Stefan Kurtz **Universität Hamburg**
Emanuel Ehmki **Zentrum für Bioinformatik**

Programmierung in der Bioinformatik
Wintersemester 2019
Übungen zur Vorlesung: Ausgabe am 23.10.2019

Beginnen Sie die Übungen mit den folgenden Schritten (das Zeichen $ steht für das Eingabeprompt im Terminal):

- falls Sie noch kein Repository gecloned haben:

  ```
  $ git clone https://gitlab.rrz.uni-hamburg.de/Bae4410/pfn1_2019.git
  ```

- Wechseln in das Repository: `$ cd pfn1_2019`
- Aktualisieren: `$ git pull`
- Erzeugen eines eigenen Verzeichnisses für die eigenen Lösungen und Materialien aus der Vorlesung, falls es noch nicht existiert. Damit Sie es später nicht vergessen, verwenden Sie gleich den Verzeichnisnamen, der die Nachnamen der Gruppenmitglieder beeinhaltet. Umlaute ü werden durch ue ersetzt und ß durch ss: Falls ein Nachname aus mehreren Worten besteht, werden diese Worte durch einen Unterstrich verbunden.

  ```
  $ cd
  $ mkdir -p my_pfn1_2019/Uebungen/Blatt01.Name1.Name2
  ```

- Wechseln in das neue Verzeichnis:

  ```
  $ cd my_pfn1_2019/Uebungen/Blatt01.Name1.Name2
  ```

- Kopieren der Materialien für Blatt01:

  ```
  $ cp -r ../../../pfn1_2019/Uebungen/Blatt01/* .
  ```

- Falls durch das Kopieren der Materialien nicht schon Unterverzeichnisse für die einzelnen Aufgabe entstanden sind, diese erzeugen: `$ mkdir Aufgabe1`
- Wechseln in das neue Verzeichnis: `$ cd Aufgabe1`
- Erstellen der Lösung in Dateien, entsprechend der Benennungen aus der Übungsaufgabe (falls Vorgaben gibt).
- Falls Sie noch keinen Texteditor beherrschen, müssen Sie sich mit einem vertraut machen, z.B. mit `vim`. Hierzu gibt es ein interaktives Tutorial: `https://www.openvim.com`

Aufgabe 1.1 (5 Punkte) Ein in der Bioinformatik häufig verwendetes Dateiformat ist TSV (Tab-separated-values). Es handelt sich um tabellarische Daten, in denen einzelne Werte mit Tabulatoren voneinander getrennt werden. Die Werte dürfen dabei das Trennzeichen nicht enthalten. Einzelne Werte dürfen auch fehlen. In diesem Fall kommen mehrere Tabulatoren direkt nacheinander vor oder ein Tabulator kommt direkt vor dem Zeilenumbruch vor.

In den Materialien zur Übung finden Sie die Datei `islands.tsv`. Diese Tabelle enthält Informationen über vorhergesagte *genomic islands* im Genom eines Bakteriums (*Thiomicrospira* sp. MA2-6). Genomic islands sind Teile des Genoms, die durch lateralen Gentransfer von anderen Bakterien akquiriert wurden. Diese Eigenschaft ist aber hier ohne Relevanz.

Die erste Zeile der Datei ist eine Kopfzeile: Diese enthält die Überschriften der Spalten. Die darauffolgenden Zeilen beschreiben Proteine, die in den Island-Bereichen des Genoms vorkommen.

In jeder Zeile (nach der Kopfzeile) enthalten die erste und zweite Spalte die Start- und Endkoordinaten des jeweiligen Island-Bereiches. Jeder Bereich schließt ggf. mehrere Proteine ein, die dann in verschiedenen Zeilen beschrieben werden (und die gleichen Werte in den ersten zwei Spalten enthalten). Die vorletzte Spalte der Tabelle (*Product*) enthält eine Beschreibung der Funktion des Proteins, wenn diese bekannt ist, ansonsten den Wert *hypothetical protein*.

Erzeugen Sie mit Hilfe geeigneter Linux-Kommandos TSV-Dateien mit dem angegebenen Namen, die keine Kopfzeile mehr und genau die folgenden Informationen enthalten:

1. `nohead.tsv`: wie `islands.tsv`, aber ohne Kopfzeile.
2. `hypo.tsv`: Proteine, die als *hypothetical protein* annotiert sind.
3. `nothypo.tsv`: Proteine, die nicht als *hypothetical protein* annotiert sind.
4. `locus.tsv`: Liste der Locus-Bezeichner aller Proteine (Inhalt der Spalte *Locus*).
5. `ranges.tsv`: Start- und Endkoordinaten der Islands-Bereiche, ohne Wiederholungen und numerisch aufsteigend nach der Startkoordinate sortiert.

Für die Lösung können die folgenden Linux-Kommandos und Funktionalitäten der Shell verwendet werden:

- Pipes (`|`), um die Standardausgabe eines Befehls als Standardeingabe eines anderen Befehls zu verwenden
- das Zeichen >, um die Standardausgabe eines Befehls in eine Datei umzuleiten
- `cat` (mit welcher Option können Sie die Tabulatoren sichtbar machen?)
- `head` und `tail` (Wie extrahiert man aus einer Textdatei alle Zeilen ausser der ersten Zeile?)
- `sort` (Wie sortiert man die Eingabe in numerischer statt lexikographischer Reihenfolge? Wie zählt man die Häufigkeit jedes Wertes in einer Liste von Werten?).
- `cut` (Wie definiert man das Trennzeichen? Wie extrahiert man Spalten?)
- `grep` (Wie sucht man Zeilen, die ein String enthalten und Zeilen, die ein String nicht enthalten?)

