

Projektanleitung

DNA-Analyse

1 Einleitung

1.1 Biologische Hintergründe

Die Prozesse der DNA-Transkription und -Translation sind entscheidend dafür, dass unsere Zellen funktionieren. Sie sorgen dafür, dass die in der DNA gespeicherte Erbinformation in Proteine umgesetzt wird. Zuerst wird bei der Transkription die genetische Information eines Gens von der DNA auf eine Boten-RNA (mRNA) übertragen. Anschließend wird diese mRNA bei der Translation in ein Protein übersetzt. Proteine sind für den Körper extrem wichtig – sie steuern chemische Reaktionen (Enzyme), geben Zellen ihre Struktur und ermöglichen zahlreiche lebenswichtige Prozesse.

Die DNA (Desoxyribonukleinsäure) ist das Molekül, das unsere genetische Information trägt. Ihre Grundbausteine sind die Basen Adenin (A), Cytosin (C), Guanin (G) und Thymin (T). Diese Basen formen zwei Strängen, die sich zu einer Doppelhelix winden. Diese Stränge sind komplementär, das bedeutet, dass sich die Basen Adenin (A) mit Thymin (T) und Cytosin (C) mit Guanin (G) paaren. In der Abbildung seht ihr ein Beispiel für einen solchen DNA-Strang.

A	T	G	G	C	A	T	G	A	<i>DNA(nicht – codogen)</i>
T	A	C	C	G	T	A	C	T	<i>DNA(codogen)</i>

Bei der Transkription wird nur einer der beiden DNA-Stränge als Vorlage genutzt, nämlich der codogene Strang, und von der DNA-Polymerase ausgelesen. Diese bindet an den Beginn eines geeigneten ORF (Open Reading Frame) und erzeugt eine mRNA (messenger RNA), welche komplementär zu dem DNA Strang ist. Wo ein G in der DNA steht wird ein C in der mRNA eingebaut, G wird zu C, T wird zu A und A wird zu U, da in RNA statt Thymin nur der Baustein Uracil (U) zur Verfügung steht.

A	T	G	G	C	A	T	G	A	<i>DNA(nicht – codogen)</i>
T	A	C	C	G	T	A	C	T	<i>DNA(codogen)</i>
A	U	G	G	C	A	U	G	A	<i>mRNA</i>

Bei der Translation wird die mRNA in eine Aminosäurekette übersetzt, aus der später ein funktionstüchtiges Protein entsteht. Dabei wird die mRNA in Tripletts (Codons) ausgelesen und die tRNA liefert die entsprechenden Aminosäuren. Um herauszufinden welches Triplet für welche Aminosäure kodiert gibt es die sogenannte "Code-Sonnen"[1](#). Diese wird von innen nach außen gelesen und zeigt die mRNA Sequenz.

Im Folgenden seht ihr wie sich unsere Beispielsequenz mit Hilfe der Code-Sonne in eine Sequenz aus Aminosäuren (auch Proteinsequenz genannt) übersetzt. Das Stop-Codon wird mit * gekennzeichnet.

A	T	G	G	C	A	T	G	A	<i>DNA(nicht – codogen)</i>
T	A	C	C	G	T	A	C	T	<i>DNA(codogen)</i>
A	U	G	G	C	A	U	G	A	<i>mRNA</i>
M				A		*			<i>Protein</i>

