordinaten und andere Attribute von Atomen in Proteinen oder anderen Makromolekülen gespeichert. Dieses Dateiformat wird zur Untersuchung von Proteinen wie in diesem Fall verwendet [9]. Biopython besitzt für solche Untersuchungen das Modul Bio.PDB.

Dazu wird die Datei zunächst geparst, also typspezifisch in ihre einzelnen Komponenten eingelesen (siehe Abb. 7). PDB-Dateien haben einen besonderen Aufbau, der sie in mehrere, ineinander verschachtelte Elemente unterteilt.

Uns interessiert hier besonders der B-Faktor, ein Attribut der Atome in der Struktur des Proteins, aus welchen die Residuen [13], also die chemischen Untereinheiten des Proteins, aufgebaut sind. Der grafische Vergleich lässt sich durch den B-Faktor der CA-Atome optimieren, weshalb man in der Struktur der Datei zuerst das erste Model

```
Ausgabe = open(new_file_name, "w")
c = 0 #Counter durchläuft alle Positionen v. Alignment
for y in aln[1]: #y enthält Namen v. Aminosäure bei c
   Häufigste = 0
   for x in range (20): #durchläuft alle Aminosäuren 0-19
      if list_of_lists [x] [c] > list_of_lists [Häufigste] [c] :
         Häufigste = x
   Ausgabe.write(D_back[Häufigste])
   c = c+1
Ausgabe.close()
```

Abb. 6: Bildung der Konsensussequenz

doi: 10.7795/320.202309

```
parser = PDBParser()
Struktur = parser.get_structure(„Datei“,“6vxx.pdb“)
Model = Struktur[0]
chain_A = Model[“A”]
chain_B = Model[“B”]
chain_C = Model[“C”]
```

Abb. 7: Einlesen der PDB-Struktur

(also das erste und in dieser Datei einzige Experiment) und dann die einzelnen Chains (Ketten) durchläuft und die Residuen abgreift, von deren CA-Atome man den B-Faktor ändern möchte.

Jetzt erhält man eine Liste, die die Residuen enthält. Um diese mit der Konsensussequenz vergleichen und über diesen Vergleich schließlich auch die Konservierungswerte zuweisen zu können, muss man die Buchstabencodes jedoch noch umwandeln.

In der Datei sind die Aminosäuren als Drei-Buchstabencode niedergeschrieben, in den hier verwendeten Sequenzen jedoch nur als Ein-Buchstabencode.

`seq1` ist eine Funktion in Biopython, die eine Rücktranslation vom 3-Buchstabencode in den 1-Buchstabencode durchführt. Man ruft also nur jede Position des Chains auf, speichert ihren momentanen Namen und konvertiert ihn entsprechend. Zum Schluss wird diese neue Bezeichnung in einer extra Variable gespeichert, mit der man im Weiteren arbeitet.

In einem Chain befinden sich alle Residuen. Allerdings sind nicht alle Residuen Aminosäuren, jedoch besitzen nur Aminosäuren das CA-Atom, dessen B-Faktor geändert werden soll.

Aminosäure-Residuen erkennt man an ihrer ID, die in einem PDB-File aus drei Informationen besteht. Ist die erste Information unbesetzt, so handelt es sich um eine Aminosäure.

Um also ein korrektes Alignment und auch ein korrektes Mapping im weiteren Verlauf zu gewährleisten, dürfen nur die Aminosäure-Residuen anhand ihrer unbesetzten ersten Information der ID herausgefiltert werden.

Für den weiteren Vergleich braucht man zwei Werte aus vorherigen Arbeitsschritten, die Konsensussequenz, die für das Vergleichen und Positionieren der Konservierungswerte wichtig ist, und natürlich die richtigen Konservierungswerte, durch welche man den B-Faktor ersetzen will.

Diese beiden Dateien werden im Folgenden Zeile für Zeile in das Programm eingelesen und von allen außertextlichen Zeichen befreit.

Im nächsten Schritt wird die Konsensussequenz mit der Sequenz der Residuen aligniert. Diese Pairwise-Alignment-Funktion in Biopython gibt vier Argumente zurück: die erste Sequenz, die man eingelesen hat, ein Chart, das die Matches verdeutlicht, die zweite Sequenz mit den entsprechenden Gaps und ein viertes, leeres Argument.

