演習問題　解答編

復習問題１　UNIX 基本コマンド

1.
アプリケーション ＞ ユーティリティ ＞ ターミナル
```
$ cd data/1_unix
$ pwd
$ ls -R
```

2. 以下は、先に~/unixtest ディレクトリに移動してから操作する場合となります。
```
$ cd ~/unixtest
$ mkdir FASTA-EX
$ cp ~/data/1_unix/sprot/*.fasta FASTA-EX
$ ls ~/unixtest/FASTA-EX
```

3. 以下は、~/unixtest がカレントディレクトリの場合です。
```
$ cd FASTA-EX
$ man grep
$ grep -h '^>' * | less
```

4. tail コマンドの引数に "-n +行数" を指定すると、先頭から数えて、指定された行数以降を表示します。
```
$ tail -n +2 -q * | less
```

5.
```
$ cat *HUMAN.fasta > ../human.fasta
$ cat *MOUSE.fasta > ../mouse.fasta
$ cat *DROME.fasta > ../drome.fasta
```

6. 以下は、~/unixtest/FASTA-EX がカレントディレクトリの場合です。
```
$ cd ..
$ tar zcvf FASTA-EX.tar.gz ./FASTA-EX
$ tar ztvf FASTA-EX.tar.gz
$ rm -rf FASTA-EX
```

上記のようにした場合は、解凍した際に FASTA-EX ディレクトリが作成されてその下にファイル群が配置される。一方

```
$ tar zcvf ../FASTA-EX.tar.gz .
```

のように対象ディレクトリにカレントディレクトリ(.)を指定すると、解凍時にカレントディレクトリに直に大量のファイルが作成されてしまうので注意すること。

7. chmod u+rw,g-rwx,o-rwx FASTA-EX.tar.gz でもよい。
```
$ chmod 600 FASTA-EX.tar.gz
```

*8. find コマンドは検索の起点になるディレクトリパスを最初に指定し、評価式 -name の後ろにファイル名を指定することによって評価式に従ってディレクトリツリーを検索します。この時指定するファイル名にはワイルドカード等を使用したパターンマッチによる検索もできます（9.で使用)。

```
$ find ~ -name SYYM_DROME.sprot
```

*9. -exec アクションは、それ以降に記述されたコマンドから";"までをコマンドラインとして実行でき、検索結果のファイルをコマンドの引数として使用することができます。'{}'は検索結果の中で現在の処理対象となるファイルに置き換えられます。ある条件に合致するディレクトリツリー下にあるファイル全てに対して特定の処理を行いたい時に便利です。

```
$ mkdir ~/SP
$ find ~ -name '*.sprot' -exec cp {} ~/SP ¥;
```

復習問題２　R 入門

1. (a) 男性で喫煙歴がある人のデータを抽出し、その行数を数える。A %in% B （A が B の要素のうちどれかと一致する）を使う。

```
> cancer.subset <- subset(cancer, gender=="male" & smoking %in% c("former", "current"))
> nrow(cancer.subset)　（→62 個）
```

(b) cancer.subset から gene1 の発現データを抜き出してその平均値をとる。

```
> mean(cancer.subset$gene1)　（→11.44719）
```

(c) タブ区切りテキストファイルとして保存。

```
> write.table(cancer.subset, sep="¥t", file="cancer.subset.txt")
```

2. (a) gene1 と gene2 の散布図を作成し、gender によって点を色分けする。

解 1)　x 軸、y 軸のデータを指定する一般的な書き方。xlab, ylab で、各軸のラベルを見やすく書き直している。

```
> plot(cancer$gene1, cancer$gene2, col=cancer$gender, xlab="gene1", ylab="gene2")
```

解 2）モデル式を使った書き方。

```
> plot(gene2 ~ gene1, cancer, col=gender)
```

(参考 1)　解 1 は attach を使うと以下のように簡潔に書ける。attach は変数名をデータフレームからとる（変数名のサーチパスに加える）ことを指示する。

```
> attach(cancer)
> plot(gene1, gene2, col=gender)
```

(参考 2)プロットの色を変更したい場合は以下のような書き方がある。

```
> color <- c("red", "blue")　　# 色を 1 が赤、2 が青に設定する
```

```
> plot(gene2 ~ gene1, cancer, col=color[gender])
```

（この動作を理解するために、color[cancer$gender]を実行して見よ）

(b) 散布図に回帰直線を引く。まず lm で線形モデルへのあてはめを行い、結果を変数(ex.lm)に格納する。それを用いて abline で回帰直線を引く。

```
> ex.lm <- lm(gene2 ~ gene1, cancer)
> abline(ex.lm)
```

回帰直線の有意性については、summary で確認する。

```
> summary(ex.lm)
```

出力結果の最後の行から p-value は 0.3965 と有意水準より大きいので、有意ではない。

3. gender によってデータを分割すると、stage による違いが観察されなくなった。これは、もともと観察された stage III での gene1 の発現量の減少は、stage III の患者が gene1 の発現が低い女性に偏っているためであることを示唆している。このことは

```
> subset(cancer, stage=="stage III")
```

によって確認できる。

4. (a,b) cancer から 5-10 カラムを抜き出して変数 expr に代入し、cor を実行する。

```
> expr <- cancer[,5:10]
> cor(expr)
```

(c) 相関係数の絶対値が 0.5 以上であるかどうかは、以下のコマンドで調べられる。

```
> abs(cor(expr)) >= 0.5
      gene1 gene2 gene3 gene4 gene5 gene6
gene1  TRUE FALSE FALSE FALSE FALSE  TRUE
gene2 FALSE  TRUE  TRUE  TRUE  TRUE FALSE
gene3 FALSE  TRUE  TRUE  TRUE  TRUE FALSE
gene4 FALSE  TRUE  TRUE  TRUE  TRUE FALSE
gene5 FALSE  TRUE  TRUE  TRUE  TRUE FALSE
gene6  TRUE FALSE FALSE FALSE FALSE  TRUE
```

