UNIX 入門・実習内容

実習１：MacOSX の UNIX 環境を確認

Dock メニューから「ターミナル」を起動する。
（ターミナルの在処は、アプリケーション/ユーティリティ）

実習２：ディレクトリの中身を見る（ls）

ターミナルのウィンドウをクリックする。
ホームディレクトリ上で **ls** と入力してリターンキーを押す（＝実行する）。
続いて下記もそれぞれ実行する。

ls ..　　ホームディレクトリの一つ上のディレクトリの中身を見る
ls /　　ルートディレクトリの中身を見る

実習３：隠しファイル（ドットファイル）の表示（ls –a）

ホームディレクトリ上で **ls –a** を実行する

実習４：ファイル名の補完

ホームディレクトリ上で **ls da** と入力してタブキーを押してファイル名の補完を試してみる。
同様に、

ls data/HN/sprot/143　と入力してタブキーを押して動作を確認する。
ls data/H <tab>
ls data/HN/s <tab>
ls data/HN/sprot/143 <tab>

実習５：コマンドヒストリ

Control + p (Control キーと p を同時に押す) を何度か入力してみる。
また、Control + n　を何度か入力してみる。
リターンを押すとコマンドが再実行される。

実習６：ディレクトリを移動する（cd）

現在のディレクトリ上で pwd と入力し、現在のディレクトリを確認する。
ディレクトリ data/HN に移動して pwd を入力する。また、ls を実行する。

pwd
cd data/HN
pwd
ls

さらに sprot に移動し、pwd で現在のディレクトリを確認する。

cd sprot

pwd

(/Users/nibb/data/HN/sprot と表示される)

実習７：ワイルドカード

カレントディレクトリは /Users/nibb/data/HN/sprot

このディレクトリ上で、下記コマンドを実行してみる。

ls *.fasta

ls *_HUMAN*

ls 1A2?_HUMAN.fasta

ls 1A2[1-5]*.fasta

ls 1A25_HUMAN.{fasta,phylip}

また、**ls *** も試してみる。

実習８：ファイルの内容を一括表示する（cat）

カレントディレクトリは /Users/nibb/data/HN/sprot

このディレクトリ上で、下記のコマンドを実行してみる。

cat 1A25_HUMAN.fasta

cat *.fasta

実習９：ファイルの部分表示（head, tail）

カレントディレクトリは /Users/nibb/data/HN/sprot

このディレクトリ上で、下記コマンドを実行してみる。

head 1A25_HUMAN.sprot

tail -20 1A25_HUMAN.sprot

cat 1A25_HUMAN.fasta

tail –n +2 1A25_HUMAN.fasta

実習１０：ファイルの内容を見る（less）

カレントディレクトリは /Users/nibb/data/HN/sprot

このディレクトリ上にある 1433B_HUMAN.sprot の内容を less で見る。

less 1433B_HUMAN.sprot

文字列検索も行ってみる（例えば binding を検索（less 内で **/binding** と入力））。

q　を押して less を終了。

参考：less の詳しい操作法は、less を立ち上げた状態で「h」 と打つと確認できる。

実習１１：ディレクトリの作成と削除（mkdir, rmdir）

cd でホームディレクトリに戻って、実習用のディレクトリ unixtest を作成

cd

mkdir unixtest

作成した unixtest ディレクトリに移動し、pwd で /Users/nibb/unixtest と表示されることを確認する。

cd unixtest

pwd

実習１２：ファイルのコピー（cp）

カレントディレクトリは /Users/nibb/unixtest

~/data/HN/sprot/1433B_HUMAN.sprot をカレントディレクトリ (.) にコピーする。

cp ~/data/HN/sprot/1433B_HUMAN.sprot .

できた 1433B_HUMAN.sprot を copyfile にコピーする（ファイルからファイルへのコピー）

cp 1433B_HUMAN.sprot copyfile

新たなディレクトリ Fasta を作成し、~/data/HN/sprot/以下の .fasta ファイルだけを Fasta ディレクトリにコピーする。その際、ワイルドカードを使う。

mkdir Fasta

cp ~/data/HN/sprot/*.fasta Fasta

実習１３：ファイル名の変更（移動）（mv）

カレントディレクトリは /Users/nibb/unixtest

copyfile を newfile に名称変更する。

mv copyfile newfile

新たにディレクトリ Human をつくり、Fasta 以下の HUMAN のファイルだけ を Human ディレクトリ に移動する。その際、ワイルドカードを使う。

mkdir Human

mv Fasta/*HUMAN* Human

参考：実際にワイルドカードを使う際は、事前に ls で対応するファイルが正しく指定されていることを確認した方がよい。

実習１４：シンボリックリンク（別名）の作成（ln –s）

カレントディレクトリは /Users/nibb/unixtest

さきほど移動させたファイル Human/1433B_HUMAN.fasta のシンボリックリンクをカレントディレクトリ（.）に作成する。

ln -s Human/1433B_HUMAN.fasta .