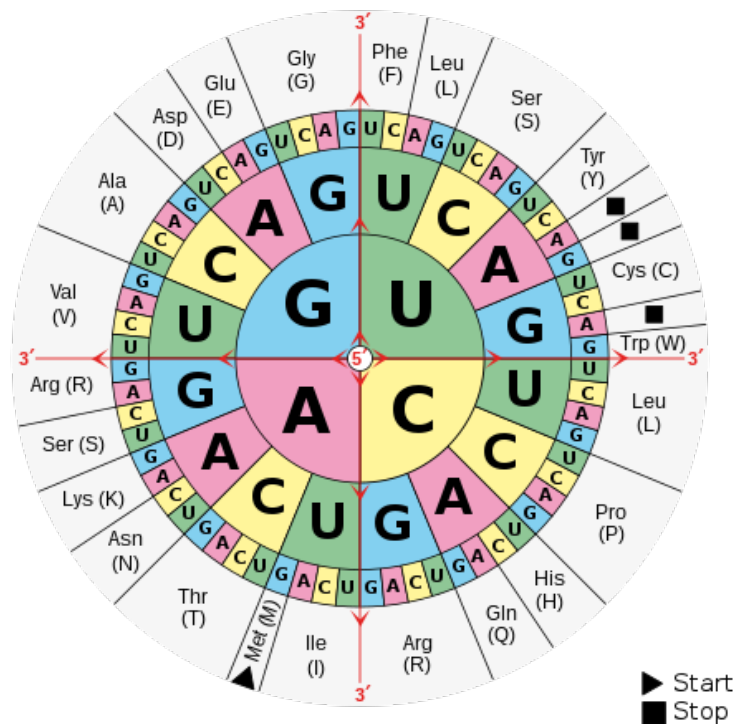


Figure 1: Quelle: [Wikipedia](#)

1.2 Daten

Ihr findet alle hier erwähnten Eingabe (englisch *input*) Daten in diesem repository: https://github.com/Jo-B-314/Infocamp_2025.

Um dieses Projekt zu bearbeiten solltet ihr die daten aus dem repository herunterladen. Dazu führt ihr folgenden Befehl an dem Ort aus, an dem die Dateien gespeichert werden sollen:

```
git clone https://github.com/Jo-B-314/Infocamp_2025.git
```

Die heruntergeladene Ordnerstruktur sieht dann wie folgt aus:

```
- Infocamp_2025
  |- input/
    |- 1980.fasta
    |- 2024.fasta
  |- mutations/
    |- ORF15_original.fa
    |- ORF15_mutations.fa
  |- Infocamp_2025_Projekt_DNA-Analyse.pdf
```

2 Aufgaben

2.1 FASTA Parser

In dieser Aufgabe sollt ihr einen FASTA Parser schreiben (weitere Infos in Abschnitt 3.1). Als Eingabe bekommt der Parser eine oder mehrere FASTA Datei(en), die auf `.fasta` oder `.fa` enden und die je eine DNA Sequenz enthalten. Ihr sollt aus dem repository, aus dem ihr diese Anleitung habt, die Dateien aus dem Ordner *input* einlesen und die Sequenzen in einer geeigneten Datenstruktur speichern. Im weiteren Verlauf der Aufgaben werdet ihr wieder damit arbeiten.

Wie eine solche FASTA Datei aussieht seht ihr hier:

```
>lcl|ORF15 CDS
ATGGGCTCTGAGACTATAAAGCCAGCGGGGGCCAGCAGCCCTCAGCCCT
CCAGGACAGGCTGCATCAGAAGAGGCCATCAAGCAGGTCTGTTCCAAGGG
CCTTTGCGTCAGGTGGGCTCAGGGTTCCAGGTGGCTGGACCCAGGCC
CAGCTCTGCAGCAGGGAGGACGTGGCTGGGCTCGTGAAGCATGTGGGGT
GAGCCAGGGGCCCCAAGGCAGGGCACCTGGCCTTCAGCCTGCCTCAGCC
CTGCCTGTCTCCAGATCACTGTCTTCTGCCATGGCCCTGTGGATGCGC
CTCCTGCCCCTGCTGGCGCTGCTGGCCCTCTGGGGACCTGACCCAGCCG
AGCCTTTGTGAACCAACACCTGTGCGGCTCACACCTGGTGGAAGCTCTCT
ACCTAGTGTGCGGGGAACGAGGCTTCTTCTACACACCAAGACCCGCCG
GAGGCAGAGGACCTGCAGGGTGAGCCAACGCCCATGCTGCCCCTGGCC
GCCCCAGCCACCCCTGCTCCTGGCGCTCCCACCCAGCATGGGCAGAAG
GGGGCAGGAGGCTGCCACCCAGCAGGGGGTCAGGTGCACTTTTAA
```

FASTA Dateien enthalten grundsätzlich immer den nicht-codogenen DNA Strang. Falls ihr euch eure Eingabe-Dateien selbst aussuchen wollt schaut am Ende, nachdem ihr mit den Aufgaben fertig seid, in Abschnitt 4 (Online-Datenbanken) vorbei. Dort findet ihr eine Anleitung um eigene FASTA Dateien herunterzuladen.

