

GITC 2019 夏 準備編 UNIX・R・NGS の基礎 演習

復習問題1 UNIX 基本コマンド (*印は応用問題です。時間に余裕があればトライしてみてください)

1. ターミナルを起動して以下のコマンドを実行せよ。
 - 1-1. `~/data/1_unix/exercise` ディレクトリに移動せよ。
 - 1-2. カレントディレクトリ (現在のディレクトリ) の名前を確認せよ。
 - 1-3. カレントディレクトリの内容を表示せよ。(使用するコマンド: `cd`, `pwd`, `ls`)
2. `~/data/1_unix/exercise` ディレクトリ内には **Human**, **Mouse** の遺伝子配列 **FASTA** ファイル (拡張子が `.fasta` のファイル) がある。これらを、一旦全てコピーしてから、生物種ごとのディレクトリに分けて保存したい。
 - 2-1. 新しく `~/data/1_unix/exercise` 内に **test** ディレクトリを作成せよ
 - 2-2. ワイルドカード「*」 を使って全ての **FASTA** ファイルを **test** の中へ、コピーせよ
 - 2-3. 正しくコピーされたかを確認するために、**test** ディレクトリの内容を表示せよ。
 - 2-4. **test** ディレクトリに移動せよ。
 - 2-5. 下記の名前で新しいディレクトリを2つ作成せよ。
Human
Mouse
 - 2-6. ファイル名に「**HUMAN.fasta**」を含むファイルは **Human** ディレクトリへ、
「**MOUSE.fasta**」を含むファイルは **Mouse** ディレクトリへ、
ワイルドカード「*」 を使って移動させよ。
 - 2-7. 正しく移動されたかを確認するために、**Human**, **Mouse** ディレクトリの内容を表示せよ。
(使用するコマンド: `cd`, `cp`, `mkdir`, `mv`, `ls`)
3. 移動させた **FASTA** ファイルは、1 ファイルに 1 つの配列しか入っていない。これらを連結してマルチ **FASTA** ファイルを作成する。
 - 3-1. **Mouse** ディレクトリに移動せよ
 - 3-2. cat コマンド および ワイルドカード を使用して、全ての **FASTA** ファイルの内容を画面に出力してみよ。
 - 3-3. 上記の出力を リダイレクト「>」 を使って、**mouse.fasta** というファイルに書き出せ。
(使用するコマンド: `cd`, `cat`)
4. 作成した **mouse.fasta** ファイルには、いくつかの遺伝子配列が含まれているを知りたい。
 - 4-1. **mouse.fasta** から、遺伝子配列名を表す行 (「>」で始まる行) を検索せよ。
 - 4-2. 上記の出力を リダイレクト「>」 を使って、**mouse_list** というファイルに書き出せ。
 - 4-3. **mouse_list** の行数を調べよ。
 - 4-4. **mouse_list** というファイルを作らずに、遺伝子配列名を表す行の数を数えよ (パイプ「|」を使うこと)
(使用するコマンド: `grep`, `wc`)
- *5. `~/data/1_unix/exercise/test/Human` ディレクトリのバックアップを作成してから削除する。
 - 5-1. `~/data/1_unix/exercise/test` ディレクトリに移動せよ
 - 5-2. **Human** ディレクトリの圧縮アーカイブを **human.tar.gz** として tar コマンド で作成せよ
(tar のオプションは "`zcvf`" を使用)。
 - 5-3. 作成した、圧縮アーカイブの中身を確認せよ (tar のオプションは "`ztvf`" を使用)。
 - 5-4. 確認できたら **Human** ディレクトリを削除せよ。
- *6. UNIX にはファイル検索に用いる find という非常に便利なコマンドがある。man コマンドを使用して find コマンドの使用法 を調べよ。この find コマンドを使用してホームディレクトリのどこか (ホームディレクトリ直下とは限らない) にある "**2D2L_rep1_R1.fastq**" というファイルを探せ。