Von der Alignment-Funktion werden mehrere Alignments erstellt, da es mehrere Möglichkeiten gibt, die beiden Sequenzen übereinander zu lagern. Diese verschiedenen Alignments werden absteigend gespeichert. Die ersten Alignments haben daher den höchsten Score und somit konnte ich für den weiteren Verlauf des Programms mit dem besten Alignment weiterarbeiten.

Danach wird das Alignment-Objekt wie bereits beschrieben in diese vier Komponenten aufgeteilt, da man jetzt die beiden alignierten Sequenzen zur Verfügung hat, um den B-Faktor auszutauschen.

Die Laufzeit des Einlesens beträgt 0,40625 Sekunden, was rund 7 % der Gesamtlaufzeit entspricht.

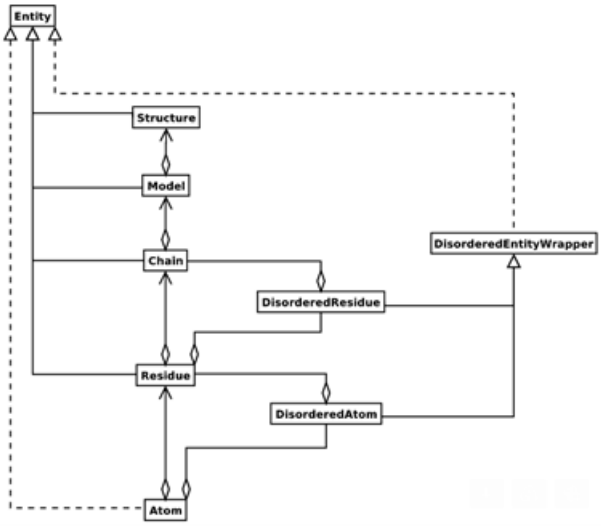


Abb. 8: Die Struktur einer PDB-Datei [8]

doi: 10.7795/320.202309

2.6 Änderung des B-Faktors

Der B-Faktor, auch Temperaturfaktor oder Deplatzierungsfaktor genannt, repräsentiert die Streuung der Atome [1]. Um die dargestellten Werte in der Grafik vergleichen zu können, muss der B-Faktor in die entsprechenden, vorher berechneten Konservierungswerte umgetauscht werden. Dazu muss für jedes der 972 CA-Atome (C-alpha, das zentrale C-Atom einer Aminosäure [2]) in der PDB-Datei 6vxx der entsprechende, durch das vorherige Alignment zugeordnete, Konservierungswert gefunden und durch den B-Faktor ersetzt werden.

Für jede Position wird also geprüft, ob ein Konservierungswert zu dem passenden Aminosäure-Residuum vorliegt und anschließend der B-Faktor entsprechend getauscht.

Dazu initialisiere ich eine Schleife, die nun jede Position eines Alignment-Objektes durchläuft (siehe Abb. 9). Da beide Alignments gleich lang sind, ist es egal, wessen Länge genommen wird. In diesem Fall wurde die erste Sequenz gewählt.

Dann werden drei Counter initialisiert: `x`, der jede Position durchläuft, `res _ counter` und `aln3 _ counter`, die für das Überspringen von Gaps in der jeweiligen Liste zuständig sind. Dies ist notwendig, da durch das Alignment Gaps eingefügt wurden, also in den Sequenzen jetzt Positionen existieren, denen gar kein Wert gegenübersteht, bei denen man also den B-Faktor durch nichts ersetzen kann.

Es gibt drei Fälle der Übereinanderlagerung, die im Folgenden betrachtet werden:

1. Es gibt ein Residuum, das mit einer Position der Konsensussequenz

```
def counting_mechanism(conservation_value, residue_list,
res_alignment, cons_alignment):
   x = 0
   aln3_counter = 0
   res_counter = 0
   for y in range(len(res_alignment)) :
      if res_alignment[x] != "-" and cons_alignment[x] != "-" :
         conservierungswert = float(conservation_value[aln3_counter])
         residue = residue_list[res_counter]
         res_counter += 1

         ats = residue.get_atoms()
         for atom in ats :
            name = atom.get_name()

            if name == "CA" :
                  richtig = atom
                  richtig.set_bfactor(conservierungswert)

         aln3_counter = aln3_counter+1
         x = x+1
      elif res_alignment[x] == "-" and cons_alignment [x] != "-" :
         aln3_counter = aln3_counter+1
         x = x+1
      elif res_alignment[x] != "-" and cons_alignment [x] == "-" :
         res_counter = res_counter+1
         x = x+1
```

