2017 GITC 大規模なデータ解析のための UNIX 入門　演習問題

復習問題１　UNIX 基本コマンド

1. bias4 へログインして、~/data/1_unix ディレクトリに移動しカレントディレクトリを確認します。加えてカレントディレクトリ(~/data/1_unix)以下にある全てのファイルを確認しましょう。オプション"-R"を使うとディレクトリを辿りながら全てのファイルを表示します。

2. ~/data/1_unix/sprot ディレクトリ内の FASTA ファイル全てを~/unixtest/FASTA-EX ディレクトリにコピーしましょう。ディレクトリがない場合は新規に作成します。上記の操作を行った後に、~/unixtest/FASTA-EX ディレクトリ内のファイルリストを確認しましょう。

3. ~/unixtest/FASTA-EX ディレクトリに移動して、コピーした FASTA ファイル全てのタイトル行のみを抜き出して less コマンドで確認しましょう。ただし、ここではすべてのファイルに配列は 1 つずつ入っています。確認方法は、grep コマンドを使用します。オプションを使用しないと結果にファイル名が付加されてしまうので、ファイル名を表示しないオプションを man コマンドで調べましょう。調べる際には"file name"をキーワードとして検索します。

4. 同じように、FASTA ファイル全ての配列行のみを less コマンドで確認しましょう。こちらは、tail コマンドのみを使用します。3.と同様にファイル名が付加されてしまうので、man コマンドでファイル名を付加しないオプションを調べます。

5. ~/unixtest/FASTA-EX にコピーした FASTA ファイルから HUMAN,MOUSE,DROME に関する(ファイル名に含まれる) Multi-FASTA 形式のファイルをそれぞれ human.fasta, mouse.fasta, drome.fasta として~/unixtest/の下に cat コマンドを使用して作成しましょう。

*6. FASTA-EX ディレクトリを削除しますが、削除する前にこのディレクトリ全体の圧縮アーカイブを~/unixtest/FASTA-EX.tar.gz として tar コマンドで作成しましょう(tar のオプションは"zcvf"を使用)。作成できたら、圧縮アーカイブの中身を確認(tar のオプションは"ztvf"を使用)してから FASTA-EX ディレクトリ全体を削除します。

*7. 圧縮アーカイブ FASTA-EX.tar.gz は自分自身のみ読み書きできるよう chmod コマンドを使用して(グループとその他は読み書きできないよう)アクセス権を変更します。

*8. UNIX にはファイルの検索に用いる 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 コマンドの使用方法を確認し、この find コマンドを使用してホームディレクトリ配下のどこかにある"SYYM_DROME.sprot"を探しましょう。

*9. 同じく、find コマンドでは探しだしたファイル（複数も可能）を対象にして（引数にして）コマンドを実行することができます（-exec を使用)。この機能を使って、ホームディレクトリ配下にある Swiss-Prot ファイル(拡張子.sprot)を全て探し出して ~/SP/ ディレクトリに一度にコピーしましょう。

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 を作成せよ。新規にファイルを作成してもよいし、(もしあるなら)exapmle3.sh を修正したあと、別名で保存しても良い。
ファイルの編集には Emacs を使用せよ。

3）$ ./exapmle4.sh GO
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、実行権が付与されているかを確認せよ。

この example4.sh は、実行した際に .tmp という中間ファイルが作成される。
中間ファイルは必要ないので、これを最終的に消したい。
そうなるように example4.sh を編集せよ。

編集した後のスクリプトを実行し、 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　UNIX によるテキストファイル処理

この演習では、ecoli.htseq を使用する。ファイルは下記のパスにある。
~/data/7_ex/ex5/
ecoli.htseq には、アノテーションテーブルを使って、各遺伝子にマッピングされたリード数をカウントしたものが書かれている。

1) ecoli.htseq に記載されていて、カウントされた値が 100 回以上ある行を標準出力 に表示せよ。

2) そのうち、必要なのは b**** という文字列から始まる行である。それらのみを取り出して、ecoli.temp というファイルに保存せよ。

3) ecoli.temp を、カウントされた回数の多い順にソートし、ecoli.htseq.sorted というファイルに保存せよ。

4) 1~3)の操作を一つのコマンドで行えるように、htsort.sh というシェルスクリプトを作成せよ。
シェルスクリプトの実行後に中間ファイル(ecoli.temp)を消せることができていればなお良い。