復習問題 2 エディタとスクリプト

この演習では、以下のフォルダのファイルを使用する。

~/data/2_editor/

- 1) テキストに記載している(演習)引数、および(演習)引数の使用のスライドを読むこと。
- 2) `example4.sh` を新規作成せよ。
ファイルの編集には Emacs を使用すること。
- 3) `$./example4.sh exon`
として、`example4.sh` を実行し、実行結果を確認せよ。
実行できない場合は実行権が付与されているか確認せよ。
- 4) 3)を実行した後、`.tmp` という中間ファイルが生成される。
中間ファイルは必要ないため、生成された`.tmp` ファイルを消したい。
そうなるように `example4.sh` を編集せよ。
編集後に `example4.sh` を実行し、`ls` コマンドで`.tmp` ファイルが存在しないことを確認せよ。

復習問題 3 R

(基本演算編)

R で標準で使える **women** データを使って、以下の問題を考えよう。

1. まずコンソール上で **women** とタイプしてデータを表示せよ。**women** データは身長(height)が inch、体重(weight)が pound であらわされている。これを、身長を cm、体重を kg の単位に直したい。以下の手順でこれを行え。

(a) **women** から身長の列を抜き出し、これを cm に変換して、**h** という変数に代入せよ。ただし、1 inch=2.54cm である。

(b) **women** から体重の列を抜き出し、これを kg に変換して、**w** という変数に代入せよ。ただし、1 pound=0.454kg である。

(c) `data.frame(height=h, weight=w)` によって新しいデータフレームを作り、**women2** という変数に代入せよ。`

2. 前問で作成した **women2** の各行について、その身長と体重からボディマス指数(BMI)を計算せよ。ただし、体重 w kg、身長 h m (cm ではない) の人の BMI は w/h^2 で定義される。

(統計解析編)

R 入門の講義で用いた **cancer** データを使って、以下の問題を考えよう。変数 **cancer** が残っていない場合は、作業ディレクトリを `~/data/3_R` に変更してから `read.table` を使って読み込むこと (テキスト「データフレームの読み込み 1」)。

3. (a) 男性で喫煙歴がある人のデータを抜き出し、結果を **cancer.subset** という変数に代入せよ。何人いるか。

(b) それらの人の **gene1** の発現データを取り出し、その平均値を計算せよ。

(c) 抽出した結果をタブ区切りテキストとして **cancer.subset.txt** というファイルに保存せよ。

4. (a) **gene1** と **gene2** の散布図を作成し、**gender** によって点を色分けせよ。

(b) 散布図に回帰直線を引け。この回帰直線へのあてはめは、有意水準を 0.01 として有意であると言えるか。

*5. テキストでは、**gene1** の発現量(**gene1**)が性別(**gender**)によって違いがあるという結論が t 検定で得られ、また癌のステージ(**stage**)によっても違いがあるという結論が分散分析から得られた。ただし、癌のステージの中で明らかな違いがあるのは **stage III** のみであった (これは `boxplot (gene1 ~ stage, cancer)` で確認できる)。

そこで、**gender** の効果を考慮しつつ **stage** の比較を行うため、Trellis Plot の技法を使ったプロットを作成してみよう。これを行う `lattice` package は R に標準で含まれているが、使う際に

はライブラリのロードが必要である。

```
> library(lattice)
```

```
> bwplot(gene1 ~ stage | gender, cancer)
```

bwplot は **boxplot** と同様に箱ひげ図を作成するが、特定の因子によって条件付けしたプロットを作成できる。ここでは、**gender** によってまず被験者を **female** と **male** に分けて、そのそれぞれで箱ひげ図を作成している。この結果から **stage** の **gene1** の発現量への効果について、どのような結論が得られるか考察せよ。

*6. (a) **cancer** データから各患者の **gene1** から **gene6** の発現量を抜き出した部分データフレームを作成し、変数 **expr** に代入せよ。