Abb. 9: Zählmechanismus für jede Position

doi: 10.7795/320.202309

```
def Schere(chain):
  for residue in chain :
      if residue.get_id() [0] == " ":
          for x in residue.get_atoms():
              if x.get_name() == "CA" :
                  B_Faktor = x.get_bfactor()
                  if B_Faktor > 1:
                      ResiduumId = residue.get_id()
                      chain.detach_child(ResiduumId)
```

Abb. 10: Ausschneiden der Aminosäuren ohne Wert

und damit einem Konservierungswert korrespondiert.
2. Es gibt zwar einen Konservierungswert, aber kein dazu korrespondierendes Residuum.
3. Es gibt ein Residuum, aber keinen dazu passenden Konservierungswert.

In Fall 2 und 3 kann man den B-Faktor nicht ersetzen, daher werden diese beiden Fälle damit abgehandelt, dass man den Counter der jeweiligen Sequenz, in der der Gap nicht aufgetreten ist, um eins nach vorne stellt, um sicherzugehen, dass an der nächsten Position nicht dieselbe Aminosäure überprüft wird.

Nur im ersten Fall lässt sich der B-Faktor ersetzen. Mittels der Sequenzcounter greift das Programm nun den zu der Position dazugehörigen Konservierungswert und das Residuum der PDB-Datei.

Von allen Atomen, aus denen ein Residuum besteht, will man, dass nur der B-Faktor des CA-Atoms geändert wird. Diesen kann man im Visualisierungsprogramm extra herausfiltern. Daher müssen alle Atome auf ihren Namen untersucht werden und schließlich muss das CA-Atom herausgefiltert und weiterbearbeitet werden. Nachdem man den B-Faktor durch den korrespondierenden Konservierungswert ersetzt hat, werden die beiden Sequenzcounter eins nach oben gesetzt, damit beim nächsten Durchlauf die nächste Position betrachtet wird.

Aufgrund der Fälle 2 und 3 existieren jedoch auch CA-Atome, deren B-Faktor nicht ausgetauscht wurde, da an dieser Stelle kein korrespondierender Konservierungswert vorhanden war (also ein Gap in `aln3`). Dies ist bei der Konservierung von Frankreich beispielsweise dreimal der Fall.

Für diese Aminosäurepositionen gibt es keinen Wert, daher müssen sie ausgeschnitten werden.

Für jedes Residuum wird geschaut, ob es sich um eine Aminosäure handelt, dann werden ihre Atome nach dem CA-Atom untersucht. Dessen B-Faktor wird dann geprüft, Konservierungswerte sind immer Werte zwischen 0 und 1, B-Faktoren sind immer größer. Durch diese Bedingung lässt sich entscheiden, welche Residuen ausgeschnitten, also von dem zugehörigen Proteinkette losgelöst werden müssen.

Die Funktion `detach _ child` überspringt jedoch nach einmaligem Ausführen das nächste Residuum, weshalb dieser Block zweimal ablaufen muss, um sicherzustellen, dass alle Stellen entfernt wurden.

