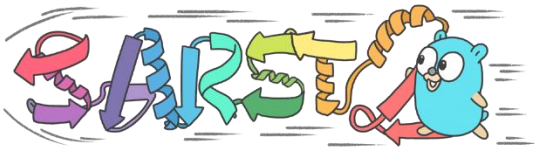
SARST2 Benutzerhandbuch

Deutsche Version



Inhaltsverzeichnis

1. Über diese Software

SARST2 (Structural Similarity Search Aided by Ramachandran Sequential Transformation, Version 2) ist ein Hochleistungsalgorithmus zum Abgleich von Proteinstrukturen. Das Programm unterstützt sowohl Datenbanksuchen nach struktureller Ähnlichkeit anhand eines angegebenen Abfrageproteins (Query) als auch paarweise Strukturvergleiche zwischen zwei Proteinstrukturen.

Diese Software wird zusammen mit der folgenden wissenschaftlichen Publikation veröffentlicht und regelmäßig über die in der Publikation angegebenen URLs aktualisiert:

Titel	SARST2, a high-throughput protein structure alignment algorithm for searching massive databases
Autoren	Wei-Cheng Lo*, Arieh Warshel, Chia-Hua Lo, Chia Yee Choke, Yan-Jie Li, Shih-Chung Yen, Jyun-Yi Yang and Shih-Wen Weng
Institut	Institute of Bioinformatics and Systems Biology, National Yang Ming Chiao Tung University, Hsinchu, Taiwan, Republic of China

*Korrespondenzautor (WadeLo@nycu.edu.tw)

Download-URLs:
https://github.com/NYCU-10lab/sarst
https://10lab.ceb.nycu.edu.tw/sarst2

2. Download, Dekomprimierung und Installation

Die neueste Version des SARST2-Programms sowie die vorgefertigten Ziel-Datenbanken stehen unter den oben aufgeführten URLs zum Download bereit.

Nach dem Herunterladen eines komprimierten Archivs können Sie die Dateien je nach Archivformat mit den Programmen tar, gzip oder zip extrahieren. Nach dem Entpacken liegen die SARST2-Programmdateien als vorkompilierte, ausführbare Binärdateien vor und erfordern keine separate Installation.

Beispiel: Wenn Sie das Archiv SARST2-v2.0.30-Linux.x86_64.tar.gz herunterladen, können Sie es unter Linux mit folgendem Befehl entpacken:

```
tar xfp SARST2-v2.0.30-Linux.x86_64.tar.gz
```

Nach dem Entpacken können Sie das SARST2-Programm mit den folgenden Befehlen ausführen:

```
cd SARST2-v2.0.30-Linux.x86_64/bin
chmod +x sarst2
./sarst2 -h
```

3. Inhalt der Software-Verzeichnisse

bin	64-Bit-Programme für Linux, Windows und macOS.
dat	Beispiel-PDB- und SCOP-Testdateien.
doc	Benutzerhandbücher in mehreren Sprachen.

4. Kurzanleitung

Dieses Softwarepaket enthält drei Hauptprogramme, die im Folgenden beschrieben werden:

sarst2	Implementierung des SARST2-Algorithmus zum Abgleich von Proteinstrukturen.
formatdb	Ein Tool zur Datenbankformatierung, mit dem Benutzer eigene Ziel-Datenbanken für Strukturrecherche und -vergleich erstellen können.
readdb	Ein Tool zum Extrahieren der Aminosäuresequenzen und linear codierten Strukturesequenzen von Proteinen aus einer vorformatierten Ziel-Datenbank.

Das Ausführen eines der oben genannten Programme ohne Parameter (oder mit -h) zeigt eine kurze textbasierte Hilfeseite für das jeweilige Programm an.