(b) 各遺伝子間の発現量の相関（散布図を描いたときに傾きを持つ直線上に分布する傾向）の強さは相関係数によって表される。相関係数は-1 から 1 までの値を取り、0 が無相関を表す。相関係数が負の値のときは、傾きが負、すなわち一方が大きくなれば他方が小さくなる関係を表す。R では、行列の各カラム間の相関係数は、**cor** 関数によって一度に計算できる。これを用いて **expr** の各カラム間の相関係数を計算せよ。

(c) 相関係数の絶対値が 0.5 以上のときに強い相関があるとして、**gene1**~**gene6** を、発現の相関の強さによっていくつかのグループに分けることができるかを検討せよ。ただし、絶対値をとるのは **abs** 関数で行える。

(関数の作成)

7. 与えられたベクトルに対し、二乗平均平方根(**root mean square**)を計算する関数を **RMS** という名前で作成せよ。ただし、二乗平均平方根は、ベクトルの各要素を二乗した値の平均値の平方根であり、与えられた値（ベクトル）の平方根をとる関数は **sqrt** である。また、関数はエディタを使って作成すること。作成した関数を使って **RMS(1:5)** を計算せよ。

*8. 「関数の作成(2)」のスライドで使用した **plotAll** 関数について考えよう。

(a) プログラムのソースコード (**plotAll.R**) を直接読み込むのではなく、エディタで開いてからマウスでコマンド全体を選択して実行してみよう（「エディタからのコマンドの入力と実行」参照）。これで **plotAll** 関数が定義される。これを用いて **plotAll(cancer[,1:4])** を実行せよ。

(b) **plotAll** 関数は、引数が一つの場合は、そのデータフレーム内での総当たりのプロットを作成するが、対角線上とそれ以外とは異なるコマンドでプロットを作成している。左上のプロット、およびその下の 2 行 1 列目のプロットと同じプロットを直接作成する **plot** コマンドはそれぞれどのようなものか、**plotAll.R** のプログラムから考えてみよ。ただし、タイトル (**main**) やラベル (**xlab**, **ylab**) をつけるところは難しいので無視してよい。

復習問題 4 NGS 基本データフォーマット

~/data/4_format に移動せよ

1. SRR073576 (SRA のアクセッション番号) はエンドウヒゲナガアブラムシのバクテリオームの RNA-seq 解析結果のデータである。
 - 1) SRR073576 を NCBI で検索し、情報を確認せよ。
 - * シングルリードなのか、ペアエンドリードなのか確認しよう。
 - 2) SRA Toolkit の `prefetch` コマンドを使用し、SRA データをダウンロードせよ。
 - * `prefetch` コマンドの使用方法是コマンドのヘルプ機能で調べること。
 - * デフォルトではファイルの出力先は `~/ncbi/pubkic/sra` となる。
 - `--output-directory` オプションを使用することで、出力先ディレクトリを指定可能。
 - 3) SRA Toolkit の `fastq-dump` コマンドを使用し、sra 形式のファイルから fastq ファイルを抽出せよ。
2. bed ファイル(ex4.bed)と gtf ファイル(ex5.gtf)はヒト染色体上にある遺伝子群について同じ情報を表している。それぞれのファイルの形式の違いに注意しつつ、以下の問に答えよ。
 - 1) 何番染色体にコードされているか。
 - 2) いくつの遺伝子(重複領域に別名のものもそれぞれ数える)が含まれているか。
 - 3) 遺伝子 BC041449 にエキソンはいくつ含まれているか。
 - 4) 遺伝子 BC041449 の最初のエキシソンの開始位置と最後のエキシソンの終了位置はそれぞれ何か。ただし、最初の塩基の位置座標は 1 とし、エキソンの開始、終了は転写される向きに沿って考えること
3. bed ファイルはタブ区切りのファイルである。
 - 1) R を使って ex4.bed からデータを読み込み、変数 bed に代入せよ。また変数 bed の内容を確認せよ。
 - 2) ex4.bed にはエキソンを一つから最大六つまでもつ遺伝子が含まれている。変数 bed からエキソン数の情報を取り出し、それぞれのエキソン数をもつ遺伝子がいくつずつあるかカウントせよ。ただし、与えられたベクトルの要素の頻度をカウントする関数は `table` である。
4. Sam ファイル(review_4-4.sam)は paired-end の map 結果である。
 - 1) ここに上がっている paired-end 数はいくつか。
 - 2) そのうち正しい paired-end の方向で map しているものはいくつか。