*5) 4)で作成したスクリプトは、ecoli.htseq という特定のファイルにのみ使用できる。上記の処理を他のファイルに対しても行う必要が出てきた。
4)で作成したスクリプトを、引数を 1 つ使用する形に書き換えよ。

例えば、ecoli_other.htseq というファイルに対し、

```
$ ./htsort.sh ecoli_other.htseq
```

というコマンド書式で、別のファイルに対しても同じことが行えるようにせよ。

復習問題 4　SunGridEngine 使用方法＆シェルスクリプト

~/data/6_script/extra に移動せよ

このディレクトリの中には、eco_1.fq 〜 eco_12.fq の 12 つの fastq ファイル（のシンボリックリンク）があります。

1. これらの fastq ファイルを ~/data/3_sge/genome/ecoli_genome にマッピングしたい。プログラムは bowtie2 を使い、結果の sam ファイルは ~/data/6_script/extra/results 内に保存すること。結果用ディレクトリは mkdir で作成すること。sam ファイル名は eco_1.sam 〜 eco_12.sam とし、実行時間を短縮するためにアレイジョブ機能を使って並列化すること。

*2. 1.で実行した結果の sam ファイルの名前を、元の：シンボリックリンクでない fastq ファイルの名前（CFT_MP-1.fastq〜MG_Ur-3.fastq）と結びつけ、名前を変更することを考える。for 文でこれを実現するにはどうしたら良いか？ extra.sh を参考にして考えてみましょう。

   ・ファイル名の変更には mv を使いますが、間違って mv を実行してしまうと元に戻すのが大変なので**必ず** mv コマンド全体を echo で出力してみて、期待通りの動作をするか確認してから本番実行すること。

*実践演習１　RNA-Seq 解析結果の集計

大腸菌の RNA-Seq の例題をもう一度考える。`~/unix15/rnaseq` に移動しよう。この下の `results` ディレクトリには、大腸菌ゲノムにマッピングした結果から `htseq-count` を使って各遺伝子領域に重なるリード数を集計した結果(`*.htseq`)が格納されている。この結果を基に、UNIX コマンドを使って、簡単な解析結果の集計を行ってみよう。

なお、ここで使ったサンプルデータ（NCBI GEO データベースの GSE59468）には、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おり、ファイルについた番号は、それぞれ以下の系統・条件の組合せに対応している（ただし、サイズを縮小するためにデータを大幅に間引いている）。

	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

1) 各 `*.htseq` は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)   (ecoli3.htseq)   ..... (ecoli12.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、結果をファイル `ecoli.paste_all` に格納せよ。

2) `ecoli.paste_all` 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,...,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。
また、`ecoli.paste_all` の最後の 5 行は遺伝子領域にマッチしなかったリード数が記載されているが、これらの行を除きたい。`grep` コマンドを使って除くことを考えよ。これらの処理を行った結果を、`ecoli.count_all` に格納せよ。

3) 発現解析でよく行われる標準化の方法に RPKM (Read Per Kilobase per Million mapped reads)がある。これを計算するには各遺伝子の長さの情報が必要である。遺伝子アノテーションファイル `ecoli.gtf` には各遺伝子（`exon`）の開始位置と終了位置の情報が含まれている。`awk` を使ってこれから遺伝子の長さを計算しよう。3 列目が exon である行について、開始位置（4 列目）と終了位置（5 列目）を使って長さを計算し、10 列目にある遺伝子名、長さの順に出力する。

正しく実行すると、以下のような出力が得られるはずである。

```
"b0001"; 66
"b0002"; 2463
"b0003"; 933
"b0004"; 1287
"b0005"; 297
.....
```