2.2 ORFfinder, Transkription und Translation

Schreibt einen "Open Reading Frame (ORF) finder", der ORFs identifizieren kann und diese in Aminosäuresequenzen übersetzt. (weitere Infos zu ORFs findet ihr in Abschnitt 3.2). Ihr müsst also

1. alle ORFs auf dem nicht-codogenen Strang finden,
2. für jeden ORF den komplementären (codogenen) Strang berechnen,
3. für jeden ORF eine mRNA berechnen (=Transkription) und
4. mit Hilfe der Code Sonne jede mRNA in eine Aminosäuresequenz (auch Proteinsequenz genannt) übersetzen (=Translation). Bitte entfernt die Stop Codons aus eurer Aminosäuresequenz.

2.3 Filter

In dieser Aufgabe werden wir Aminosäuresequenzen aussortieren. Warum wir das tun erfahrt ihr in Abschnitt 3.3.

Zunächst dürft ihr alle Aminosäuresequenzen verwerfen die kürzer als 20 Zeichen sind, da diese zu kurz sind. Danach sollt ihr eine Methode schreiben, die eure gefundenen Aminosäuresequenzen aus Datei A mit denen aus Datei B vergleicht. Sind die Sequenzen gleich oder ist eine Sequenz in der Anderen enthalten speichert ihr diese bzw. die längere Sequenz.

2.4 SmartBLAST

Kopiert die übrig gebliebenen Aminosäuresequenzen nacheinander in das Suchfeld von [SmartBLAST](#) und startet die Suche (weitere Infos zu SmartBLAST findet ihr in Abschnitt 3.4). Kopiert euch die Sequenzen, bei denen die Suche erfolgreich war in eine Textdatei. Diese Sequenzen braucht ihr für die nächste Aufgabe.

Bekommt ihr eine Meldung, dass die Suche nicht erfolgreich war dürft ihr diese Sequenz überspringen und mit der nächsten weitermachen.

2.5 AlphaFold

Ladet die Sequenz aus Aminosäuren in AlphaFold hoch und lasst euch ein 3D Modell berechnen (mehr Infos zu AlphaFold in Abschnitt 3.5). Es empfiehlt sich eine Sequenz pro Modell zu verwenden, da AlphaFold sonst gerne die Strukturen vermischt.

Ladet euch das Modell als .cif Datei herunter.

2.6 PyMol

Ladet euer Model 0 in PyMol und färbt es nach spectrum/b-factors ein (mehr Infos zu PyMol in Abschnitt 3.6). Diese Färbung zeigt euch die confidence des Models für jede Aminosäure. Ihr könnt dazu entweder die Kommandozeile in PyMol verwenden oder euch rechts durch das Menü klicken (wie das aufgebaut ist findet ihr [hier](#)).

Vergleicht euer Model nun mit einer experimentell bestimmten 3D Struktur. Dazu ladet die Stuktur mit der ID '1mso' mit dem Befehl `fetch` in euren Arbeitsbereich und färbt sie nach "chains" ein. Um die beiden Strukturen besser vergleichen zu können müsst ihr ein ALignment (to molecule) durchführen.

Optionaler Teil:

Statt euch durch PyMol zu klicken könnt ihr auch ein Python Skript schreiben, dass die oben genannten Anweisungen umsetzt.

2.7 Mutationen

Für diese Aufgabe stellen wir euch im Ordner *mutations* zwei FASTA Dateien zur Verfügung: ein Original und eine Datei mit Mutationen. Eure Aufgabe ist es die verschiedenen Mutationen, die sich eingeschlichen haben zu finden. Der Einfachheit enthält jede Datei nur einen ORF.

Um alle Arten von Mutationen zu finden müsst ihr sowohl die DNA Sequenzen als auch die Aminosäuresequenzen vergleichen (mehr zum Thema Mutationsarten findet ihr in Abschnitt 3.7). Merkt euch die Positionen der Mutationen in der Aminosäuresequenz um diese im 3D Modell zu überprüfen. Verwendet die Methoden die ihr in den vorherigen Aufgaben erstellt habt und nutzt die Tools die ihr kennengelernt habt.