この結果から、相関の有無によって(gene1,gene6) と (gene2,gene3,gene4,gene5)の 2 つのグループに分けられることがわかる。

多変数間の関係を解析する多変量解析には、クラスタリングや主成分分析など様々な手法があるが、相関係数を計算することは、多くの手法においてその基盤となっている。

復習問題３　UNIX によるテキストファイル処理

1. $ awk '$2>=100{print}' ecoli.htseq
2. $ awk '$2>=100{print}' ecoli.htseq | grep '^b' > ecoli.temp
3. $ sort –k 2,2nr ecoli.temp > ecoli.htseq.sorted
4.

```
awk '$2>=100{print}' ecoli.htseq | grep '^b' > ecoli.temp
sort –k 2,2nr ecoli.temp > ecoli.htseq.sorted
rm ecoli.temp
```

最終行に rm ecoli.temp を加える事により、中間ファイルを削除できる。

別解

```
awk '$2>=100{print}' ecoli.htseq | grep '^b' | sort –k 2,2nr >
  ecoli.htseq.sorted
```

5.

```
filename=$1
awk '$2>=100{print}' $filename | grep '^b' > ecoli.temp
sort –k 2,2nr ecoli.temp
rm ecoli.temp
```

$ ./htsort.sh ecoli_other.htseq を実行して確認すること。

復習問題４　シェルスクリプト

1.

```
#!/bin/sh
org=${1}
for i in 4 5 6
 do
   bowtie2 –x ../${org}_genome –U ${org}.${i}.fastq –S ${org}.${i}.sam
 done
```

2.

・標準出力に出力されるもの

```
ecoli.1.fastq was not found
ecoli.2.fastq was not found
ecoli.3.fastq was not found
ecoli.4.fastq has already been mapped
ecoli.5.fastq has already been mapped
ecoli.6.fastq has already been mapped
```

・このスクリプトの目的

① 既に実行し終えたマッピングを再度実行しない

② 存在しない fastq ファイルを指定した時はメッセージを出力してマッピングを実行しない

2.解説：

このスクリプトは、

1. 既に sam ファイルが存在した場合 `(if)` → ... has already been mapped と出力して終了
2. sam ファイルは存在しないが、fastq ファイルが存在した場合 `(elif)` → bowtie2 によるマッピングを実行して終了
3. どちらでもない場合 `(else)` → ... was not found と出力して終了

と 3 つの条件によって動作を変えます。

ecoli.1.fastq～ecoli.3.fastq ファイルは、~/data/5_script 内にあるため、条件 3 となり「... was not found」と出力されます。ecoli.4.fastq～ecoli.6.fastq ファイルは、問題 1 でマッピングが終了しているため「... has already been mapped」と出力されます。

このように if 文を使い、既にファイルが存在するかどうかで、スクリプトの働きを変更させることができます。

実践演習 1　RNA-Seq 解析結果の集計

1) ファイルが 1,2,3,...の順に表示されるのは B)の方。

```
$ paste results/ecoli.{?,1?}.htseq > ecoli.paste_all
```

2) 遺伝子にヒットしなかったリードの情報の行は、_で始まっているのでこれを除く。

grep –v はマッチする行を除いて出力する。

```
$ cut –f 1,2,4,6,8,10,12,14,16,18,20,22,24 ecoli.paste_all | grep -v
'^_' > ecoli.count_all
```

3) 長さは （終了位置）—（開始位置）＋ 1 で計算される

```
$ awk '$3=="exon" {print $10, $5-$4+1}' ecoli.gtf | sed 's/[";]//g' >
ecoli.gene_length
```

4) 行の先頭からアルファベット順に並べたい場合は sort のオプションは不要。また、join のキーがどちらも 1 列目であるときは join のオプションは不要。最後の sed コマンド中の*<TAB>* は、Control-v を押してから Tab キーを押す。

```
$ sort ecoli.gene_length > ecoli.gene_length_sorted
$ join ecoli.gene_length_sorted  ecoli.count_all | sed 's/ /<TAB>/g' >
ecoli.result_all
```

実践演習 2　R を用いた RPKM 値の計算

1） セパレータはタブ(sep="¥t")、ヘッダはなし(header=F)、1 列目を行の名前として読み込む（row.names=1)。

```
> eco_rna <- read.table("ecoli.result_all", sep="¥t", header=F,
row.names=1)
```

2） eco_rna から 2-13 列目を抽出した行列を、1 列目を抽出したベクトルで割る。

```
> eco_rna_tmp <- eco_rna[,2:13] / eco_rna[,1] * 1000
```

3） eco_rna の 2-13 列目を列方向にベクトルとして取り出して sum 関数を適用する。

```
> eco_rna_readsum <- apply(eco_rna[,2:13], 2, sum)
```

4） eco_rna_tmp を行方向のベクトルとみて、同じ要素数のベクトル eco_rna_readsum で割る。それを 1,000,000 倍する。その後、転置行列をとる。

```
> div <- function(x,y){x/y}
> eco_rna_rpkm0 <- apply(eco_rna_tmp, 1, div, eco_rna_readsum) *
1000000
> eco_rna_rpkm <- t(eco_rna_rpkm0)
```

（別解）行方向にベクトル演算をしたい場合、まず転置行列をとって列方向のベクトルとして演算した後（これは簡潔に書ける)、再度転置行列をとる方法もある（結果を比べてみよ)。

```
> eco_rna_rpkm2 <- t(t(eco_rna_tmp) / eco_rna_readsum) * 1000000
```