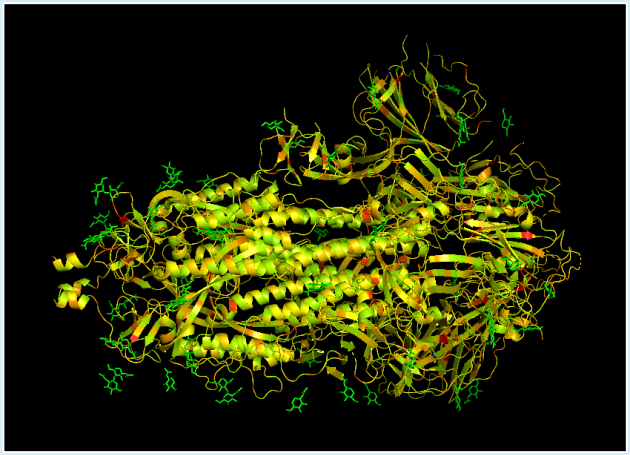


Abb. 11: Spikeprotein mit farblicher Markierung der Konservierung in Frankreich

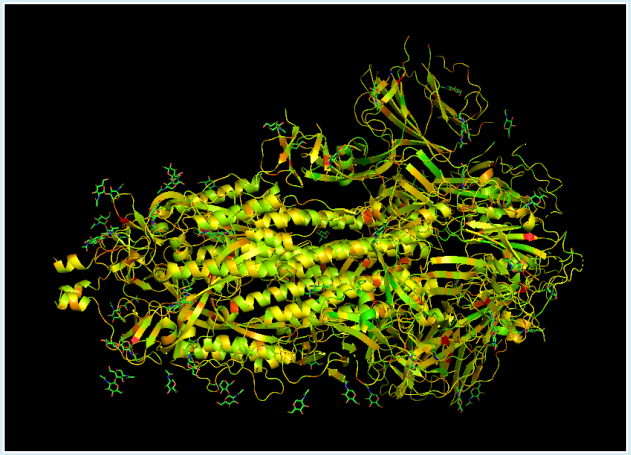


Abb. 12: Spikeprotein mit farblicher Markierung der Konservierung in Deutschland

doi: 10.7795/320.202309

Die Chains A, B und C der entsprechenden PDB-Datei eignen sich, um einen grafischen Vergleich darzustellen [11], deshalb läuft das Programm für diese drei Chains ab, die sich dann einfärben lassen. Anschließend wird die neu bearbeitete PDB-Datei ausgegeben.

Die Änderung des B-Faktors benötigt 0,21875 Sekunden, ca. 4 % der Gesamtlaufzeit von 6, 3125 Sekunden.

3. Ergebnisse

3.1 Visualisierung

Lädt man die ausgegebene PDB-Datei in einem Visualisierungsprogramm wie PyMOL und lässt die B-Faktoren farblich darstellen, ergeben sich die Abb. 11 und Abb. 12. Diese sind sich sehr ähnlich, woraus man schließen kann, dass die Varianz in den beiden Ländern ähnlich hoch ist. Die Vermutung, die Viren in Deutschland und Frankreich seien sich sehr ähnlich, kann man damit bestätigen.

Wie man im Boxplot (Abb. 13) sehen kann, gibt es jedoch eine kleine Abweichung zwischen Deutschland und Frankreich: Der Median in Frankreich ist etwas höher, es sind also mehr Positionen besser konserviert und somit nicht so häufig verändert.

Betrachtet man die Proteinstruktur mit der Einfärbung, sieht man jedoch, dass die Lage der konservierten und nicht konservierten Stellen grundsätzlich bei beiden Proteinen übereinstimmt, die am wenigsten veränderten und wichtigsten Positionen sind also bei beiden gleich.

3.2 Ergebnisdiskussion

Wie bereits vermutet, sind sich die beiden Konservierungen relativ ähnlich, die einzelnen Aminosäuren sind also gleich gut erhalten. Zwischenzeitlich sah es jedoch nicht so aus, als bestünde eine große Ähnlichkeit. Durch dieses Projekt sind die Sequenzen durch die Berücksichtigung der Konservierung vergleichbar geworden. Die Unterschiede in der Sequenzierung, die auf experimentelle Unterschiede in verschiedenen Laboren zurückgehen, sind mit meiner Methode irrelevant geworden.

Technisch gesehen gab es Probleme beim Austauschen der B-Faktoren an den CA-Atomen, was dadurch behoben wurde, dass nur noch die Aminosäure-Residuen betrachtet wurden. Das Auftauchen der Lücken im Alignmentobjekt der Konservierungswerte wurde schlussendlich durch das Herausschneiden der überflüssigen Werte gelöst. Bei der Visualisierung wurden für eine optimale Darstellung Residuen ohne Konservierungswert weggelassen.

Schlussendlich waren die deutschen und französischen Konservierungen gut visualisierbar und vergleichbar. Um weitere Erkenntnisse zu erhalten, könnte man in einer Untersuchung Daten aus anderen Ländern (innerhalb und außerhalb Europas) einbeziehen, sowie unterschiedliche Zeiträume getrennt voneinander betrachten.

Abschließend kann man festhalten, dass mit der hier vorgelegten Methode neue Coronavirus-Mutationen optisch aufgezeigt werden können.