Hinweise:

- die notwendige Information über die Optionen der Linux-Kommandos erhalten Sie durch die Manualseiten, die man z.B. durch Aufruf von `man cat` erhält.
- `cut`: die Option, um das Trennzeichen zu definieren, verlangt ein einziges Zeichen als Argument. Damit ist `\t` keine gültige Eingabe. Stattdessen verwenden Sie die Shellvariable `$"\t"` (welche einen Tabulator enthält).

Schreiben Sie die Befehlsfolge, um die o.g. Dateien zu erzeugen, in ein Skript mit dem Namen `tsv.sh`. Dieses soll mit dem magischen String `#!/bin/sh` anfangen. Vergessen Sie nicht, die Datei `tsv.sh` durch `chmod u+x tsv.sh` ausführbar zu machen.

Überprüfen Sie anschließend Ihre Lösung anhand des Befehls `make test`. Der gewünschte Inhalt jeder der o.g. Tabellen ist im Verzeichnis der Aufgabe in den Dateien mit der Endung `.solution` zu finden. Sie dürfen die `.solution`-Dateien nicht verändern, da diese für den Test notwendig sind.

Aufgabe 1.2 (5 Punkte) In den Materialien zur Übung finden Sie die Datei `alignments.sam`. Diese Datei liegt in SAM-Format vor. Das ist ein TSV-Format für die Repräsentation von Alignments. Dabei enthält jede Zeile Informationen über das Alignment eines kurzen Sequenz-Abschnitts (Sequencing Read) mit einem Referenzgenom (oder, in einigen Fällen, die Information, dass ein Read nicht aligniert werden konnte). Die Begriffe Sequencing-Read und Alignment sind für das Verständnis und die Lösung dieser Aufgabe nicht relevant.

Zeilen, die mit einem `@`-Symbol anfangen, sind Kopfzeilen.

Die erste Spalte enthält den Namen eines Reads.

Das Referenzgenom besteht häufig aus verschiedenen Referenzsequenzen (z.B. Chromosomen). In der dritten Spalte des SAM-Formates wird der Name der Referenzsequenz, mit der das Read aligniert wurde, angegeben. Der spezielle Wert `*` bedeutet dabei, dass ein Read nicht aligniert ist.

Erzeugen Sie mit Hilfe geeigneter Linux Befehle TSV-Dateien mit dem angegebenen Namen, die genau die folgenden Informationen enthalten:

1. `references.txt`: lexikographisch sortierte Liste der Namen der Referenzsequenzen. Diese Information ist zwar auch im SAM-Header enthalten, soll in der Lösung dieser Teilaufgabe aber aus der dritten Spalte der SAM-Datei extrahiert werden.
2. `aligned_only.sam`: Zeilen, die keine Header-Zeilen sind und den Namen einer der Referenzsequenzen enthalten.
3. `references_count.tsv`: Tabelle mit der Anzahl der Alignments für jede Referenzsequenz sowie die Anzahl der Reads, die nicht aligniert wurden.
4. `reads_n_alignments.tsv`: Anzahl der Alignments für jeden Read, der mindestens einmal aligniert ist (d.h. der in Zeilen vorkommt, die den Namen einer Referenzsequenz enthalten).
5. `reads_n_alignments_distri.tsv`: Anzahl der Reads, die einmal, zweimal, usw. aligniert sind (aus reads_n_alignments.tsv berechnet). Die Ausgabedatei enthält also eine Häufigkeitsverteilung.

Für die Lösung können die folgenden Linux-Kommandos und Funktionalitäten der Shell verwendet werden:

- `grep`: Wie sucht man Zeilen, die einen der Strings enthalten, die in einer Datei aufgelistet sind?
- `uniq`: Wie zählt man in einer sortierten Eingabe, die Anzahl von gleichen Elementen?
- `tr`: Wie wandelt man Leerzeichen in Tabs um? Wie eliminiert man Wiederholungen des gleichen Zeichens?
- `cut`: Wie gibt man alle Spalten einer Datei, außer der ersten, aus?

Tipps:

- die notwendige Information über die Optionen der Linux-Kommandos erhalten Sie durch die Manualseiten, die man z.B. durch Aufruf von `man tr` erhält.
- `uniq` (mit einem geeigneten Parameter) liefert eine Liste, in der steht, wie häufig aufeinanderfolgende identische Elemente in der Eingabe vorkommen. Diese Zahl wird leider nicht als Tabulator getrennt ausgegeben. Um die Ausgabe in einem TSV umzuwandeln, müssen Sie folgende drei Schritte in ihrer Pipe implementieren: (1) `tr`, um die wiederholten Leerzeichen

am Anfang der Zeile zu entfernen; (2) wieder `tr`, um die Leerzeichen in Tabs umzuwandeln; (3) `cut`, um die erste leere Spalte aus der Ausgabe zu entfernen.

Schreiben Sie die Befehlsfolge, um die o.g. Dateien zu erzeugen, in ein Skript namens `sam.sh`. Dieses soll mit der Zeile `#!/bin/sh` beginnen. Vergessen Sie nicht, die Datei `sam.sh` durch `chmod u+x sam.sh` ausführbar zu machen.

Überprüfen Sie anschließend Ihre Lösung durch Ausführen von `make test`. Der gewünschte Inhalt jeder der o.g. Tabellen ist im Verzeichnis der Aufgabe in den Dateien mit der Endung `.solution` zu finden. Sie dürfen die `.solution`-Dateien nicht verändern, da diese für den Test notwendig sind.

Bitte die Lösungen zu diesen Aufgaben bis zum 28.10.2019 um 18:00 Uhr an pfn1@zbh.uni-hamburg.de schicken. Die Besprechung der Lösungen erfolgt am 30.10.2020.