なお、mv や cp と同様に、最後の引数がディレクトリのときは、そのディレクトリ上にオリジナルと同じ名前のシンボリックリンクが作られる。

参考：-s オプションをつけない場合は、ハードリンクと呼ばれ、オリジナルファイルを移動・消去してもリンクを通した参照が維持されるようになる。ただし、ディレクトリのハードリンクは作れないなどの制約がある。

実習１５：ファイル情報の表示（ls –l）

カレントディレクトリは /Users/nibb/unixtest
~/unixtest 上で以下を実行してみる

```
ls -l
ls -lt
```

実習１６：ファイルの削除（rm）

カレントディレクトリは /Users/nibb/unixtest
newfile を消去する。次に Fasta ディレクトリ全体を消去する。rm -rf を使う

```
rm newfile
rm -rf Fasta
```

実習１７：マニュアルの参照（man）

```
man ls
```
を実行してみる。

実習１８：行数・単語数のカウント（wc）

カレントディレクトリは /Users/nibb/unixtest
以下のコマンドを実行してみる。

```
wc 1433B_HUMAN.fasta
wc
This is a pen.
(Control-D)
```

実習１９：パターン検索（grep）

カレントディレクトリは /Users/nibb/unixtest
以下のコマンドを実行してみる

```
grep GO 1433B_HUMAN.sprot
grep ^FT 1433B_HUMAN.sprot
```

実習２０：リダイレクトの例（出力）

カレントディレクトリは /Users/nibb/unixtest
以下のコマンドを実行して GO_count ファイルを作成し、中身を less で確認する。

```
grep GO 1433B_HUMAN.sprot
grep GO 1433B_HUMAN.sprot  > GO_count
less GO_count
```

実習２１：パイプの例

カレントディレクトリは /Users/nibb/unixtest

下記のコマンドを実行してみる。

```
grep GO 1433B_HUMAN.sprot | wc
grep ^FT 1433B_HUMAN.sprot | less
grep ^FT 1433B_HUMAN.sprot | grep HELIX | less
```

実習２２：シェルスクリプト

カレントディレクトリは /Users/nibb/unixtest
~/data/HN/sprot/testpg をカレントディレクトリ（.）にコピーし中身を確認する

```
cp ~/data/HN/sprot/testpg .
less testpg
```

実行してみる。

```
./testpg
```

実行権限がなくエラーとなるため、実行権を付与する

```
chmod +x testpg
./testpg
```

また、最後に./testpg でなく、単に testpg （パス指定なしで実行）とした場合の出力も確認しておく。

```
testpg
```

実習２３：コマンドパス

```
echo $PATH
```
を実行してコマンドパスを確認する。

実習２４：コマンドパスの設定

カレントディレクトリは /Users/nibb/unixtest
~/.bash_profile の内容を閲覧し、コマンドパスに ~/bin が追加されていることを確認する。

```
less ~/.bash_profile
```

ホームディレクトリ配下に bin ディレクトリを作成し、testpg コマンドを bin ディレクトリへ移動する。最後にパスの指定なしで testpg を再度実行してみる。

```
cd
mkdir bin
mv unixtest/testpg bin
testpg
```

実習２５：SAMtools のインストール

Safari を起動し、samtools で検索
SF Download Page -> samtools -> 1.2 へ移動

samtools-1.2.tar.gz2 をクリックしてダウンロード。
~/Downloads/に保存されるので、~/Downloads から ~/unixtest にファイルを移動してから展開する。
ソースディレクトリで make を実行した後、実行コマンド samtools, bcftools, vcfutils.pl を ~/bin/ にコピーする。

```
cd
mv Downloads/samtools-1.2.tar.gz2 unixtest
cd unixtest
tar xvfj samtools-1.2.tar.bz2
cd samtools-1.2
less INSTALL
make prefix=~ install
samtools
export MANPATH=$MANPATH:~/share/man
man samtools
```

備考：「INSTALL」というファイルを less で読み、インストール方法を確かめている。
慣例的に「prefix=」で指定したディレクトリ直下の「bin」ディレクトリにインストールされるので、ここでは「~/bin」にインストールするために「prefix=~」を指定している。
samtools のように make 時に prefix を指定できるものは少なく、configure 時に prefix を指定することの方が圧倒的に多い。（INSTALL や、README というファイルが同梱されているので、手順を確認してからインストール作業をすると良い）。
最後から 2 番目の行は、同時に samtools のマニュアルも「~/share/man」にインストールされるので、man コマンドでそれを見られるようにするための設定。最後の行で samtools のマニュアルを確認している。

ex1: TopHat

キイロショウジョウバエ Drosophila melanogaster のRNA-seqを行った。ライブラリは2種類。それぞれsingle end（インサートの片側だけ読む）で75bpシークエンスした。10万リード得られた。これらのリードをD. melanogasterのゲノムにマッピングしたい。TopHatを用いてsplice-awareなマッピングを行う。

Data

Input reads (''~/data/EX/'' 以下にある）

- C1_10k_Read1.fq
- C2_10k_Read1.fq

Reference

- D. melanogaster genome and annotation (Ensembl BDGP5.25)

Notes

本来はD. melanogaster genome and annotation (Ensembl BDGP5.25) をiGenomes (http://tophat.cbcb.umd.edu/igenomes.html) からダウンロードする。

- ftp://igenome:G3nom3s4u@ussd-ftp.illumina.com/Drosophila_melanogaster/Ensembl/BDGP5.25/Drosophila_melanogaster_Ensembl_BDGP5.25.tar.gz

ただ、ファイルサイズが比較的大きくダウンロードに時間がかかると思われる。演習用のMacに同じファイルが置いてある (''~/data/EX/'' 以下にある）ので、今回はそれを使って欲しい