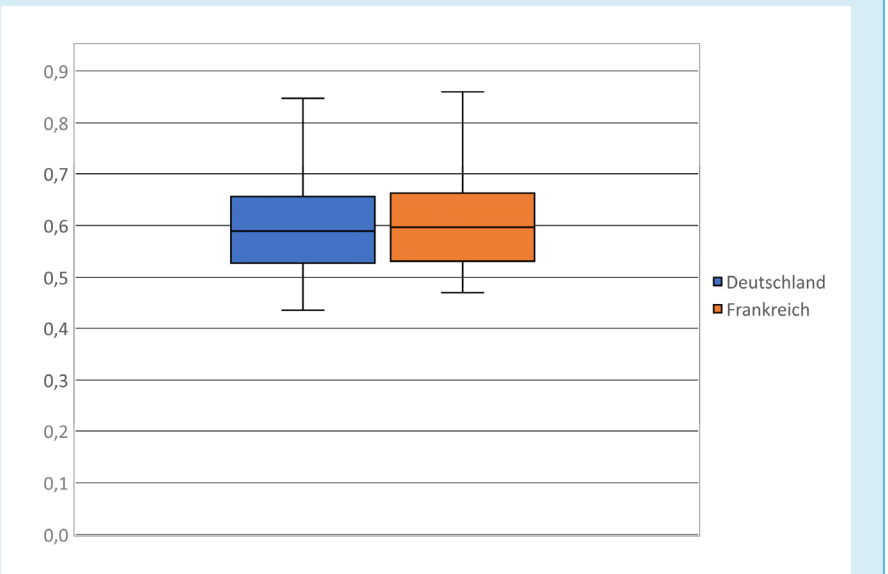


Abb. 13: Boxplot der Konservierung

doi: 10.7795/320.202309

Danksagung

Mein Dank geht an folgende Personen, die mich bei der Erstellung der Arbeit unterstützt haben:

Prof. Dr. Olga Kalinina, Bioinformatikerin, Helmholtz Institut für Pharmazeutische Forschung Saarland, hat mir bei der Themenwahl geholfen und neuartige Vorgänge erklärt.

Fawaz Dabagghie, Doktorand der Bioinformatik, Helmholtz Institut für Pharmazeutische Forschung Saarland, stand immer bei Rückfragen zur Verfügung und half mir, Komplikationen der PDB-Datei auszubügeln.

Prof. Dr.-Ing. Damian Weber, Studiendekan, htw Saarbrücken, hat mich bei der Verbesserung der Klarheit der Darstellung meiner Arbeit unterstützt.

Quellenverzeichnis

[1] Zhoutong Sun, Qian Liu, Ge Qu, Yan Feng, and Manfred T. Reetz,Utility of B-Factors in Protein Science: Interpreting Rigidity, Flexibility, and Internal Motion and Engineering Thermostability
Chemical Reviews 2019 119 (3), 1626-1665
DOI: 10.1021/acs.chemrev.8b00290

[2] C-alpha Atom, https://wiki.cmbi.umcn.nl/wiki/index.php/C-alpha, zuletzt eingesehen am 02.04.2022

[3] Clustal Omega, Multiple Sequence Alignment, https://www.ebi.ac.uk/Tools/msa/clustalo/, zuletzt besucht am 12.02.2022

[4] GISAID Datenbank, https://gisaid.org/, zuletzt eingesehen im Januar 2021

[5] Konsensussequenz, https://flexikon.doccheck.com/de/Konsensussequenz, zuletzt besucht am 12.02.2022

[6] Konservierung, https://flexikon.doccheck.com/de/Konservierung_(Biologie), besucht 12.02.2022

[7] Lexikon der Biologie, Spektrum, Sequenzalignment, https://www.spektrum.de/lexikon/biologie/sequenz-alignment/61121, zuletzt besucht am 12.02.2022

[8] PDB-Datei Aufbau, Abbildung, https://biopython.org/wiki/The_Biopython_Structural_Bioinformatics_FAQ

[9] PDB-Dateiformat, https://www.wwpdb.org/documentation/file-format-content/format33/sect9.html#MODEL
zuletzt eingesehen am 24.10.2023

[10] Protein Data Bank, https://www.rcsb.org/docs/general-help/organization-of-3d-structures-in-the-protein-data-bank,
zuletzt eingesehen am 10.01.2023

[11] Protein Data Bank, Structure of the SARS-CoV-2 spike glycoprotein
https://www.rcsb.org/structure/6vxx, zuletzt besucht am 12.02.2022

[12] PyMOL, https://pymol.org/2/, zuletzt eingesehen am 23.03.2023

[13] Residuen, https://flexikon.doccheck.com/de/Residuum_(Biochemie), besucht 12.02.2022

[14] Valdar, „Scoring Residue Conservation“, PROTEINS: Structure, Function, and Genetics 48:227–241 (2002), S. 234

Publiziere auch Du hier!