Optionaler Teil:

Es bietet sich an ein Alignment der beiden Sequenzen zu berechnen, das die Sequenzen zeichenweise vergleicht. Sind die Zeichen aus beiden Ketten gleich spricht man von einem *match*, sind die Zeichen unterschiedlich von einem *mismatch*.

Gibt das Alignment für die DNA Sequenzen und die Aminosäuresequenzen auf der Konsole aus. Verbindet gleiche Buchstaben mit einem vertikalen Strich ('|'), bei unterschiedlichen Buchstaben bleibt der Zwischenraum frei. Gebt auch Positionsnummern für jede fünfte Position an. Es bietet sich an Zeilenumbrüche einzufügen oder das Alignment in einer Textdatei zu speichern (das ist euch überlassen).

Das ganz könnte dann so aussehen:

				5					10		
A	T	G	G	C	A	A	G	A	T	A	G
A	A	C	G	G	A	A	G	T	T	A	A

3 Tools und Konzepte

3.1 FASTA Parser

Ein FASTA Parser ist ein Programm oder ein Code, der verwendet wird, um [FASTA-Dateien](#) zu lesen und die darin enthaltenen Sequenzen (wie DNA, RNA oder Proteine) zu extrahieren. FASTA-Dateien bestehen aus einer Beschreibungslinie, die mit einem „>“ beginnt, und einer Sequenz, die in der folgenden Zeile steht.

3.2 ORF - open reading frames

ORFs (Open Reading Frames) sind spezifische Bereiche innerhalb eines DNA- oder RNA-Strangs, die eine Proteinkodierung ermöglichen. Ein ORF beginnt auf dem nicht-codogenen Strang mit einem Startcodon („ATG“ in der DNA) und endet mit einem Stoppcodon (z. B. „TAA“, „TAG“ oder „TGA“). Die Regionen zwischen Start- und Stoppcodon enthalten die Information, die für die Herstellung von Proteinen notwendig ist.

Die Bereiche außerhalb der ORFs sind jedoch nicht völlig nutzlos. Sie kodieren für regulatorische RNAs, sind Andockstelle für Moleküle die die Proteinherstellung regulieren oder sind für die DNA Struktur verantwortlich.

3.3 ORF Filter

Würden wir alle ORFs direkt in mRNA transkribieren wäre das biologisch falsch, da wir damit auch Bereiche verwenden, aus denen gar keine Proteine werden sollen.

Nicht alle ORFs kodieren tatsächlich Proteine, weil nicht jeder offene Leserahmen von der Zelle als funktionelles Gen genutzt wird. Einige ORFs könnten zufällig in der DNA vorkommen, ohne dass sie wirklich exprimiert werden. Außerdem gibt es regulatorische Mechanismen, die verhindern, dass bestimmte ORFs in Proteine übersetzt werden.

3.4 SmartBLAST

Um herauszufinden welche ORFs Proteine kodieren gibt es [SmartBLAST](#), ein Tool das Proteinsequenzen mit der NCBI Datenbank vergleicht.

Um SmartBLAST zu verwenden kopiert man eine Proteinsequenz oder eine Protein ID in das Suchfeld und startet die Suche. War diese erfolgreich bekommt man eine Seite mit Ergebnissen angezeigt. War die Suche nicht erfolgreich erscheint die Meldung *SmartBLAST found no matches, please try BLAST search against nr*.

3.5 AlphaFold

AlphaFold ist eine Künstliche Intelligenz (KI), die von DeepMind (einem Tochterunternehmen von Google) entwickelt wurde, um die 3D-Struktur von Proteinen vorherzusagen. Normalerweise kann die Bestimmung der Proteinstruktur im Labor Jahre dauern, aber AlphaFold schafft das in Sekunden bis Stunden (je nach Genauigkeit, Größe und Komplexität der Proteine).

AlphaFold lässt sich [hier](#) ausprobieren, allerdings ist hierfür ein Google Account notwendig (falls nicht vorhanden uns Betreuer ansprechen) und die Anzahl an Versuchen ist auf 20/Tag begrenzt.