Linux, macOS

```
./sarst2
./formatdb
./readdb
./sarst2 -h
./formatdb -h
./readdb -h
```

Windows (Cmd oder PowerShell)

```
.\sarst2.exe
.\formatdb.exe
.\readdb.exe
.\sarst2.exe -h
.\formatdb.exe -h
.\readdb.exe -h
```

5. Handbuch: sarst2

5.1 Verwendung

```
./sarst2    Abfrage-Struktur    Zielstruktur(en)    [Optionen]
            ---------------     ----------------
            > eine PDB-/CIF-Datei > Zielstruktur(en) kann sein:
                                  1. -db + eine vorformatierte Datenbank
                                  2. eine Liste von PDB-/CIF-Dateien
                                  3. Verzeichnisse mit PDB-/CIF-Dateien
                                  4. eine einzelne PDB-/CIF-Datei (Paarvergleich)
```

5.2 Programmoptionen

-db	[str]	Ziel-Datenbank der zu durchsuchenden Proteinstrukturen. (Standard: none)
-brief	[int]	Anzahl der Zielstrukturen, für die eine einzeilige Zusammenfassung angezeigt wird. (Standard: 500)
-detail	[int]	Anzahl der Zielstrukturen, für die detaillierte Ausrichtungsdaten angezeigt werden. (Standard: 500)
-t	[int]	Anzahl der Threads. Muss ≥ 0 sein; bei 0 werden alle Prozessoren verwendet. (Standard: 0, alle Prozessoren)
-w	[int]	Word-Größe. (Standard: 5)
-orderby	[int]	Sortierung der Trefferliste nach einem der folgenden Faktoren: 1: Conf-Score--, 2: TM-Score--, 3: Sequenzidentität-- oder 4: RMSD++, wobei --/++ für absteigende/aufsteigende Reihenfolge steht (Standard: 1, Conf-score--) (bei paarweisem Abgleich ignoriert)
-mode	[int]	Such-/Alignierungsmodus, 1: genau, 2: ausgewogen, 3: schnell, andere Werte: auto (Datenbanksuche) oder wie 1: genau (paarweises Alignment). (Standard: auto für Datenbanksuche; 1 für paarweises Alignment)
-f	[int]	Minimale Filter aktivieren, 0: aus, 1: ein, oder sonst: auto. (Standard: auto) (bei paarweisem Abgleich immer 0, deaktiviert)

-C	[float]	Schwellenwert für den Conf-Score (Confidence Score). Muss zwischen 0 und 1 liegen; bei 0 wird kein Schwellenwert angewendet. (Standard: 0.5) (bei paarweisem Abgleich immer deaktiviert)
-pC	[float]	Schwellenwert für den finalen pC-Wert, d. h. -log2(C). Muss ≥ 0 sein; bei 0 wird kein Schwellenwert angewendet. (Standard: 1.0, entspricht -C = 0.5) (bei paarweisem Abgleich immer deaktiviert)
-e	[float]	Schwellenwert für den pC-Wert, angewendet in jedem Filter- und Verfeinerungsschritt. Bei gleichem -e und -pC verwirft -e mehr irrelevante Treffer. Muss ≥ 0 sein; bei 0 wird kein Schwellenwert angewendet. (Standard: 1.0) (bei paarweisem Abgleich immer deaktiviert)
-tmcut	[float]	TM-Score-Schwellenwert. Muss ≥ 0 sein; bei 0 wird kein Schwellenwert angewendet. Ein TM-Score ≥ 0.7 gemäß SARST2 deutet möglicherweise auf eine Homologie auf Ebene einer Protein-Familie hin. (Standard: 0.15) (bei paarweisem Abgleich immer deaktiviert)
-mem	[T/F]	Puffert alle Daten der Zielproteine im Speicher. (Standard: T) (bei paarweisem Abgleich immer T, aktiviert)
-q	[T/F]	Schneller Ausgabestil. Gibt die Ergebnisse in einem vereinfachten und parserfreundlichen Format aus. (Standard: F)
-sa	[T/F]	Gibt die strukturbasierte Sequenzausrichtung aus. (Standard: T)
-mat	[T/F]	Gibt die Transformationsmatrix für die Überlagerung aus. (Standard: F)
-nmsbj	[T/F]	Normalisiert den TM-Score basierend auf der Größe der Zielstruktur. (Standard: F)
-nmavg	[T/F]	Normalisiert den TM-Score basierend auf der durchschnittlichen Größe der Abfrage- und Zielstruktur. (Standard: F)
-nmusr	[float]	Protein-Größe zur Normalisierung des TM-Scores. Sie sollte ≥ der minimalen Größe der beiden Strukturen sein; andernfalls kann der TM-Score > 1 werden.

