生物情報科学演習

森下研M1 鈴木 裕太

sed, awk

- 生物学で公開されるデータの形式の多くがascii-text
  - fasta, fastq, sam, bed, wig, その他独自の形式,...
- プログラムの入力や出力を整形する作業が生じる
  - 「パールのようなもの」に頼る
    - Perl, Python, Ruby, ...
  - （小さな）glue 言語としての sed, awk, bash
- 仮説を立てる/結果を吟味するためにデータを眺める
  - awk, gnuplot, R

sed とは

- sed (**s**tream **ed**itor)
- ストリーム指向・非対話型のテキストエディタ
- ストリーム指向 ＝ ファイルの先頭から順に処理する
- 非対話型 ＝ 対話的なエディタ(e.g. メモ帳, emacs, vim)とは異なり、編集は一括に起こる
- 大量のファイルを一括で処理したい時に活躍

sed の実行

- sed では、編集の内容はコマンドにより指示され、入力ファイルに適用される
- 入力ファイルは変更を受けず、結果はstdoutへ出力される
- `$ sed -e 'script' input_file`
- `$ cat input_file | sed -e 'script'`
- `$ sed -f script_file input_file`
- `$ cat input_file | sed -f script_file`

sed の実行サイクル

- sed は、次のようなループを実行する
  - 1. 入力ファイルから一行読み込む
  - 2. これに（必要ならば）コマンドを適用する
  - 3. 結果を出力する → 1. へ戻る
- sed は、各行ごとに独立に同様の処理をするタスクに向く
- Turing 完全？
  - 大きい処理には、素直に大きい道具を

sed の構文

- <address><command>
  - 1d (先頭1行を削除)
- <address>,<address><command>
  - 1,100d (1~100行を削除)
- <address> := Φ(全てマッチ) | 行番号 | /正規表現/
  | $(最終行) | ...
- <command> := d | p | s | q | ...
  - /^>/s/_.*$//g
    ('>'で始まる行について、'_'から行末までを削除)

sed の例

- `cat sample.fa | sed -e '/^>/!y/cC/tT/ > sample_CtoT.fa`
  - !はアドレスの否定，yは文字単位での置換
  - fasta形式の配列にC/T変換を施す
- `cat -n sample.log | sed -ne '/ERROR/,+10p' | less -S`
  - -nは出力の抑制，+Nは相対座標の指定
  - ログファイルから、ERRORを含む行とそれに続く10行を閲覧する

sed の参考文献

- www.gnu.org/software/sed/manual/sed.html
  - GNU本家のマニュアル
- www.ibm.com/developerworks/jp/linux/library/l-sed1/
  - Daniel Robbins氏による入門記事（和訳;全3回）
- pekopeko1.bitbucket.org/blog/html/2013/03/23/sed.html
  - yet another 入門記事
    - 正規表現の後方参照を、課題で利用します

おまけ

```
#!/bin/bash
bred=`printf "\033[31m"` # a
bgre=`printf "\033[32m"` # c
byel=`printf "\033[33m"` # g
bblu=`printf "\033[36m"` # t
ecol=`printf "\033[39m"`

while read line < $1
do
  echo "$line" |
  sed -e "s/\([Aa]\)/$bred\1$ecol/g" \
      -e "s/\([Cc]\)/$bgre\1$ecol/g" \
      -e "s/\([Gg]\)/$byel\1$ecol/g" \
      -e "s/\([Tt]\)/$bblu\1$ecol/g"
done | less -R
```

課題1

- （状況）input/nohup.out は nohup(1)を利用しPacBio付属のpipelineを実行したときのログの一部である。このpipelineの実行では、入力が246のchunkに分割されP_Filter, P_Mappingの各モジュールに掛けられた。各モジュールはさらに小さなtaskからなり、ログファイルにはchunkごとのtaskの成功/失敗が書き出された。
- （課題）input/nohup.out から、“P_Filter.filter” taskに成功しなかったchunkの番号を改行区切りで列挙せよ
  - hint: “filter”, “sucessfully completed”を含む行を探す
  - sed で整形し、taskが成功したchunkの番号を列挙する
  - もう一声