復習問題 5 クオリティコントロールと NGS 基本ツール

~/data/5_ngs に移動せよ

1. 2D2L_rep1_R1.fastq と 2D2L_rep1_R2.fastq ファイルはアラビドプシスの発芽・緑化後の芽生えをサンプリング、ライブラリー作製した Paired-end read(76base x2)の RNA-Seq の生リードの fastq ファイルである。これを用いて、以下のパラメータを参考にし、paired-end での cutadapt をかけよ。

```
-q 30
-O 7
-m 50
-a AGATCGGAAGAGCACACGTCTGAACTCCAGTCAC
-A AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGTAGATCTCGGTGGTCGCCGTATC
```

- 1) Cutadapt の log を見て、pass した pair 数、quality trim された base 数を調べよ。
- 2) Cutadapt 処理前後の fastq ファイルを less コマンド等で見比べよ
- 3) wc コマンドで cutadapt 前後の read 数を調べよ。
- 4) Cutadapt 処理前後の fastq ファイルを fastqc につけ、cutadapt 処理による、低品質配列が除かれていることを確認せよ。

2. Seqkit を使ってリードファイル ecol1.2.fastq, ecol1.3.fastq の statistic 情報を確認せよ。

3. bowtie2 を使って、リードファイル ecol1.2.fastq, ecol1.3.fastq を、リファレンス: eco にマッピングし、結果をファイル eco_ex.sam に出力せよ。その際、リードファイルはカンマ区切りで複数指定できることを使え。

4. samtools を使って、eco_ex.sam を bam に変換し、eco_ex.bam として保存せよ
5. samtools を使って、eco_ex.bam をソートし、eco_ex_sorted.bam として保存せよ

現行 samtools は 4.5. の作業は一度にできるが過程確認のため、今回は個別に行う

6. samtools を使って、eco_ex_sorted.bam にインデックスを作成せよ
7. samtools を使って、eco_ex_sorted.bam から以下の遺伝子にマップされたリードを取り出して数を数えよ。抽出された行を数えるには、wc コマンドを使うこと。

染色体名	開始位置-終了位置	遺伝子名
chr	337 - 2799	thrA
chr	4179268 - 4183296	rpoB

復習問題 6 UNIX によるテキストファイル処理

この演習では、**ex6.sam** を使用する。ファイルは下記のパスにある。

~/data/6_text/

ex6.sam ファイルは **SAM** 形式で記されているデータである。内容に関しては「**NGS 基本データフォーマット**」の章を参照すること。

ex6.sam の中で、順鎖にも逆鎖にもマッピングされていないフラグメントがあるかどうかを知りたい。そのために以下の操作をせよ。

1) **ex6.sam** からヘッダー部のみを抜き出して出力せよ。(ヒント: **grep** を使用せよ)