- Drosophila_melanogaster_Ensembl_BDGP5.25.tar.gz

Setup

Setup environment

ex1 ディレクトリをつくり、以下の解析はその下で作業しよう。

```
$ mkdir ex1
$ cd ex1
```

dataのコピー

```
$ cp ~/data/EX/C1_10k_Read1.fq ./
$ cp ~/data/EX/C2_10k_Read1.fq ./
$ cp ~/data/EX/Drosophila_melanogaster_Ensembl_BDGP5.25.tar.gz ./
```

Sequence reads

''less'' などのコマンドで、 C1_10k_Read1.fq の内容を確認する。

注）本番の解析では、リード数の確認、フォーマットの確認、クオリティの確認などを行う。必要であればアダプター配列の除去、低クオリティ部位のトリムも行う。今回の演習ではスキップ。

Reference sequence and annotation files

Drosophila_melanogaster_Ensembl_BDGP5.25.tar.gz を解凍する

$tar xzvf Drosophila_melanogaster_Ensembl_BDGP5.25.tar.gz

Run tophat

TopHatを実行。

まず、C1_10k_Read1.fq　をマッピングしよう。

```
$ tophat -p 4 -G genes.gtf  -o C1_tophat_out  genome  read.fq
```

上のコマンドが基本形（そのままコピペしても動かない）。genes.gtf, genome, read.fq の部分には適切なファイル名等をいれて実行しよう。以下を参考にしてほしい。

- ''gene.gtf'' はknown transcriptが記録されたgtfファイル。ファイルパスを指定すること。ダウンロードした Drosophila_melanogaster_Ensembl_BDGP5.25　の中のどこかにあるので探してみよう。
- ''genome'' にはbowtie2用のゲノムのインデックスファイルのbase nameを指定する。ダウンロードした Drosophila_melanogaster_Ensembl_BDGP5.25　の中のどこかにあるので探してみよう。
- read.fq にはマッピングしたいシーケンスのファイルをfastqフォーマットで与える。
- -p は使うCPU coreを指定するオプション。使用するコンピュータのスペックに合わせて。
- 発展： --transcriptome-index　オプションは指定した方が良い。初回に作製したbowtie2 indexが2回目以降使い回せる。複数ライブラリを解析する際は大幅に時間の節約になる。今回は無視してよい。
- tophatコマンドのオプションや引数について詳しく知りたいときは、tophat -h としてヘルプ画面を表示させる。
- 解答 -- 自分で動かせるようになるまで見ない。

同様にC2_10k_Read1.fq をマッピング。

```
$ tophat -p 4 -G genes.gtf  -o C2_tophat_out genome C2_10k_Read1.fq
```

Inspect Results

計算が終わったら、どのようなファイルが生成されたか確認する。

```
$ ls -l C1_tophat_out/
total 1048
-rw-r--r--  1 shige staff 1028268 Mar  6 23:10 accepted_hits.bam
-rw-r--r--  1 shige staff      52 Mar  6 23:10 deletions.bed
-rw-r--r--  1 shige staff      54 Mar  6 23:10 insertions.bed
-rw-r--r--  1 shige staff   25506 Mar  6 23:10 junctions.bed
drwxr-xr-x 17 shige staff     578 Mar  6 23:04 logs
-rw-r--r--  1 shige staff      66 Mar  6 23:04 prep_reads.inf
```

prep_reads.info の中身を''less''で確認しよう。

accepted_hits.bam がアライメント結果である。中身を''samtools''で確認しよう。

```
$ samtools view C1_tophat_out/accepted_hits.bam |less
```

IGV

IGV で可視化しよう。

IGVでbamファイルを読むためには、インデクシングをしなければいけない。sort => indexing の段階をふむ。

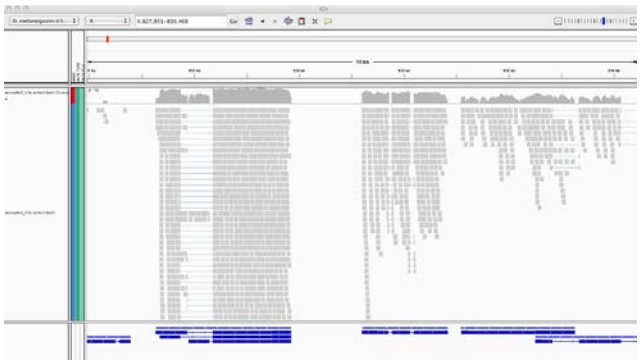
```
$ samtools sort accepted_hits.bam accepted_hits.sorted
# => accepted_hits.sorted.bam ができる
$ samtools index accepted_hits.sorted.bam
# => accepted_hits.sorted.bam.bai ができる
```

1. IGVを立上げる。

2. 左上のプルダウンメニューからDrosophila melanogaster のゲノムを選ぶ。(実は今回使っているリファレンスと同一のバージョンのD. melanogaster のゲノムデータではないが今回の練習ではr5.33を選んで問題はない)

3. メニュー　File > Load from File ... => accepted_hits.sorted.bam を選択

4. 適当な染色体の適当な場所を指定し、適当にズームアップする。（今回はX:830,000付近を見て欲しい）



X:830,000 近辺

注：今回は練習のために、X:830,000　付近にマップされるリードのみを利用しているため、その他の領域ではマッピングはほとんど見られない。

ex2: Transcript-based Mapping with Bowtie2

マウス Mus musculus のRNA-seqを行った。ライブラリは1種類のみで、single end (片側 Read1のみ) 75bpシークエンスを行った。これらのリードをマウスmRNAリファレンスにマッピングさせたい。