ダブルクォート(`"`)とセミコロン(`;`)が余分なので、取り除こう。これには `sed` コマンドが使える。`sed` はファイルの簡単な編集を行うプログラムで、'`s/`置換する文字列`/`置換後の文字列`/g`'によって、ファイル中の指定した文字列を一斉に置換することができる。置換後の文字列を空にすることで、特定の文字列を除去することもできる。また、置換する文字列には正規表現が使えるので、〔ダブルクォートまたはセミコロン〕という指定は正規表現を使って書ける。
そこで、以下のコマンドによって、file 中のすべてのダブルクォートとセミコロンを除くことができる。

```
$ sed 's/["";]//g' file
```

これによって、以下のような出力が得られるはずである。

```
b0001 66
b0002 2463
b0003 933
b0004 1287
b0005 297
....
```

これをファイル ecoli.gene_length に保存せよ。

4) ecoli.count_all と ecoli.gene_length を結合して一つのファイルにしよう。実は、ecoli.count_all は遺伝子名でソートされているが、ecoli.gene_length は遺伝子名でソートされていない（ecoli.gtf が位置によってソートされているため）。従って、そのまま paste すると正しく結合されない（確認せよ）。そこで、まず ecoli.gene_length を遺伝子名によってソートし、ecoli.gene_length_sorted というファイルに保存しよう。

ecoli.count_all と ecoli.gene_length_sorted を paste コマンドで結合することもできるが、また遺伝子名の行が余分にできてしまう。ここでは別のコマンド join を使った結合をしてみよう。join は２つのファイルに共通の列（ここでは１列目の遺伝子名）があり、かつファイルがいずれもそれらの列に関してソートされているとき、それらの列をキーとして２つのファイルを対応づけて結合するコマンドである。

```
$ join -1 1 -2 1 ecoli.gene_length_sorted ecoli.count_all
```

引数の -1 1 -2 1 は１番目のファイルの１列目と２番目のファイルの１列目をキーとして対応づけることを意味しているが、これはデフォルトの動作なので、省略しても同じ結果になる。

paste はキー（ここでは遺伝子）の並びが２つのファイルで完全に一致していないと、行が合わなくなって意味がなくなるが、join はキーに重複や抜けがあって２つのファイルの間でずれていても、同じキーを持つ行を正しく対応づけてくれる。非常に強力なコマンドである。

なお、join はタブ区切りのファイルをスペース区切りに変更してしまう。見やすさを維持するためにタブ区切りに直したい場合は、sed ‘s/ /　　/g’ file によって、タブ区切りに変換できる。ただし、sed コマンドの最初の文字列はスペースで２番目はタブである。コマンドラインでタブを入力するには Control-v を押してから Tab キーを押す。または、この場合タブの代わりに¥t を入れても同じ意味になる。
これによって以下のようなファイルができるはずである。

```
b0001    66    11   0   7    4   7   17   9   0   0   3   7  1
b0002    2463  117  9   131  51  20  161  20  1   0   32  34 2
b0003    933   33   1   31   10  4   30   9   1   2   7   9  4
.....
```

この結果を ecoli.result_all に保存しよう。

5) `ecoli.result_all` のようなタブ区切りテキストは、Excel などの表計算ソフトや R などの統計解析ソフトに読み込んで処理できる。これ以降はおそらくそのようにした方が早いが、UNIX の練習のため、もう少し UNIX コマンドを使った解析を続けよう。

RPKM を計算するには、あと各サンプルでマップされたリード数の和を計算する必要がある。これも `awk` で計算することはできるが、やや面倒なのでここでは省略しよう。

`ecoli.result_all` は 1 行目に遺伝子名、2 行目に遺伝子の長さが入っており、3,4,5 列目、6,7,8 列目、9,10,11 列目、12,13,14 列目が、それぞれ 4 つの条件の組合せについての 3 回ずつの反復実験の結果に対応している。この 3 回の反復実験の結果を平均し、さらに遺伝子の長さで割った上で 1000bp あたりの値に標準化しよう。1 列目に遺伝子名、2,3,4,5 列目に各条件の反復実験の平均値を入れる。この処理を、`awk` を使って行い、結果を `ecoli.result_mean` に格納せよ。

*実践演習２　BLAST を用いた比較ゲノム解析

配列データを解析する上で、類似性検索プログラム BLAST は最もよく用いられるツールの一つである。BLAST 検索をゲノム規模のデータに対して行うには、UNIX コマンドを用いて行うのが効果的である。ここでは、大腸菌のゲノム配列と、サルモネラ菌(*Salmonella enterica* subsp. *enterica* serovar Typhymurium)のゲノム配列を、BLAST を使って比較してみよう。