-d	[float]	d_0-Wert zur Skalierung des TM-Scores, z. B. 5,0 Ångström (Å).
-ml	[T/F]	Maschinelles Lernen anwenden. (Standard: T) (bei paarweisem Abgleich immer F, deaktiviert)
-fdp	[str]	Algorithmus für dynamische Programmierung in den Filter-Schritten. Unterstützte Optionen: NW (Needleman-Wunsch), SW (Smith-Waterman) (Standard: NW)
-rdp	[str]	Algorithmus für dynamische Programmierung im Verfeinerungsschritt. Unterstützte Optionen: NW (Needleman-Wunsch), SW (Smith-Waterman) (Standard: NW)
-swp	[str]	Pfad zur benutzerdefinierten Swap-Datei. Die Verwendung einer Swap-Datei kann den Speicherbedarf reduzieren. (Standard: none)
-Sout	[str]	Ordner für die Ausgabe der Überlagerungsdateien (Structure Superimposition). Der Ordner wird erstellt, falls er nicht existiert. Die Anzahl der Überlagerungsdateien wird durch die Option -detail begrenzt (Standard: none)
-html	[str]	Erstellung eines HTML-Ausgabeordners. Überlagerungsdateien der Strukturen werden ebenfalls im HTML-Ordner erzeugt. (Standard: none)
-jsmol	[str]	Pfad zum JSmol-JavaScript-Paket für die Anzeige von überlagerten Strukturen im HTML-Ausgabedokument. Kann ein lokaler Ordner oder eine HTTP(S)-URL sein. (Standard: none) (Test-URL: "https://10lab.ceb.nycu.edu.tw/ext/jsmol")
-pssm_out	[str]	Datei zum Speichern der PSSMs der Struktur- und Sequenzcodes, die in diesem Algorithmus verwendet werden. (Standard: none)
-pssm_pC	[float]	Grenzwert des pC-Werts für die PSSM-Erstellung. (Standard: 0.05)
-h		Die Hilfemeldung anzeigen (Kurzanleitung).

5.3 Beispiel: Datenbanksuche nach struktureller Ähnlichkeit

Die Abfragestruktur gegen eine vorformatierte Ziel-Datenbank durchsuchen

```
./sarst2 Qry.pdb -db my_db/my_proteins.db -brief 10 -w 7 -e 0.1
./sarst2 Qry.pdb -db my_db/my_proteins.db -brief 10 -d 5.0 -sa F
```

In dem obigen Beispiel befindet sich die Zieldatenbank im Ordner "my_db", und "my_proteins.db" ist der Dateiname ohne Erweiterung der Zieldatenbankdateien. Siehe das **Handbuch: formatdb** für Anweisungen zur Erstellung Ihrer eigenen Zieldatenbank.

5.4 Beispielverwendung: One-against-all-Ausrichtungen

Die Abfragestruktur gegen aufgelistete Zielstrukturen durchsuchen

```
./sarst2 Qry.pdb Sbj1.pdb Sbj2.cif Sbj3.pdb -mat T
```

Die Abfragestruktur gegen Zielstrukturdateien mit Wildcard-Mustern durchsuchen

```
./sarst2 Qry.pdb "set1/*.pdb" "set2/1a???.cif" -nmavg T
```

In diesem Beispiel sind "set1/*.pdb" und "set2/1a???.cif" in Anführungszeichen gesetzt und enthalten Wildcard-Zeichen. Das Programm sarst2 erweitert diese Wildcard-Muster automatisch und ermittelt intern die entsprechenden Dateinamen. Wenn die Muster nicht in Anführungszeichen gesetzt sind, übernimmt das Betriebssystem die Erweiterung der Wildcards. Bei einer großen Anzahl passender Dateien kann die resultierende Befehlszeilenargumentliste die Systemgrenzen überschreiten und dazu führen, dass der Befehl fehlschlägt. Daher empfehlen wir, Wildcard-Muster in Anführungszeichen zu setzen, damit sarst2 die Dateiliste intern verarbeitet, anstatt sich auf das Standardverhalten des Betriebssystems zu verlassen.

Die Abfragestruktur gegen mehrere Ordner mit Zielstrukturen durchsuchen

```
./sarst2 Qry.pdb set1 set2 -nmavg T
```

In diesem Beispiel sind set1 und set2 Ordner, die Proteinstrukturdateien enthalten können. Das Programm sarst2 ruft automatisch alle Dateien in diesen Ordnern ab (entspricht set1/* und set2/*). Dateien im PDB- oder CIF-Format werden ausgewählt und gegen die Abfragestruktur ausgerichtet.