戦略：bowtie2でmRNAリファレンスにマッピング。

Data

データファイルは、~/data/SS 以下に保存してある。

Input reads

- IlluminaReads1.fq

Reference

- minimouse_mRNA.fa

Setup

Setup environment

ex2 ディレクトリをつくり、以下の解析はその下で作業しよう。

Sequence reads

''less'' などのコマンドで、 シーケンスファイル（IlluminaReads1.fq）の内容を確認する。

注）本番の解析では、リード数の確認、フォーマットの確認、クオリティの確認などを行う。必要であればアダプター配列の除去、低クオリティ部位のトリムも行う。

Reference sequence and annotation files

''minimouse_mRNA.fa'' の内容をless などで確認する。

Create index of reference

$ bowtie2-build reference.fasta output_basename

- reference.fasta : referenceのfastaファイル。今回の場合は minimouse_mRNA.fa（のパス）
- output_basename : 生成されるインデックスファイル群のbase name。

たとえば

```
bowtie2-build Data/RefSeq.MM9.cds.nr.fasta myref
```

を実行すると、

```
myref.1.bt2  myref.4.bt2
myref.2.bt2  myref.rev.1.bt2
myref.3.bt2  myref.rev.2.bt2
```

の6つのファイルができる。

Run Bowtie2

bowtie2でマッピングしよう。

```
Usage:
  bowtie2 [options]* -x <bt2-idx> {-1 <m1> -2 <m2> | -U <r>} [-S <sam>]
```

bowtie2には様々なオプションがあるが今回は最低限のオプションだけを設定して実行する。どのようなオプションが利用可能かは、''bowtie2 -h'' で確認できる。また開発者ホームページに詳細な解説がある。本番の解析では、適切なオプションを適切なパラメータで実行しなければいけない。実際は、いくつかパラメータを振って試行錯誤することになる。

```
$  bowtie2 -p 4 -x RefSeq.MM9.cds.nr -U mouse_200k.left.fq -S out.sam
```

- out.sam がマッピング結果 SAM format
- -p は使うCPUコア数。使用するコンピュータにあわせて設定する。

コマンドを実行するとしばらくして、

```
200000 reads; of these:
  200000 (100.00%) were unpaired; of these:
    114740 (57.37%) aligned 0 times
    68238 (34.12%) aligned exactly 1 time
    17022 (8.51%) aligned >1 times
42.63% overall alignment rate
```

のようなレポートが表示されて終了する。マッピング率など有用な情報なので、テキストファイルにコピー＆ペーストして保存しておくと良い。

Inspect Results

計算が終わったら、どのようなファイルが生成されたか確認する。 (''ls -l''など)

out.sam の内容を確認しよう (''less, head, tail''など).最初の約2万行はヘッダで、アライメントはそのあとに続く。

SAM to BAM

mapping結果を可視化したりカウントしたり、様々な下流解析を行うために、SAMファイルをsort済のBAMに変換する。そしてインデクシングする。SAM <=> BAM の変換は、NGS解析ではよく行う作業なので必ず身に付けること。

```
$ samtools view -bS out.sam > out.bam
$ samtools sort out.bam out.sorted
# => out.sorted.bam が生成される
$ samtools index out.sorted.bam
# => out.sorted.bam.bai が生成される
```

(optional) Count by transcript

samtoolsを使って、transcriptごとにカウントする簡易な方法を紹介する。

amtoolsのサブコマンド idxstats　は　reference sequenceのエントリー毎にマップされたリード数を集計する。今回は各シークエンスエントリーが各トランスクリプトに相当するので、これを利用するとtranscriptごとのカウント情報が得られる。

```
$ samtools idxstats out.sorted.bam
```

ex3: Count data import and scatter plot

''arab2.txt''は、6 libraries (2 groups x 3 biological replicates) のシロイヌナズナRNA-seqのデータである。すでにマッピング済みで遺伝子毎のリードカウントがタブ区切りテキストとして提供されている。このexerciseでは、テーブルの中身を確認しデータの概要を把握する基本テクニックを習得する。

Data

(~/data/SS/以下にある)

- arab2.txt : count table

Inspect table with MS Excel

1）表計算ソフトMS Excelを使って ''arab2.txt'' の中身を確認しよう。

2）MS Excelで、m1とm2のscatter plot（散布図）を書いてみよう。このふたつは同一コンディションのbiological replicateなので、発現パターンは両者で良く似ているはずである。次にm1とh1を比較しよう。このふたつはコントロールと実験群の比較なので有る程度の発現パターンの違い（すくなくともm1 vs m2よりも大きい違い）が期待される。

ヒント：xy軸ともに対数をとること。

コメント：ノーマライズなどを施していない生データでもこれだけ豊富な情報が得られることを認識して欲しい。

Inspect table with R

R でテーブルの確認とscatter plotを書いてみよう。

Data import

```
> dat <- read.delim("arab2.txt", row.names=1, head=T)  # read tab-delimited text
> head(dat)
          m1 m2 m3 h1 h2 h3
AT1G01010 35 77 40 46 64 60
AT1G01020 43 45 32 43 39 49
AT1G01030 16 24 26 27 35 20
AT1G01040 72 43 64 66 25 90
AT1G01050 49 78 90 67 45 60
AT1G01060  0 15  2  0 21  8
```