Forschungsarbeiten von Schüler/Inne/n und Student/Inn/en

In der Jungen Wissenschaft werden Forschungsarbeiten von SchülerInnen, die selbstständig, z. B. in einer Schule oder einem Schülerforschungszentrum, durchgeführt wurden, veröffentlicht. Die Arbeiten können auf Deutsch oder Englisch geschrieben sein.

Wer kann einreichen?

SchülerInnen, AbiturientInnen und Studierende ohne Abschluss, die nicht älter als 23 Jahre sind.

Was musst Du beim Einreichen beachten?

Lies die Richtlinien für Beiträge. Sie enthalten Hinweise, wie Deine Arbeit aufgebaut sein soll, wie lang sie sein darf, wie die Bilder einzureichen sind und welche weiteren Informationen wir benötigen. Solltest Du Fragen haben, dann wende Dich gern schon vor dem Einreichen an die Chefredakteurin Sabine Walter.

Lade die Erstveröffentlichungserklärung herunter, drucke und fülle sie aus und unterschreibe sie.

Dann sende Deine Arbeit und die Erstveröffentlichungserklärung per Post an:

Chefredaktion Junge Wissenschaft
Dr.-Ing. Sabine Walter
Paul-Ducros-Straße 7
30952 Ronnenberg
Tel: 05109 / 561508
Mail: sabine.walter@verlag-jungewissenschaft.de

Wie geht es nach dem Einreichen weiter?

Die Chefredakteurin sucht einen geeigneten Fachgutachter, der die inhaltliche Richtigkeit der eingereichten Arbeit überprüft und eine Empfehlung ausspricht, ob sie veröffentlicht werden kann (Peer-Review-Verfahren). Das Gutachten wird den Euch, den AutorInnen zugeschickt und Du erhältst gegebenenfalls die Möglichkeit, Hinweise des Fachgutachters einzuarbeiten.

Die Erfahrung zeigt, dass Arbeiten, die z. B. im Rahmen eines Wettbewerbs wie **Jugend forscht** die Endrunde erreicht haben, die besten Chancen haben, dieses Peer-Review-Verfahren zu bestehen.

Schließlich kommt die Arbeit in die Redaktion, wird für das Layout vorbereitet und als Open-Access-Beitrag veröffentlicht.

Was ist Dein Benefit?

Deine Forschungsarbeit ist nun in einer Gutachterzeitschrift (Peer-Review-Journal) veröffentlicht worden, d. h. Du kannst die Veröffentlichung in Deine wissenschaftliche Literaturliste aufnehmen. Deine Arbeit erhält als Open-Access-Veröffentlichung einen DOI (Data Object Identifier) und kann von entsprechenden Suchmaschinen (z. B. BASE) gefunden werden.

Die Junge Wissenschaft wird zusätzlich in wissenschaftlichen Datenbanken gelistet, d. h. Deine Arbeit kann von Experten gefunden und sogar zitiert werden. Die Junge Wissenschaft wird Dich durch den Gesamtprozess des Erstellens einer wissenschaftlichen Arbeit begleiten – als gute Vorbereitung auf das, was Du im Studium benötigst.

doi: 10.7795/320.202309

Richtlinien für Beiträge

Für die meisten Autor/Inn/en ist dies die erste wissenschaftliche Veröffentlichung. Die Einhaltung der folgenden Richtlinien hilft allen – den Autor/innen/en und dem Redaktionsteam

Die Junge Wissenschaft veröffentlicht Originalbeiträge junger AutorInnen bis zum Alter von 23 Jahren.

- Die Beiträge können auf Deutsch oder Englisch verfasst sein und sollten nicht länger als 15 Seiten mit je 35 Zeilen sein. Hierbei sind Bilder, Grafiken und Tabellen mitgezählt. Anhänge werden nicht veröffentlicht. Deckblatt und Inhaltsverzeichnis zählen nicht mit.

- Formulieren Sie eine eingängige Überschrift, um bei der Leserschaft Interesse für Ihre Arbeit zu wecken, sowie eine wissenschaftliche Überschrift.

- Formulieren Sie eine kurze, leicht verständliche Zusammenfassung (maximal 400 Zeichen).