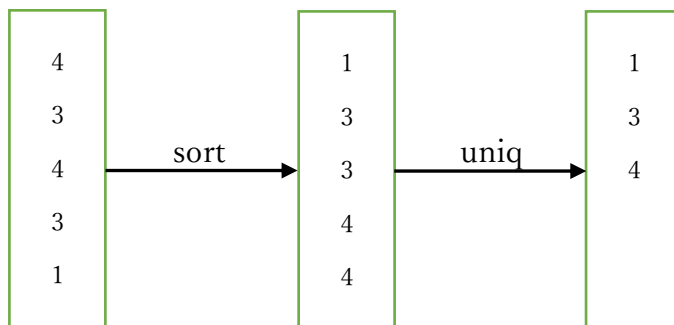
2) **ex6.sam** からヘッダー部以外の行を抜き出し、**ex6_2.sam** として保存せよ。

ex6_2.sam ファイルはマッピング結果部分のみのファイルとなった。

3) **ex6_2.sam** にある各フラグメントの **FLAG** 値を確認するために、「2 カラム目」のみの値を出力せよ。(ヒント: **awk** を使用せよ)

4) 3)で出力された値がどのようなものであるか知りたい。このような場合、パイプ(|)を使い **sort** コマンドを実行してから、**uniq** コマンドを実行することで、「重複している値」を全て取り除くことができる。

例)



3)の結果に対して **sort|uniq** を使用し、どのような値の種類があるかを調べよ。

*5) **ex6.sam** には順鎖にも逆鎖にもマッピングされていないフラグメントがあるか？

あるならば、それはどのフラグメント名であるかを出力せよ。

(ヒント: **SAM** フォーマットの **FLAG** 値はどのような意味を持つかを確認せよ。

その上で、4)で得られた結果と比較して考えるとよい。)

*実践演習 1 RNA-Seq 解析結果の集計(1)

RNA-Seq 解析のデータを使って、Unix や R の基本コマンドを使ってもう一歩先の解析を行ってみよう。~/data/8_ex/rnaseq に移動しよう。この下の **results** ディレクトリには、12 サンプルの RNA-Seq データをそれぞれ大腸菌ゲノムにマッピングした結果から、**htseq-count** コマンド(https://htseq.readthedocs.io/en/release_0.11.1/count.html)を使って、遺伝子ごとにその領域に重なるリード数を単純にカウントした結果(**ecoli.*.htseq**)が格納されている。UNIX コマンドを使って、この結果を整理してみよう。

なお、ここで使うサンプルデータは、NCBI GEO データベースの **GSE59468** からデータを間引いて作成したもので、「UNIX 基本コマンド」の講習で用いたものと同じものである。大腸菌の 2 つの系統を、それぞれ 2 つの異なる条件で培養して、それぞれについて 3 つの反復をとった、計 12 個のサンプルのデータが含まれている。12 個のファイルには連番が振られており、各番号はそれぞれ以下の系統・条件の組合せに対応している。

番号	strain	condition	replicate	SRA accession
1	MG1655	MPOS	1	SRR1515282
2	MG1655	MPOS	2	SRR1515283
3	MG1655	MPOS	3	SRR1515284
4	MG1655	Urine	1	SRR1515285
5	MG1655	Urine	2	SRR1515286
6	MG1655	Urine	3	SRR1515287
7	CFT_073	MPOS	1	SRR1515276
8	CFT_073	MPOS	2	SRR1515277
9	CFT_073	MPOS	3	SRR1515278
10	CFT_073	Urine	1	SRR1515279
11	CFT_073	Urine	2	SRR1515280
12	CFT_073	Urine	3	SRR1515281

以下コマンド先頭の \$ は Unix のプロンプトを表すので、それ以降を改行なしで入力すること。

1) 各 ***.htseq** ファイルには、それぞれのサンプルにおける遺伝子ごとのリード数が記載されている。この 12 個のファイルを、以下のようにすべて横に並べて 1 つのファイルを作りたい。

```
(ecoli.1.htseq)  (ecoli.2.htseq)  (ecoli.3.htseq)  .....  (ecoli.12.htseq)
b0001   11      b0001    0      b0001    7
b0002  117      b0002    9      b0002  131
b0003   33      b0003    1      b0003   31
b0004   44      b0004    4      b0004   58
```



```
b0005  3      b0005  0      b0005  2
```