Inspect table

```
> dim(dat)
[1] 26221     6
```

Q1：dimコマンドは何をするものですか？

Q2：また、その結果得られた、26221 と 6　は何を意味しますか？

Inspect data by column

それぞれのライブラリの、リードカウント合計は重要な基礎情報である。計算してみよう。

Q3: それぞれのライブラリのリードカウント合計を求めなさい。

例としてm1カラムの合計を計算する。

```
> sum(dat$m1)
[1] 1902032
```

他のカラムも合計を計算しよう。また、これらは基礎情報として重要なので記録しておこう。

Q4 (やや難): 約2万5千遺伝子にはカウント０のものから非常にたくさんのカウントをもつものがある。カウントの、i)最大値、最小値、平均値、中央値はいくつか調べよう。ii) ヒストグラムを書きなさい。

m1 を例に実行例を示す。

i) 最大値、最小値、平均値、中央値

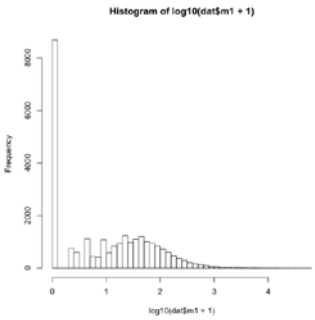
```
> sum(dat$m1)
[1] 1902032
> max(dat$m1)
[1] 61791
> min(dat$m1)
[1] 0
> mean(dat$m1)
[1] 72.5385
> median(dat$m1)
[1] 9
```

ii) ヒストグラム

```
> hist(dat$m1)
```

しかし、このグラフではあまり特徴がつかめないと思う。対数をとってみよう。

```
> hist(log10(dat$m1 + 1), breaks="Scott")
```



Scatter plot

Q5: m1 vs m2 をscatter plotで比較しよう。

```
> plot(dat$m1 + 1, dat$m2 + 1, log="xy")
```

それぞれ+1しているのは、log0は計算できないため。+1して下駄を履かせている。

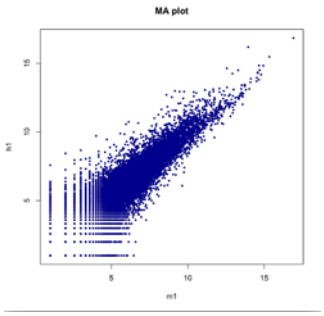
Q6: ほかのライブラリどうしもscatter plotを描いて比較しよう。

Play with scatter plot

plotなどのグラフィックス関数には、様々な引数を与えることによって非常に多くの描画パラメータを変更でき、グラフの見栄えを変更することが出来る。scatter plotの色、形、などを変更する練習をしてみよう。

以下のコマンドをテンプレートにして、col（色）, pch（点の形状）, cex（点の大きさ）, main（グラフのタイトル）, xlab|ylab（x軸やy軸のラベル）を変更して、変化を確認しよう。

```
> plot(log2(dat$m1)+1, log2(dat$h1)+1, col="DarkBlue", pch=16, cex=0.6,
      main="MA plot", xlab="m1", ylab="h1")
```



ex4: MA plot

MA plot は2グループの遺伝子発現を視覚化する便利な散布図である。マイクロアレイ解析でもRNAseq解析でも頻用される。

[Definition of M & A]

- M: log of the ratio = 発現量の比
  - M = log(intensityB / intensityA)
- A: intensity average of log intensity = 発現量の相乗平均
  - A = log(sqrt(intensityA * intensityB))

''arab2.txt'' のデータでMA plotを書いてみよう。(ex3の手続きによってデータを変数datに読み込み済とする)

Q1) m1 vs m2 のMA-plot を書きなさい。

基本形

```
M <- log2(dat$m2 / dat$m1)
A <- log2(sqrt(dat$m1 * dat$m2))
plot(A, M)
```

ただしこのままではエラーが発生する。（なぜか、エラーメッセージをもとに考えてみよう）。

エラー対応と少し見栄えを良くするために、スクリプトを修正。

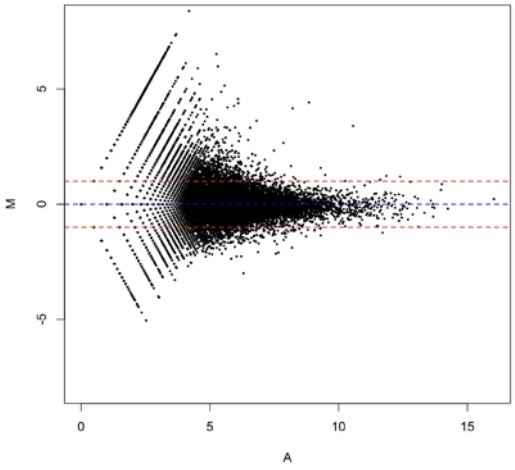
```
M <- log2(dat$m2 + 1) - log2(dat$m1 + 1)
A <- 1/2 * (log2(dat$m2 + 1) + log2(dat$m1 + 1))
plot(A, M, pch=16, cex=0.4, ylim=c(-8,8))
```

コメント：edgeRにはより気の利いたMAplotを描画するコマンドが定義されている。ほかのRNAseq解析用パッケージやマイクロアレイ解析用パッケージにも同様のコマンドが用意されている場合が多い。ただ、MAプロットは自力で作製できるようにしておきたい。