- Die Beiträge sollen in der üblichen Form gegliedert sein, d. h. Einleitung, Erläuterungen zur Durchführung der Arbeit sowie evtl. Überwindung von Schwierigkeiten, Ergebnisse, Schlussfolgerungen, Diskussion, Liste der zitierten Literatur. In der Einleitung sollte die Idee zu der Arbeit beschrieben und die Aufgabenstellung definiert werden. Außerdem sollte sie eine kurze Darstellung schon bekannter, ähnlicher Lösungsversuche enthalten (Stand der Literatur). Am Schluss des Beitrages kann ein Dank an Förderer der Arbeit, z. B. Lehrer und Sponsoren, mit vollständigem Namen angefügt werden. Für die Leser kann ein Glossar mit den wichtigsten Fachausdrücken hilfreich sein.

- Bitte reichen Sie alle Bilder, Grafiken und Tabellen nummeriert und zusätzlich als eigene Dateien ein. Bitte geben Sie bei nicht selbst erstellten Bildern, Tabellen, Zeichnungen, Grafiken etc. die genauen und korrekten Quellenangaben an (siehe auch Erstveröffentlichungserklärung). Senden Sie Ihre Bilder als Originaldateien oder mit einer Auflösung von mindestens 300 dpi bei einer Größe von 10 · 15 cm! Bei Grafiken, die mit Excel erstellt wurden, reichen Sie bitte ebenfalls die Originaldatei mit ein.

- Vermeiden Sie aufwendige und lange Zahlentabellen.

- Formelzeichen nach DIN, ggf. IUPAC oder IUPAP verwenden. Gleichungen sind stets als Größengleichungen zu schreiben.

- Die Literaturliste steht am Ende der Arbeit. Alle Stellen erhalten eine Nummer und werden in eckigen Klammern zitiert (Beispiel: Wie in [12] dargestellt …). Fußnoten sieht das Layout nicht vor.

- Reichen Sie Ihren Beitrag sowohl in ausgedruckter Form als auch als PDF ein. Für die weitere Bearbeitung und die Umsetzung in das Layout der Jungen Wissenschaft ist ein Word-Dokument mit möglichst wenig Formatierung erforderlich. (Sollte dies Schwierigkeiten bereiten, setzen Sie sich bitte mit uns in Verbindung, damit wir gemeinsam eine Lösung finden können.)

- Senden Sie mit dem Beitrag die Erstveröffentlichungserklärung ein. Diese beinhaltet im Wesentlichen, dass der Beitrag von dem/der angegebenen AutorIn stammt, keine Rechte Dritter verletzt werden und noch nicht an anderer Stelle veröffentlicht wurde (außer im Zusammenhang mit **Jugend forscht** oder einem vergleichbaren Wettbewerb). Ebenfalls ist zu versichern, dass alle von Ihnen verwendeten Bilder, Tabellen, Zeichnungen, Grafiken etc. von Ihnen veröffentlicht werden dürfen, also keine Rechte Dritter durch die Verwendung und Veröffentlichung verletzt werden. Entsprechendes Formular ist von der Homepage www.junge-wissenschaft.ptb.de herunterzuladen, auszudrucken, auszufüllen und dem gedruckten Beitrag unterschrieben beizulegen.

- Schließlich sind die genauen Anschriften der AutorInnen mit Telefonnummer und E-Mail-Adresse sowie Geburtsdaten und Fotografien (Auflösung 300 dpi bei einer Bildgröße von mindestens 10 · 15 cm) erforderlich.

- Neulingen im Publizieren werden als Vorbilder andere Publikationen, z. B. hier in der Jungen Wissenschaft, empfohlen.

doi: 10.7795/320.202309

Impressum

Junge Wissenschaft
c/o Physikalisch-Technische
Bundesanstalt (PTB)
www.junge-wissenschaft.ptb.de

Redaktion
Dr. Sabine Walter, Chefredaktion
Junge Wissenschaft
Paul-Ducros-Str. 7
30952 Ronnenberg
E-Mail: sabine.walter@verlag-jungewissenschaft.de
Tel.: 05109 / 561 508

Verlag
Dr. Dr. Jens Simon,
Pressesprecher der PTB
Bundesallee 100
38116 Braunschweig
E-Mail: jens.simon@ptb.de
Tel.: 0531 / 592 3006
(Sekretariat der PTB-Pressestelle)

Design & Satz
Sebastian Baumeister
STILSICHER – Grafik & Werbung
E-Mail: baumeister@stilsicher.design
Tel.: 05142 / 98 77 89