ファイルを行単位で結合する UNIX コマンドとして、**paste** がある。以下を実行してみよう。

```
$ paste results/ecoli.[1-3].htseq |less
```

paste は引数の順にファイルを結合するので、引数の順番に注意する必要がある。以下のワイルドカードを使った 2 つのコマンドの出力を比べてみよう。ただし、**echo** は受け取った引数をそのままの順で表示するコマンドである。

A) **echo results/ecoli.*.htseq**

B) **echo results/ecoli.?.htseq results/ecoli.1?.htseq**

ファイルが 1, 2, 3, ... の順に並ぶのはどちらか。なお、コマンド B) は、以下のようにより簡潔に書くこともできる（試してみよ）：**echo results/ecoli.{?,1?}.htseq**

この結果を用いて、**paste** コマンドによって **htseq** の出力ファイルを 1, 2, 3, ... の順に結合したファイルを作成し、結果を **ecoli.count_all.tmp1** というファイル名で保存せよ。

2) **ecoli.count_all.tmp1** は、b0001 などの遺伝子名が奇数列に繰り返し出現するため、冗長である。そこで最初の列だけ残して、あとの遺伝子名の列は除きたい。すなわち、1, 2, 4, 6, 8, ..., 22, 24 列目のみを残したい。これは **awk** を使っても行えるが、**cut** の方がより簡潔に書ける。**cut** は **-f** オプションによって指定された列のみを出力する。たとえば、**cut -f 1,3,4 file** は、**file** からタブ区切りで 1, 3, 4 番目の列を抜き出して表示する。**cut** コマンドを使って上記の処理を行い、結果を **ecoli.count_all.tmp2** というファイルに保存せよ。

3) **ecoli.count_all.tmp2** の最後の 5 行は、特定の遺伝子にきちんと対応づけられなかったリードに関する情報が記載されている。**tail** コマンドで確認しよう。

```
$ tail ecoli.count_all.tmp2
```

これらの行はいずれも **__** で始まっており、例えば、**__nofeature** は遺伝子領域以外にマップされたリードの数、**__ambiguous** は複数の遺伝子にまたがる領域にマップされたために 1 つの遺伝子には対応づけできなかったリードの数を表している。これらの行を、**grep** コマンドを使って除き、結果を、**ecoli.count_all** に格納せよ。

結果として、以下のようなファイルができるはずである。

```
b0001    11     0     7   4     7   17  9     0   0   3   7   1
b0002   117     9    131 51    20  161 20    1   0   32  34  2
b0003    33     1     31  10     4   30  9     1   2   7   9   4
.....
b4660     3     0     1   0     1   0   0     0   0   0   0   0
b4661     5     0     2   1     1   6   0     0   0   0   0   0
```

*実践演習 2 RNA-Seq 解析結果の集計 (2)

実践演習 1 で作成したファイルは、各遺伝子について 12 個のサンプルのリード数を記録したタブ区切りのテーブルであり、R で処理するのに適した形になっている。12 個のサンプルは、4 つの条件について、それぞれ 3 つずつのレプリケートをとったデータであった。そこで、R を用いて、各条件についてそれぞれ 3 つのレプリケーションの平均値を計算してみよう。ただし、サンプルごとにリード数の総和が違っているため、それを補正するためにまず標準化（ノーマライズ）を行う必要がある。

ここでは、各サンプルのリード数の総和が 100 万になるように標準化する CPM (Counts per million reads) という値を計算しよう。それには、もとのリード数をサンプルごとに各サンプルのリード数の総和で割って 100 万倍すればよい。なお、標準化については実践編で詳しく説明されるが、異なる遺伝子間で発現量を比較する場合は、リード数に加えて遺伝子の長さについての補正も必要で、そのために RPKM (Read per kilobase per million reads) という値がよく用いられる。これはもとのリード数を、遺伝子の長さ 1,000 bp あたり、各サンプルのリード数の和 100 万あたりになるように標準化したものである。ここではより簡単な CPM を計算するが、同じ遺伝子の発現量をサンプル間で比較するだけの場合にはこれも有効な指標である。