発展：MA plotに、発現比が1, 2, 1/2 を示す線分を追加してみよう。

(例)

```
abline(h=log2(2), col="red", lty=2)
abline(h=log2(1/2), col="red", lty=2)
abline(h=0, col="blue", lty=2)
```



ex5: Differential expression analysis with edgeR

arab2データの遺伝子発現の2群間比較を、edgeRで行う。

edgeRは複雑なパッケージである。開発者が詳細のユーザーガイドやマニュアルを提供しているので、これらを活用して欲しい（リンクは下記参照）。

Import library

```
> library(edgeR)
```

Import data

```
> dat <- read.delim("arab2.txt", row.names=1)

# ... dat中身の確認作業 ...
```

2グループ、各3繰り返し実験、という実験デザインを定義する。

```
> grp <- c("M", "M", "M", "H", "H", "H")
> grp
[1] "M" "M" "M" "H" "H" "H"
```

edgeRのDGEList関数でカウントデータを読み込む。

```
> D <- DGEList(dat, group=grp)
> head(D)
...
```

Normalization

TMM法で、ノーマライズする。calcNormFactorsを使う。

```
> D <-calcNormFactors(D, method="TMM")
```

計算結果の確認

```
> D$samples
   group lib.size norm.factors
m1     M  1902032    1.0399197
m2     M  1934029    1.0611305
m3     M  3259705    0.8841923
h1     H  2129854    1.0266944
h2     H  1295304    1.1412144
h3     H  3526579    0.8747345
```

DE testing

estimate dispersion

```
> D <- estimateCommonDisp(D)
> D$common.dispersion
```

```
[1] 0.342609

> D <- estimateTagwiseDisp(D)
> summary(D$tagwise.dispersion)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.1173  0.1834  0.4728  1.0540  1.7400  3.7390
```

DE test

```
> de.tagwise <- exactTest(D, pair=c("M", "H"))
```

Multiple comparison correction and View results

```
> topTags(de.tagwise)
Comparison of groups:  H-M
             logFC    logCPM       PValue          FDR
AT5G48430 6.233066  6.706315 3.281461e-21 8.604319e-17
AT3G46280 5.078716  8.120404 1.110955e-19 1.456517e-15
AT2G19190 4.620707  7.381817 1.710816e-19 1.495310e-15
AT4G12500 4.334870 10.435847 4.689616e-19 3.074161e-15
AT2G44370 5.514376  5.178263 9.902189e-18 5.192906e-14
AT2G39380 5.012163  5.765848 2.010501e-17 8.786223e-14
AT3G55150 5.809677  4.871425 3.065826e-17 1.148414e-13
AT4G12490 3.901996 10.198755 8.068822e-17 2.455369e-13
AT1G51820 4.476647  6.369685 8.490613e-17 2.455369e-13
AT2G39530 4.366709  6.710299 9.364131e-17 2.455369e-13
```

Dump the table into a text file

```
> write.table(de.tagwise$table, "de.tagwise.txt", sep="\t", quote=F)
```

もしくは、

```
> tmp <- topTags(de.tagwise, n=nrow(de.tagwise$table))
> write.table(tmp$table, "de.tagwise2.txt", sep="\t", quote=F)
```

後者はFDRの値も出力される。

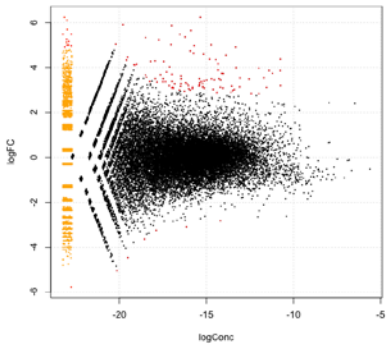
MA plot

edgeR提供のplotSmear関数を使うと便利。

```
plotSmear(D)
```

有意に発現差のある遺伝子を赤色でハイライトすることもできる。

```
> de.names <- row.names(dat[decideTestsDGE(de.tagwise, p.value=0.05) !=0, ])
> plotSmear(D, de.tags=de.names)
```



Inspect DE result

example: fold-change > 10を抽出・カウント

```
> detab <- tmp$table
# get fold-change > 10
> detab[detab$logFC > log2(10),]
> nrow(detab[detab$logFC > log2(10),])
```

example: FC > 5 AND FDR < 0.01

```
> detab[(detab$logFC > log2(2)  & detab$FDR < 0.05), ]
```

edgeRに組み込まれている''decideTestDGE''関数も便利。

```
> summary(decideTestsDGE(de.tagwise, p.value=0.05))
   [,1]
-1    49
0  25903
1    269
```

dumpしたタブ区切りテキストをMS Excelで読み込んで、フィルタ機能やソート機能を駆使してデータを探索するのも良いだろう。

Links

- http://www.bioconductor.org/packages/release/bioc/html/edgeR.html | edgeR
- http://www.bioconductor.org/packages/release/bioc/vignettes/edgeR/inst/doc/edgeRUsersGuide.pdf | edgeR User's Guide

ex6: Clustering

ex6-1

データセットSato_A_thaliana-P_syringae_arvRpt2_6h_expRatio_small.txt(61遺伝子x8遺伝子型）を使って、ユークリッド距離を使った場合とコサイン係数を使った距離でクラスタリングした時の違いを調べなさい（クラスタリング結果の違いの可視化はdendextendライブラリーを使うのが便利である）。このデータはシロイヌナズナ変異体 にバクテリアを感染させた際の発現プロファイルを取り、野生型との比（log2)をとったものである。PR-1は*P.syringae*に対する防御応答の主要なホルモンであるサリチル酸を介したシグナル伝達経路のマーカー遺伝子である。