gnuplot とは

- テキストで表現されたデータを簡単にプロットするソフトウェア
- 対話的にも、非対話的にも呼び出せる
- デフォルトでXに出力されるが jpg, png, svgなども生成でき、レポートや論文に仕様するグラフも作れなくもない
- プロット専用なので、データの統計解析は基本的に（平均や分散の算出すら）出来ない。必要な数値は事前に求める必要がある（Rとの大きな違い）

gnuplot の実行

- `$ gnuplot`
  - 対話環境へ入り、コマンドを順次実行
  - データを軽く確認したいとき
- `$ gnuplot` *`plot_file`*
  - コマンドを記述したファイルを渡し実行
  - プロットの数が多いとき
  - 見た目を微調整して綺麗な出力にしたいとき

gnuplot の例

```
unset key
set xrange [0:1]
set yrange [0:1]
set grid
set size square
set term svg

set output "ROC_whole.svg"
set key outside
set parametric
plot "opt2pwus.dat" u (1-($3)):2 w l title "aggregate",\
     "point2.dat" u (1-($3)):2 w l title "pointwise" lc rgb "green",\
     (1-0.9833), 0.9020 w p title ""

set output "SP_whole.svg"
plot "opt2pwus.dat" u 2:4 w l title "aggregate",\
     "point2.dat" u 2:4 w l title "pointwise" lc rgb "green",\
     0.9020, 0.8679 w p title ""

#set output "ROC_whole_len.svg"
#plot "opt2_whole_pos_l10.dat" u (1-($3)):2 w l title "10",\
#     "opt2_whole_pos_l30.dat" u (1-($3)):2 w l title "30",\
#     "opt2_whole_pos_l50.dat" u (1-($3)):2 w l title "50",\
#     "opt2_whole_pos_l70.dat" u (1-($3)):2 w l title "70"

#set output "SP_whole_len.svg"
#plot "opt2_whole_pos_l10.dat" u 2:4 w l title "10",\
#     "opt2_whole_pos_l30.dat" u 2:4 w l title "30",\
#     "opt2_whole_pos_l50.dat" u 2:4 w l title "50",\
#     "opt2_whole_pos_l70.dat" u 2:4 w l title "70"

set output "gamma-whole.svg"
set xrange [-3:-1]
set yrange [*:*]
plot "opt2pwus.dat" u 1:2 w l title "sensitivity",\
     "opt2pwus.dat" u 1:3 w l title "specificity",\
     "opt2pwus.dat" u 1:4 w l title "precision",\
     "opt2pwus.dat" u 1:5 w l title "F1 score"
~
~
```

2

1

演習

- input/FilledBases.dat
  - metrics of filled gaps in medaka reference ver.1 scaffold
  - gap_id, #A, #C, #G, #T, #bases, GC rate
- GC rateが偏るとシーケンシングが難しい → gap領域のGC rate は偏りがある？ → 予想を確かめるため描画
- plot “input/FilledBases.dat” u 6:7
  - 長さ 1 kbp 以下の領域を見てみよう
  - 長さ 200 bp 以下の領域の幾何学模様は何か？
- 横軸にGC rateを取り、gap数の累積分布を描け
  - sort(1), cat(1)を利用

gnuplot の参考文献

- www.gnuplot.info/documentation.html
  - Official documents
  - set ~ で指定できるオプションを探すと解決することが多い
- http://lowrank.net/gnuplot/index.html
  - gnuplot not so FAQ

awk とは

- CSV(カンマ区切り) に代表される、構造を持つデータファイルを処理するスクリプト言語
- sed と同じく、非対話型の処理をする
  - awk ～ sed + C-like operation
- sed で行ごとの整形をするとしたら、awk(+gnuplot)はExcelやCalcの様な表計算ソフトの代替ができる
- GNUで拡張された GNU awk (gawk)が有名
  - 以下、awk と書けば gawk のこととする

awk の実行

- awk では、編集の内容はコマンドにより指示され、入力ファイルに適用される
- 入力ファイルは変更を受けず、結果はstdoutへ出力される
- `$ gawk 'script' input_file`
- `$ cat input_file | gawk 'script'`
- `$ gawk -f script_file input_file`
- `$ cat input_file | gawk -f script_file`