Um einen job an AlphaFold zu senden könnt ihr links die Art eurer Sequenz auswählen (Protein, DNA,

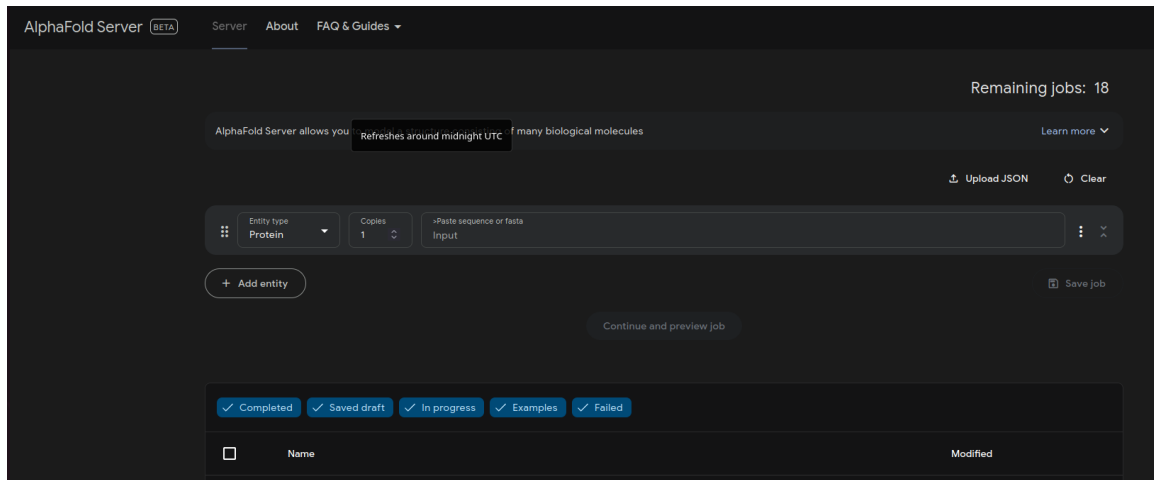


Figure 2: AlphaFold Server

RNA, Ligand, Ion) und rechts eure Sequenz in das Textfeld hinein kopieren. Hier wurde Protein ausgewählt und eine Sequenz aus Aminosäuren eingefügt.

Achtung, wenn ihr hier DNA auswählt und eure DNA Sequenz angebt bekommt ihr ein 3D Modell der Doppelhelix und nicht von dem Protein das bei Transkription und Translation herauskommt.

Sobald das Model fertig berechnet ist könnt ihr im Menü mit "open results" euch das 3D Modell anschauen. AlphaFold berechnet 5 verschiedene Modelle (nummeriert von 0 bis 4) und zeigt euch das mit der höchsten confidence (model 0) an.

Wenn ihr die Modelle herunterladet erhaltet ihr ein .zip file mit allen 5 Modellen (.cif Format) sowie verschiedenen Informationen (.json Format).

3.6 PyMol

PyMol ist ein Programm zur Visualisierung und Analyse von Molekülen, insbesondere von Proteinen und DNA-Strukturen. PyMOL bietet eine einfache Benutzeroberfläche, mit der man Moleküle drehen, zoomen und in verschiedenen Darstellungsformen anzeigen kann.

Ein besonders nützliches Feature von PyMOL ist, dass es Python-Skripte unterstützt. Das bedeutet, dass du die grafische Oberfläche von PyMOL nicht nur manuell bedienen musst, sondern auch automatisierte Befehle in Python schreiben kannst, um wiederkehrende Aufgaben effizient auszuführen. Mit Python kannst du beispielsweise mehrere Moleküle gleichzeitig bearbeiten, spezielle Berechnungen durchführen oder bestimmte Strukturen nach deinen Vorgaben einfärben. Das Skripting in PyMOL bietet somit viel mehr Flexibilität und Kontrolle bei der Analyse und Visualisierung von Molekülen. Mehr Informationen gibt's leider nur auf englisch im [Wiki](#).

