Übungen zum Sequenzanalyse-Praktikum

Universität Bielefeld, SoSe 2015
Dr. Roland Wittler · M.Sc. Linda Sundermann
http://wiki.techfak.uni-bielefeld.de/gi/Teaching/2015summer/SequaPrak
praktikum-seqan@CeBiTec.Uni-Bielefeld.DE

Übungsblatt 7 vom 01.06.2015
Abgabe bis Donnerstag, 24:00 Uhr.

Aufgabe 1 (q-Gram-Distanz als Filter)

Vervollständige eine bereitgestellte Java-Klasse, die die q-Gram-Distanz und die Edit-Distanz zweier Strings berechnet und die Laufzeiten, die die unterschiedlichen Distanzen benötigen, misst. Die Java-Datei kannst du dir unter `http://wiki.techfak.uni-bielefeld.de/gi/Teaching/2015summer/SequaPrak` herunterladen. In den folgenden Aufgabenteilen werden fehlende Programmteile ergänzt.

1. Mache dir kurz für dein weiteres Vorgehen klar, weshalb man einem q-Gram einen Rank zuordnen sollte.

2. Ein Rank eines q-Grams an Position i kann aus dem Rank des vorhergehenden q-Grams an Position $i-1$ und dem "neuen" Buchstaben an Position $i+q-1$ berechnet werden. Warum ist diese Vorgehensweise sinnvoll? Implementiere die vorgegebene Funktion `getRank(Character c, int prevRank)`.

3. Wie lang ist ein q-Gram-Profil in Abhängigkeit von der Alphabetgröße und q? Wie groß darf q bei Alphabetgröße 4 höchstens gewählt werden, so dass der Typ `int` zur Adressierung verwendet werden kann? Vervollständige die Funktion `getProfile(String s)`, welche das q-Gram-Profil eines Strings s zurückgibt.

4. Implementiere die vorgegebene Funktion `qGramDistance(String a, String b)`, welche die Distanz zwischen zwei Strings a und b auf Basis ihrer q-Gram-Profile berechnen soll.

Nachdem nun alle `TODO`s abgearbeitet wurden, verwende das Programm um die Berechnung der q-Gram- und der Edit-Distanz zu evaluieren.

5. Verwende die Main-Methode, um die 7-Gram- und die Edit-Distanz von DNA-Sequenzen mit den Längen 1–5 kb zu berechnen. Auf der Veranstaltungsseite findest du eine entsprechende Multiple-FASTA-Datei als Eingabe. Gib die Laufzeiten an und stelle sie grafisch in einem Diagramm gegenüber. Erläutere das Laufzeitverhalten für beide Distanzmaße.

6. Berechne beide Distanzen ($q = 7$) für zwei beliebige Sequenzen, die sich nur in einer (etwa mittleren) Position unterscheiden und erläutere die Ergebnisse, insbesondere den Zusammenhang mit q.

7. Wie kann die q-Gram-Distanz als Filter für die Edit-Distanz genutzt werden? Überprüfe die theoretische Schranke an allen gemessenen Distanzen.

Aufgabe 2 (De Buijn-Graphen für Assembly)

In dieser Aufgabe werden wir ein Assembly mit *Velvet* durchführen. Velvet läuft unter Linux oder OS X, lässt sich aber auch mit Cygwin durchführen.

1. Gehe auf die Seite `http://www.ebi.ac.uk/~zerbino/velvet/` und lade dir die aktuelle Version von Velvet herunter. Kompiliere anschließend das Programm. Informationen dazu findest du in der Datei `README.txt` und `For_MAC_or_SPARC_users.txt`. Nun solltest du in dem Verzeichnis von Velvet `velvetg` und `velveth` vorfinden.

2. Informiere dich im Manual darüber, was die Aufgaben von `velveth` und `velvetg` sind. Mache dich außerdem mit der einfachen Syntax der beiden Programme bekannt.

3. Wie beeinflusst die Wahl der k-mer-Länge das Ergebnis des Assemblys? Was muss beachtet werden?

4. Teste Velvet mit fünf unterschiedlichen k-mer-Längen. Benutze dazu die Daten `data/test_reads.fa`. Rufe zuerst `velveth` auf:

   ```
   ./velveth <myDirectory> <k-mer size> -shortPaired data/test_reads.fa
   ```

   Und danach `velvetg`:

   ```
   ./velvetg <myDirectory> -cov_cutoff 10
   ```

   Vergleiche anschließend die Ergebnisse. Wie viele Knoten haben deine Graphen und wie unterscheiden sich die n50-Werte? Erkläre, wie die Ergebnisse zustande gekommen sind.

5. Wähle ein Assembly mit möglichst wenigen Contigs und vergleiche die assemblierte(n) Contigsequenz(en) mit der Referenzsequenz `data/test_reference.fa`, z.B. mit BLAST, und diskutiere kurz dein Ergebnis.