コサイン係数での距離、クラスタリングは下記 のカスタム関数を用いてよい。また、heatmapおよびheatmap.2(gplotsライブラリー）では引数に Rowv=as.dendrogram(“行のクラスタリング結果”), Colv=as.dendrogram(“列のクラスタリング結果”)と指定することで任意のクラスタリング結果でヒートマップを描くことができる。

準備

```
library(colorspace)
library(dendextend)
library(dendextendRcpp)
library(gplots)

inputMatrix <- read.delim("~/data/MS/Sato_A_thaliana-P_syringae_arvRpt2_6h_expRatio_small.txt",
               header=TRUE, row.name=1)
heatmapColors <- colorpanel(10, low="blue", mid="white", high="orange")
```

実行

1. まずはユークリッド距離を使ってヒートマップを描いてみる。

```
heatmap.2(as.matrix(inputMatrix),
          scale="none",           # 発現量比のスケーリング無し
          trace="none",           # heatmap.2デフォルトのトレースをキャンセル
          # ヒートマップのマス目設定
          sepcolor="black", colsep=0:ncol(inputMatrix), rowsep=0:nrow(inputMatrix),
          sepwidth=c(0.01, 0.01),
          density.info="none",    # ヒストグラム
          col=heatmapColors,
          cexRow=(0.2 + 1/log10(nrow(inputMatrix)))/3*2,
          RowSideColors=ifelse(rownames(inputMatrix)=="At2g14610", "magenta", "grey")
)
```

2. 次にコサイン係数（ベクトルの角度）を距離尺度としたヒートマップを描いてみる。コサイン係数は関数が無いので自作する。

```
cosine.coef <- function(x,y) {
    a <- sum(na.omit(x * y)) / sqrt( sum(na.omit(x)^2) * sum(na.omit(y)^2) )
    return(a)
}

# making a distance table between columns using uncentered Pearson correlation
cosine.table <- function(x) {
    numberOfPoints <- ncol(x)
    columnNames    <- colnames(x)
    distanceTable  <- matrix(data = NA, nrow = numberOfPoints, ncol = numberOfPoints,
                             dimnames = list( columnNames, columnNames )
                            )

    for ( i in 1:(numberOfPoints-1) ) {
        for ( j in (i+1):numberOfPoints ) {
```

```
            v1 <- x[ , i]
        v2 <- x[ , j]
        d  <- 1 - cosine.coef(v1, v2)
        distanceTable[i, j] <- d
        distanceTable[j, i] <- d
    }
    }

    for ( i in 1:numberOfPoints ) { distanceTable[i, i] <- 1 } # fill the diagonal
        return(distanceTable)
}
```

3. 上記関数とas.dist関数を使ってコサイン係数の距離行列を求め、hclust関数でクラスタリングを実行する。

```
# 行のクラスタリング
rowClusters <- hclust(as.dist(cosine.table(as.matrix((t(inputMatrix))))))
# 列のクラスタリング
colClusters <- hclust(as.dist(cosine.table(as.matrix((inputMatrix)))))
```

4. ヒートマップを描く。Rowv, Colv引数にはas.dendrogram関数を介して上記クラスタリング結果を渡す。

```
heatmap.2(as.matrix(inputMatrix), Rowv=as.dendrogram(rowClusters), Colv=as.dendrogram(colClusters),
scale="none", trace="none", sepcolor="black", colsep=0:ncol(inputMatrix), rowsep=0:nrow(inputMatrix),
sepwidth=c(0.01, 0.01), density.info="none", col=heatmapColors,
cexRow=(0.2 + 1/log10(nrow(inputMatrix)))/3*2,
RowSideColors=ifelse(rownames(inputMatrix)=="At2g14610", "magenta", "grey"))
```

5. ユークリッド距離、コサイン係数を使ったクラスタリング結果の違いをdendextendパッケージに含まれる関数を使って可視化する。

```
rowClusters1 <- as.dendrogram(hclust(as.dist(dist(as.matrix((inputMatrix))))))
rowClusters2 <- as.dendrogram(hclust(as.dist(cosine.table(as.matrix(t(inputMatrix))))))
rowDendrogramList <- dendlist(rowClusters1, rowClusters2)
png("tanglegram_row.png", width=480*4, height=480*2, res=200)
tanglegram(rowDendrogramList, common_subtrees_color_branches = TRUE,
columns_width= c(10,3,10), lab.cex=0.6, lwd=2, main_left="Euclidian", main_right="Cosine")
dev.off()
```

ex6-2

同じデータセットを主成分分析を用いて解析しなさい。この演習ではAt2g14610(PR-1)の主成分得点、npr1-1, sid2-2の負荷量などサリチル酸シグナル伝達経路に注目して主成分分析の結果を評価しなさい。