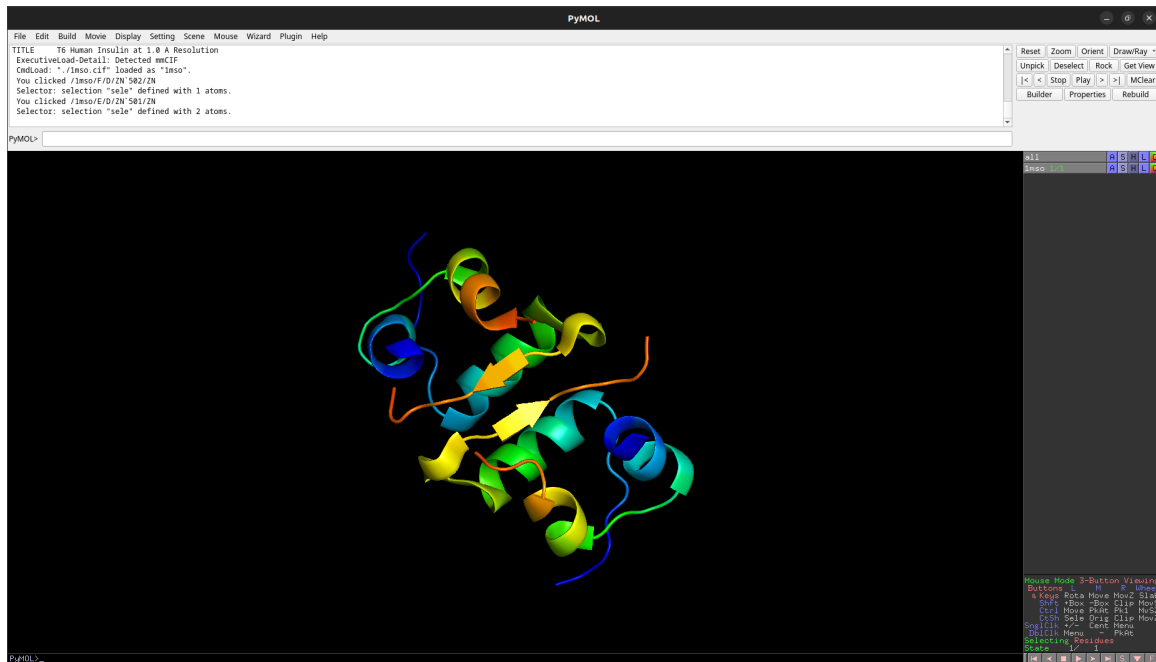


Figure 3: PyMols graphische Oberfläche

3.7 Mutationen

In der Genomexpression, also dem Prozess der Umwandlung von DNA-Sequenzen in Proteine, gibt es 3 verschiedene Arten von Mutationen:

1. *silent*
2. *missense*
3. *nonsense*

Bei allen gelisteten Mutationen handelt es sich um Einzel-Nukleotid Mutationen, es werden also nur einzelne Zeichen (Basen oder Aminosäuren) ersetzt. Mutationen können in beiden DNA Strängen oder nur in einem Strang vorkommen. Wichtig für uns sind die Mutationen, die im codogenen Strang vorkommen.

Eine *silent* Mutation findet nur in der DNA Sequenz statt, die daraus resultierende Aminosäure bleibt unverändert. Bei der *missense* Mutation verändert sich durch die Änderung in der DNA auch die Aminosäuresequenz. Die *nonsense* Mutation ist eine Sonderform der *missense* Mutation: hier wird statt irgendeiner anderen Aminosäure ein Stop Codon eingebaut.

Im unten gezeigten Beispiel sind die Mutationen in der DNA fett markiert.