5.5 Beispielverwendung: Paarweises Strukturausrichten

Die Abfragestruktur mit einer Zielstruktur ausrichten

```
./sarst2 Qry.pdb Sbj.pdb -sa F
```

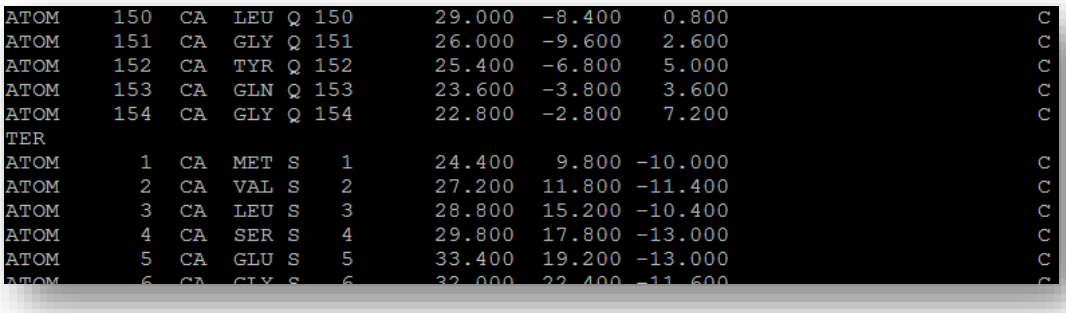
```
./sarst2 Qry.pdb Sbj.cif -mat T
```

5.6 Ausgabe von überlagerten Proteinstrukturen

Generierung von Überlagerungsdateien (PDB) für Abfrage- und Zielstrukturen

```
./sarst2 Qry.pdb -db prot/myDb -detail 100 -Sout output_folder
./sarst2 Qry.pdb "set1/*.cif" -detail 100 -Sout output_folder
```

Mit der Option -Sout output_folder werden die überlagerten Proteinstrukturen im PDB-Format in den vom Benutzer angegebenen Ordner ausgegeben. Die Anzahl der erzeugten Überlagerungsdateien wird durch die Option -detail festgelegt. Jede Ausgabedatei wird als Qry-SbjSN.pdb benannt, wobei SN die laufende Nummer der Zielstruktur in der Trefferliste darstellt. In jeder Überlagerungsdatei sind die Ketten-IDs der Abfrage- und der Zielstruktur mit Q bzw. S gekennzeichnet. Die beiden Ketten werden durch einen TER-Datensatz getrennt, wie unten dargestellt:



Wie in der Abbildung gezeigt, enthalten die Überlagerungsdateien nur Cα-Atome (Alpha-Kohlenstoff, Cα). Dies liegt daran, dass SARST2 alle Berechnungen ausschließlich auf den Cα-Koordinaten durchführt. Die Orientierung der Abfragestruktur bleibt in allen generierten Überlagerungsdateien konsistent erhalten, während jede Zielstruktur gemäß dem Alignment zur Abfragestruktur transformiert (rotiert und translatiert) wird, um die Überlagerung zu erzielen.

Zur Visualisierung der überlagerten Strukturen empfehlen wir die Verwendung von RasMol oder RasWin (http://www.openrasmol.org/). Da die Überlagerungsdateien ausschließlich Cα-Atome enthalten, sollte der Anzeigemodus in RasMol auf "backbone" eingestellt werden.