Q: sed との相違はどこか？

awk の実行サイクル

- sed が行ベースで処理するのに対し、awk の処理は「レコード」と「フィールド」に基く
- レコードはレコードセパレータ(RS; デフォルトでは改行)で区切られた単位
- フィールドはフィールドセパレータ(FS; デフォルトでは一つ以上の空白)で(ry
- awk は入力ファイルの先頭からレコードを一つ読み、それをフィールドに分割する
- $1, $2, $3, ... に各フィールドが代入され、パターンが真ならばコマンドが実行される
- 構文は
  - *pattern1* {*command1*; *command2*; ... }
  - *pattern2* {*command1*; *command2*; ... }
- pattern は、sed 同様のΦ・index・正規表現の他、色々な条件文が指定できる

awk の例

- `{ print $0 }`
  - 入力をそのまま出力（`$0` はレコード全体を指す）
- `BEGIN{ FS=”,” } { print $5 }`
  - CSVの、各レコードの第5フィールドを出力
- `$6>0.5{ print $1, $3, $6}`
  - 第6フィールドが0.5より大きいレコードの、第1,3,6フィールドを出力
- `$1 ~ /`*`pattern`*`/{ sum += $2 } END{ print sum }`
  - 第1フィールドが/*pattern*/にマッチしたレコードの、第2フィールドの総和を出力

演習

- input/FilledBases.dat
  - カンマ区切り、タブ区切りに変換する
    - sed ならどうなるか？
  - 最終列を、1,2列目の間へ移動する
    - sed にはできない……？
  - レコードにpurine rate, pyrimidine rateを追加せよ
    - プロットしてみよう？

awk の参考文献

- www.gnu.org/software/gawk/manual/gawk.html
  - 最新のマニュアル
- http://gauc.no-ip.org/translation/gawk.html
  - 少し古いマニュアルの和訳（これでも充分？）
- www.ibm.com/developerworks/jp/linux/library/l-awk1/
  - Daniel Robbins氏による入門記事（和訳;全3回）

課題2

- 入力
  - input/medakaAssembly_version1_0_fasta.fasta.fai
  - input/asm_CA_01_20131125_175458.ctg.fasta.fai
- 二つの配列それぞれの平均長，N50値，N90値を計算するawkスクリプトを書け
  - または、awkを呼び出して計算するshellスクリプト
  - (注意) awk だけで完結させる必要は無い
- 全てのNxx値(0<xx<100)が同時に分かるプロットを描け
  - つまり、「配列長」の累積分布を描けばよい
  - プロットファイルか、生成した画像又はスクリーンショットを提出

bash の参考文献

- パイプ・リダイレクトなどを理解したら、if, for, whileや変数を利用して何か書いてみる
- string substitution, here documentなども便利
- www.ibm.com/developerworks/jp/linux/library/l-bash.html
  - Daniel Robbins氏による入門記事（和訳;全3回）
- http://tldp.org/LDP/abs/html/
  - 教科書

sed, awk に頼るとき

- パイプの中に混ぜることができ、文字列処理に長けた他多数のUNIXツールとの相性が良い
  - デフォルトでSTDIN, STDOUTを読み書きするため
- ファイルを読み一行づつ読む、または、レコードに分割しフィールドまでパースする作業が組み込んである
  - 他のLL言語では少なくとも1行は定型の処理が必要
- 得意なら perl で済ますことも出来るが・・・・・
- 使捨てのスクリプトやone-linerが多い
  - 素早く書けないと感じたら別のより高級な言語を

課題の提出

- 課題に必要なファイルは演習のページに置きます
  - http://mlab.cb.k.u-tokyo.ac.jp/courseware
- 課題1,2について、書いたスクリプトを提出
  - 結果を再現できる程度に
- 締切: JST 12/31 23:59
  - (真の締切): ???
- moris@cb.k.u-tokyo.ac.jp
- ysuzuki@cb.k.u-tokyo.ac.jp
  - 質問は随時受付けます