以下でコマンド先頭の `>` は R のプロンプトを表すので、それ以降を改行なしで入力すること。

1) まず R の作業ディレクトリを `~/data/8_ex/rnaseq` に移動しよう。メニュー（その他→作業ディレクトリの変更）から移動しても、コンソールから `setwd("ディレクトリ名")` を打ち込んでもよい。移動したら、`getwd()` で正しく移動できていることを確認しよう。

次に、`ecoli.count_all` からデータを読み込む。`read.table` 関数を使ってデータを読み込み、変数 `eco_rna` に代入しよう。このファイルは、セパレータはタブで、ヘッダはなしである。1 列目に遺伝子名が入っているので、これを各行の「名前」として指定しよう。これには、`read.table` のオプションとして `row.names=1` を指定する。このとき、`row.names` で指定した列（1 列目）が行の名前として読み込まれる。

読み込んだら、`head` 関数で先頭数行を表示して、正しく読めているかを確認しよう。

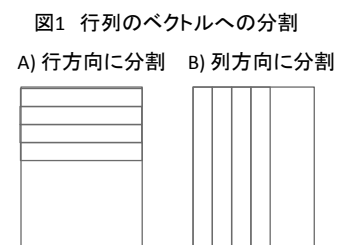
```
      V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13
b0001  11  0  7  4  7 17  9  0  0  3  7  1
b0002 117  9 131 51 20 161 20  1  0 32 34  2
b0003  33  1  31 10  4  30  9  1  2  7  9  4
....
```

最初の行で `V1` がないことに注意。1 列目はデータではなく、行の名前として読み込まれている。行に名前をつけると、`eco_rna["b0020",]` のように名前で行を抽出することが可能となり、興味のある遺伝子の情報を抽出したい場合などには便利である。

なお、1 行目はヘッダを表しており、最初のデータである **b0001** は 2 行目から始まる。「入力ファイルにヘッダあり」として読み込んでしまうと行がずれてしまうので注意しよう。また、今回はファイルにヘッダ行が含まれないので、各列にはデフォルトで **R** がつけた名前 (**v+元のファイルの列番号**) がついている。列名を変更したい場合は **colnames** という関数で行えるが、ここでは省略する。

2) CPM を計算するには、**eco_rna** の値をそれぞれのサンプルのリード数の和で割る必要がある。この和をまず計算しよう。**eco_rna** の各列に各サンプルのリード数が入っているので、それぞれの列を抽出したベクトルについて和をとるとよい。ベクトルの和をとるのは **sum** 関数で行えるので、たとえば **eco_rna** の 1 列目の和は **sum(eco_rna[,1])** で計算することができる (試して見よ)。この計算を各列について繰り返し行えばよい。

このように、行列やデータフレームから各行または各列をベクトルとして抽出して、それに特定の関数を適用する処理を繰り返すときは、**apply** 関数を使うのが便利である。**apply** は、**apply(x, MARGIN, FUN)** のように 3 つの引数をとる。**x** は入力データとなる行列またはデータフレーム、**MARGIN** は 1 または 2 をとり、1 は行 (横) 方向 (図 1A)、2 は列 (縦) 方向 (図 1B) のベクトルに分割した上で、行列全体にわたって処理を繰り返すことを指定する。**FUN** は適用する関数名である (**help(apply)** で確認せよ)。



この **apply** を使って、**eco_rna** の各列について列方向にベクトルを抽出し、**sum** 関数を適用して、結果を **eco_rna_readsum** という変数に格納せよ。結果は要素数 12 のベクトルになるはずである。その最初の要素が **sum(eco_rna[,1])** と一致することを確認せよ。