<i>T</i>	<i>G</i>	<i>U</i>	<i>T</i>	<i>G</i>	C	<i>T</i>	<i>G</i>	G	<i>T</i>	<i>G</i>	A	<i>DNA(nicht-codogen)</i>
<i>A</i>	<i>C</i>	<i>A</i>	<i>A</i>	<i>C</i>	G	<i>A</i>	<i>C</i>	C	<i>A</i>	<i>C</i>	T	<i>DNA(codogen)</i>
<i>U</i>	<i>G</i>	<i>U</i>	<i>U</i>	<i>G</i>	C	<i>U</i>	<i>G</i>	G	<i>U</i>	<i>G</i>	A	<i>mRNA</i>
	<i>C</i>			<i>C</i>			<i>W</i>			*		<i>Protein</i>
			<i>silent</i>			<i>mis-</i>			<i>non-</i>			<i>Mutation</i>
						<i>sense</i>			<i>sense</i>			

4 Online-Datenbanken

Um eure Analyse nun auf anderen Proteinen als dem Beispiel aus Aufgabe 1 durchzuführen gibt es hier eine Anleitung um verschiedene Daten herunterzuladen und zu bearbeiten.

4.1 FASTA Dateien herunterladen

Die [NCBI-Datenbank](#) (National Center for Biotechnology Information) ist eine umfangreiche Sammlung von biologischen Daten, die von Genetik über Proteinstrukturen bis hin zu Sequenzen von DNA, RNA und Proteinen reicht.

Um DNA Sequenzen herunterzuladen wählt als Suchdatenbank 'Gene' aus oder klickt [hier](#). In der Suchzeile könnt ihr nun alles mögliche eingeben, beispielsweise *human hemoglobin*.

Die Suche zeigt euch zu jedem Eintrag unter anderem eine ID, aus welchem Organismus die Daten stammen und auf welchem Chromosom der Genabschnitt liegt. Wenn ihr nun einen beliebigen Eintrag auswählt landet ihr auf der dazugehörigen Übersichtsseite, wie in Abbildung 4.

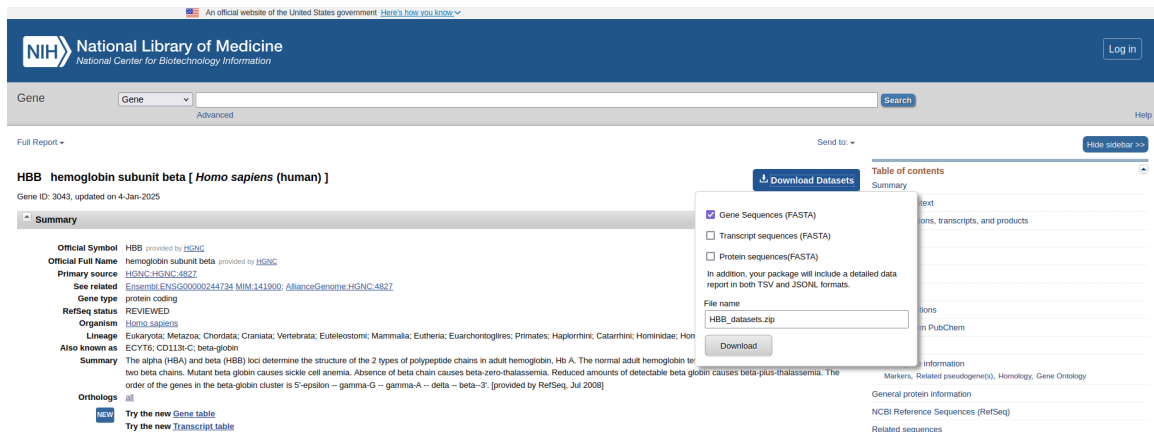


Figure 4: Zusammenfassung Hämoglobin

Dort könnt ihr entweder die Gene Sequences direkt herunterladen oder ihr scrollt zum Abschnitt "Genomic regions, transcripts, and products", dort könnt ihr euch über FASTA direkt die Sequenz anzeigen lassen und als Text kopieren.

4.2 Experimentelle Proteinstrukturen

Um Proteinstrukturen zu finden die man in PyMol laden kann empfiehlt sich die [Protein Data Bank](#) von RCSB (Research Collaboratory for Structural Bioinformatics), da diese die weltweit größte Sammlung von strukturellen Informationen über Moleküle (Proteine, DNA und RNA) enthält.

Sucht man hier nach einem Ausdruck wie *human hemoglobin* bekommt man diverse Einträge angezeigt die mit einer 4 stelligen ID abgekürzt sind (bspw. 1JY7). Diese ID kann man im Pymol command `fetch` verwenden um die Struktur zu laden.