まず、`~/data/7_ex/genome_comparison` に移動しよう。`ecoli_genome.fa` が大腸菌のゲノム配列、`salmonella_genome.fa` がサルモネラ菌のゲノム配列である。まず、`less` でこれらのファイルの中身を確認しよう。`ecoli_genome.fa` は、マッピングの実習で用いたものと同じ配列だが、ゲノム比較に用いるために、先頭のタイトル行は `E.coli` となっている（マッピングで用いたものは、タイトルが染色体配列であることを示す `chr` となっていた)。

1) BLAST は基本的にデータベースを検索するツールなので、２つのファイルのうち一方をクエリ、一方をデータベースと考える。ここでは大腸菌をクエリ、サルモネラ菌をデータベースとしよう。BLAST では、検索の前にデータベースに対してインデックスを作成する必要がある。以下のコマンドを実行しよう。`-dbtype` オプションは、データベースのタイプがタンパク質配列ではなく核酸配列であることを指定している。

```
$ makeblastdb –in salmonella_genome.fa –dbtype nucl
```

実行が終わったら ls でファイルを確認しよう。nhr, nin, nsq という拡張子のついたファイルが新たに生成されているはずである。

インデックスファイルができていることを確認したら、早速 BLAST を実行しよう。核酸配列同士の比較を行うには、blastn コマンドを使う。

```
$ blastn –query ecoli_genome.fa –db salmonella_genome.fa –outfmt 6 >
eco_sal.blast
```

このくらいの小さなゲノム間の比較であれば、実行は瞬時に終わる。なお、BLAST オプションの `–outfmt` は出力形式を指定するオプションで、６番は Tabular 形式といって、タブ区切りで各種の情報を出力してくれる最も使いやすい形式である。blastn のオプションの詳細は、
`blastn –help` とタイプすると見ることができる。

2）結果として得られた eco_sal.blast の中身を less で見てみよう。各行は 2 つの配列間で類似性が高い領域のアライメントの情報を表しており、タブ区切りで左から順に、クエリ配列名、データベース配列名、%一致度、アライメントの長さ、ミスマッチの数、ギャップ開始の数、クエリ配列の開始位置、クエリ配列の終了位置、データベース配列の開始位置、データベース配列の終了位置、E-value、スコアとなっている。

```
E.coli   S.typhimurium  92.360  19975  1445  61   3431550  3451476   3576772  3596713  0.0  28358
E.coli   S.typhimurium  94.165  14294  799   29   4173330  4187603   4359965  4374243  0.0  21743
E.coli   S.typhimurium  86.367  14766  1926  58   194847   209581    259294   274003   0.0  16035
E.coli   S.typhimurium  88.723  12317  1326  44   2391224  2403524   2427658  2439927  0.0  14992
E.coli   S.typhimurium  80.722  19426  3496  208  2118186  2137481   2185574  2204880  0.0  14896
```

このファイルはタブ区切りテキストなので、UNIX のテキスト処理コマンドを使って情報の抽出ができる。`eco_sal.blast` を使って以下の問題を考えてみよう（いずれも awk を使う）。

2-1）アライメントが意味を持つには、E-value が十分に小さい必要がある。E-value が 10^{-20}（コンピュータではこれは 1e-20 と表現される）より小さい行だけを抜き出し、結果を `eco_sal.cut20.blast` というファイルに保存せよ。

2-2) BLAST では相補鎖も同時に検索されるが、相補鎖側にマッチしたときはデータベース配列側の開始位置と終了位置が逆になる（開始位置＞終了位置となる）。そのようなアライメントはいくつあるか。

2-3）大腸菌ゲノム上の 1,260,000 から 1,280,000 までの領域とオーバーラップするアライメントをすべて抜き出せ。それらを、大腸菌ゲノム上の開始位置によってソートせよ。
（ヒント：以下の図で、大腸菌ゲノム上の領域が A, B, C のいずれの場合も条件を満たす）