3) CPM を計算するには **eco_rna** の各行について、値を **eco_rna_readsum** の同じ位置の値で割って 1,000,000 倍すればよい。今回は列方向ではなく、行方向にベクトルを抽出して計算することになるが、再び **apply** 関数を使うことができる。

まずここで行う計算を確認しよう。たとえば 1 行目については以下のベクトルの割り算を行う。

```
> eco_rna[1,] / eco_rna_readsum
```

apply を使うには関数の形にする必要があるが、実は割り算の演算子 **/** は、クオートで囲むことによって 2 つの引数を持つ関数名としても表すことができる。例えば、**6 / 3** は、関数形で表すと、**'/'(6,3)** と書ける (試して見よ)。

従って、上記の計算を行うには **apply** に関数名として **'/'** を与えればよい。ただし、**sum** と違って **'/'** は引数を 2 つとる。1 つめの引数は **apply** 内部で与えられるが、2 番目の引数は外から与えなければならない。**apply** 関数では 4 番目以降の引数に、適用する関数の 2 番目以降の引数を与えることができるので、除数 **eco_rna_readsum** を 4 番目の引数として指定する。これで行方向に関数を適用するように **apply** 関数を実行すればよい。**apply** を実行した後で全体

を 1000000 倍する。この結果を `eco_rna_cpm0` という変数に代入せよ。

これを `head` で確認すると、結果が大量に表示されて流れてしまうはずである。実は、ここでの結果は行と列が入れ替わってしまっており（これは `apply` 関数の仕様）、1 行の長さが非常に長くなっている。これは `dim` 関数で確認できる。`dim` 関数は、指定した行列やデータフレームの行数と列数を表示する。`dim(eco_rna)` と `dim(eco_rna_cpm0)` を比較してみよう。

これでは扱いづらいので、`eco_rna_cpm0` の行と列を入れ替えよう。これは転置行列をとる (`transpose`) という操作であり、R では `t` 関数で行う (`help(t)` で確認せよ)。この結果を `eco_rna_cpm` に代入しよう。`head` で結果を確認すると、以下のようになるはずである。

	V2	V3	V4	V5	V6	V7	V8
b0001	107.05284	0.00000	67.87681	105.54368	260.74648	119.41976	72.98145
b0002	1138.65289	776.73255	1270.26608	1345.68194	744.98994	1130.97538	162.18101
b0003	321.15851	86.30362	300.59732	263.85920	148.99799	210.74075	72.98145

4) 最後に、標準化した値を使って、サンプルごとに 3 つのレプリケートの平均値を計算しよう。各行（遺伝子）について、各サンプルは、それぞれ 1-3, 4-6, 7-9, 10-12 列目に対応しているので、それらの値の平均値を計算すればよい。

再び `apply` 関数を使おう。そのために、まず各行について、上記の 4 つの平均値を計算するための関数 `calc_means` を作る。関数の作成は、コマンドラインからそのまま打ち込んでもよいが、ちょっと複雑なのでエディタを使って行うとよいだろう。

```
calc_means <- function(v) {  
  c( mean(v[1:3]), mean(v[4:6]), mean(v[7:9]), mean(v[10:12]) )  
}
```

この関数は、引数に `v` として各行における要素 12 のベクトル値をとり、それを 3 要素ずつの 4 つのベクトルに分けて、それぞれの平均値を計算し、`c()` 関数によって値をまとめたベクトルとして返している。`calc_means(1:12)` を実行して動作を確認しよう。

前問と同様にして、`apply` 関数によってこの関数を `eco_rna_cpm` に適用し、サンプルごとの平均値を計算しよう。その際、行と列の入れ換えにも注意すること。正しく作成できれば、`head` で確認すると、以下のような結果が得られるはずである。

	[,1]	[,2]	[,3]	[,4]
b0001	58.30988	161.90331	24.32715	46.25669
b0002	1061.88384	1073.88242	106.92005	222.94657
b0003	236.01981	207.86598	153.35993	116.14609