5.7 Interaktive HTML-Ergebnisseiten erzeugen

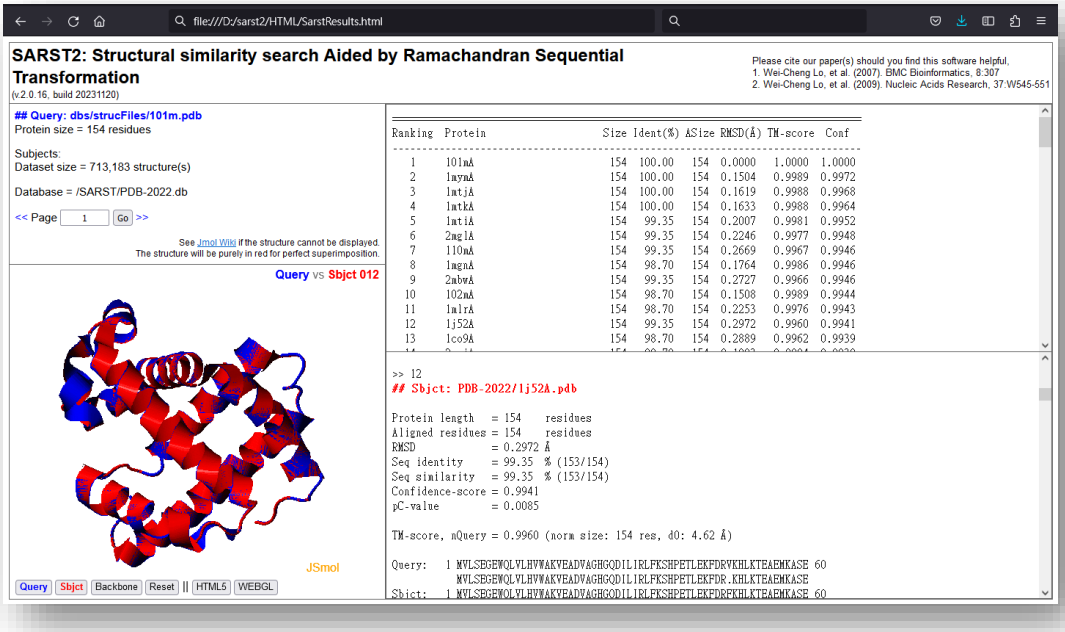
Ein HTML-Dokument mit Online-JSmol-Skripten erzeugen

```
./sarst2 Qry.pdb -db my_db/my_proteins.db -html output_folder
-jsmol "https://10lab.ceb.nycu.edu.tw/ext/jsmol"
```

Ein HTML-Dokument mit lokalem JSmol-Skriptordner (Windows) erzeugen

```
./sarst2 Qry.pdb "set1/*.cif" -detail 100 -html output_folder
-jsmol "file:///D:/software/jsmol"
```

Mit der Option "-html output_folder" werden ein HTML-Ergebnisdokument sowie die zugehörigen überlagerten Proteinstrukturdateien im angegebenen Ordner erzeugt. Die Option "-html" muss zusammen mit "-jsmol" verwendet werden, wobei "-jsmol" die URL oder den lokalen Pfad des JSmol-Pakets (Version 2013) angibt. JSmol ist ein interaktiver 3D-Viewer für molekulare Strukturen, der in Webbrowsern ausgeführt wird und von den meisten modernen Browsern unterstützt wird.



Die Hauptdatei im HTML-Ausgabeordner ist SarstResults.html, die in einem Webbrowser geöffnet werden sollte. Weitere HTML-Dateien sind als Inline-Frames in die Hauptseite eingebettet. Außerdem wird ein Unterordner mit dem Namen "sup" erstellt. Darin werden die Überlagerungsdateien gespeichert, die die Abfrage- und die jeweilige Zielstruktur aus der Trefferliste enthalten.

Je nach Betriebssystem, Browser oder Antivirensoftware müssen Sie möglicherweise die Sicherheitseinstellungen anpassen, um dem Browser das Ausführen von JavaScript und den Zugriff auf die überlagerten Strukturdateien im Ordner "sup" zu erlauben, sodass der JSmol-3D-Viewer korrekt funktioniert.

6. Handbuch: formatdb

6.1 Verwendung

```
./formatdb      Zielstruktur(en)    -db Datenbank    [Optionen]
                ----------------
                > Zielstruktur(en) kann sein:
                1. eine Liste von PDB-/CIF-Dateien
                2. Verzeichnisse mit PDB-/CIF-Dateien
                3. eine einfache Textdatei mit einer Liste von
                   PDB-/CIF-Dateipfaden
```

6.2 Programmoptionen

-db	[str]	Ziel-Datenbank der zu erstellenden Proteinstrukturen. (Standard: none)
-flist	[str]	Einfache Textdatei mit einer Liste von PDB-/CIF-Dateipfaden. Dieses Argument kann zusammen mit den üblichen Argumenten für Zielstrukturen verwendet werden. (Standard: none)
-t	[int]	Anzahl der Threads. Muss ≥ 0 sein; bei 0 werden alle Prozessoren verwendet. (Standard: 0, alle Prozessoren)
-split	[int]	Datenbank in Teilmengen aufteilen, wobei jede Teilmenge die in dieser Option angegebene Anzahl von Zielstrukturen enthält. Das Aufteilen der Datenbank hilft dabei zu verhindern, dass die Datenbankdateien die maximal zulässige Dateigröße des Speichermediums überschreiten. (Standard: none)
-save_disk	[T/F]	Die Atomkoordinaten von drei auf eine Nachkommastelle runden, um Speicherplatz zu sparen. (Standard: F)
-keep_order	[T/F]	Die Reihenfolge der in der Datenbank gespeicherten Zielstrukturen beibehalten, wie sie in den Eingabeargumenten angegeben wurde. Das Setzen von T verlangsamt die Erstellung der Datenbank. (Standard: F)
-h		Die Hilfemeldung anzeigen (Kurzanleitung).

6.3 Beispielverwendungen

Eine Ziel-Datenbank aus aufgelisteten Zielstrukturdateien erstellen

```
./formatdb Sbj1.pdb Sbj2.cif Sbj3.pdb -db myDb -keep_order T
```

Es werden mehrere Datenbankdateien mit Dateinamen, die mit myDb beginnen, erstellt. Durch Aktivieren von -keep_order wird die Reihenfolge der Zielstrukturen in der Ziel-Datenbank entsprechend der Reihenfolge in den Befehlszeilenargumenten beibehalten.

Eine Ziel-Datenbank aus aufgelisteten Zielstrukturdateien mit Wildcards erstellen

```
./formatdb "set1/*.pdb" "set2/*.cif" Sbj1.pdb Sbj2.cif -db myDb
```

Beim Auflisten von Zielstrukturdateien können Argumente mit und ohne Wildcards gemischt werden. Es wird empfohlen, Wildcard-Argumente in Anführungszeichen zu setzen, damit das Programm die Dateierweiterung korrekt verarbeiten kann.

Eine Ziel-Datenbank basierend auf einer Dateiliste von Zielstrukturen erstellen

```
./formatdb -flist protlist.txt Sbj1.pdb Sbj2.cif -db myDb
```

Die Datei protlist.txt sollte eine Liste von Dateipfaden enthalten, wobei pro Zeile genau ein Dateipfad angegeben wird.

Eine Ziel-Datenbank aus Ordnern mit Zielstrukturdateien erstellen

```
./formatdb folder1 folder2 -db myDb -save_disk T -split 50000
```

Durch Aktivieren von -save_disk werden die Cα-Koordinaten auf eine Nachkommastelle gerundet, um Speicherplatz zu sparen. Die Option "-split 50000" führt dazu, dass mehrere Teilmengen-Datenbanken erstellt werden, von denen jede maximal 50000 Strukturen enthält. Diese Aufteilungsoption ist besonders nützlich, wenn die Größe der formatierten Datenbankdateien die maximal zulässige Dateigröße bestimmter Betriebssysteme oder Dateisysteme überschreiten könnte.

7. Handbuch: readdb

7.1 Verwendung

```
./readdb  Zieldatenbank    Ausgabedatei      [-seq Sequenztyp]
          ---------------  ------------      -----------
          > Muss sein:     > Wird im               > Kann sein:
            vorformatierte   FASTA-Format         1. AA
            SARST2-Datenbank ausgegeben           2. AAT
                                                  3. SARST
                                                  4. SSE
```

7.2 Programmoptionen

-seq	[str]	Der Ausgabesequenztyp. AA Aminosäuresequenz AAT Fünfsymbolige Aminosäuretyp-Sequenz SARST SARST-Ramachandran-Code-Sequenz SSE Vier-Symbol-Sekundärstrukturelement-Sequenz (Standard: AA)
-h		Die Hilfemeldung anzeigen (Kurzanleitung).

7.3 Beispielverwendungen

Zielsequenzen aus einer SARST2-Zieldatenbank extrahieren

```
./readdb my_db/my_proteins.db seqs.fasta
./readdb my_db/my_proteins.db seqs.fasta -seq SARST
./readdb my_db/my_proteins.db seqs.fasta -seq AAT
```

Wenn -seq nicht angegeben ist, wird standardmäßig die Aminosäuresequenz als Ausgabetyp verwendet. Die Ausgabedatei wird (seqs.fasta) im FASTA-Format erstellt. Falls die Ausgabedatei bereits vor dem Ausführen von readdb existiert, wird sie überschrieben.