1. 主成分分析を行うpca関数を定義する。

```
pca <- function(dat)                    # データ行列
{
    if (is.null(rownames(dat))) rownames(dat) <- paste("#", 1:nrow(dat), sep="")
    dat <- subset(dat, complete.cases(dat))     # 欠損値を持つケースを除く
    nr <- nrow(dat)                 # サンプルサイズ
    nc <- ncol(dat)                 # 変数の個数
    if (is.null(colnames(dat))) {
        colnames(dat) <- paste("X", 1:nc, sep="")
    }
    vname <- colnames(dat)
    heikin <- colMeans(dat)             # 各変数の平均値
    bunsan <- apply(dat, 2, var)            # 各変数の不偏分散
    sd <- sqrt(bunsan)              # 各変数の標準偏差
    r <-cor(dat)                    # 相関係数行列
```

```
    result <- eigen(r)              # 固有値・固有ベクトルを求める
    eval <- result$values               # 固有値
    evec <- result$vectors              # 固有ベクトル
    contr <- eval/nc*100                # 寄与率（%）
    cum.contr <- cumsum(contr)          # 累積寄与率（%）
    fl <- t(sqrt(eval)*t(evec))         # 主成分負荷量
    fs <- scale(dat)%*%evec*sqrt(nr/(nr-1))     # 主成分得点
    names(heikin) <- names(bunsan) <- names(sd) <-
        rownames(r) <- colnames(r) <- rownames(fl) <- colnames(dat)
    names(eval) <- names(contr) <- names(cum.contr) <-
        colnames(fl) <- colnames(fs) <- paste("PC", 1:nc, sep="")
    return(structure(list(mean=heikin, variance=bunsan,
            standard.deviation=sd, r=r,
            factor.loadings=fl, eval=eval,
            evec=evec, nr=nr,  # added for subsequent PCA projection
            contribution=contr,
            cum.contribution=cum.contr, fs=fs), class="pca"))
}

# print メソッド
print.pca <- function(  obj,                # pca が返すオブジェクト
            npca=NULL,          # 表示する主成分数
            digits=3)           # 結果の表示桁数
{
    eval <- obj$eval
    nv <- length(eval)
    if (is.null(npca)) {
        npca <- sum(eval >= 1)
    }
    eval <- eval[1:npca]
    cont <- eval/nv
    cumc <- cumsum(cont)
    fl <- obj$factor.loadings[, 1:npca, drop=FALSE]
    rcum <- rowSums(fl^2)
    vname <- rownames(fl)
    max.char <- max(nchar(vname), 12)
    fmt1 <- sprintf("%%%is", max.char)
    fmt2 <- sprintf("%%%is", digits+5)
    fmt3 <- sprintf("%%%i.%if", digits+5, digits)
    cat("\n主成分分析の結果\n\n")
    cat(sprintf(fmt1, ""),
        sprintf(fmt2, c(sprintf("PC%i", 1:npca), "  Contribution")), "\n", sep="", collapse="")
    for (i in 1:nv) {
        cat(sprintf(fmt1, vname[i]),
            sprintf(fmt3, c(fl[i, 1:npca], rcum[i])),
            "\n", sep="", collapse="")
    }
    cat(sprintf(fmt1, "Eigenvalue"),   sprintf(fmt3, eval[1:npca]), "\n", sep="", collapse="")
    cat(sprintf(fmt1, "Contribution"), sprintf(fmt3, cont[1:npca]), "\n", sep="", collapse="")
    cat(sprintf(fmt1, "Cum.contrib."), sprintf(fmt3, cumc[1:npca]), "\n", sep="", collapse="")

}
# summary メソッド
summary.pca <- function(obj,                # pca が返すオブジェクト
            digits=5)           # 結果の表示桁数
{
    print.default(obj, digits=digits)
}

# plot メソッド
plot.pca <- function(obj,               # pca が返すオブジェクト
             which=c("loadings", "scores"), # 主成分負荷量か主成分得点か
             pc.no=c(1,2),          # 描画する主成分番号
             ax=TRUE,                   # 座標軸を描き込むかどうか
             label.cex=0.6,         # 主成分負荷量のプロットのラベルのフォントサイズ
                    markers=NULL,
                ...)                # plot に引き渡す引数
{
    which <- match.arg(which)

    if (which == "loadings") {
        d <- obj$factor.loadings
```

```
    }
    else {
    d <- obj$fs
    }

    label <- sprintf("PC%i", pc.no)
    plot(d[, pc.no[1]], d[, pc.no[2]], xlab=label[1], ylab=label[2], ...)

    if (which == "loadings") {
        labelPosition <- ifelse(d[, pc.no[1]] < 0, 4, 2)
        if (is.null(markers) == FALSE){
            for (marker in markers){
                 points(x=d[marker, pc.no[1]], y=d[marker, pc.no[2]], col="magenta", pch=16)
            }
        }

        text(d[, pc.no[1]], d[, pc.no[2]], rownames(obj$factor.loadings),
             pos=labelPosition, cex=1  #label.cex
            )
    }

    if (which == "scores" && is.null(markers) == FALSE){
        for (marker in markers){
             points(x=d[marker, pc.no[1]], y=d[marker, pc.no[2]], col="magenta", pch=16)
             #text(x=d[marker, pc.no[1]], y=d[marker, pc.no[2]], labels=marker, col="red")# At2g14610
        }
    }

    abline(h=0, v=0)
}
```

2. 主成分分析を実行する。

```
PCAresults <- pca(inputMatrix)
```

3. 結果を主成分得点、因子負荷量を用いて評価しなさい。

```
# 主成分得点（scores）から評価
par(mfrow=c(4,7))
for (kPC1 in 1:(ncol(inputMatrix)-1)){
    for (kPC2 in (kPC1+1):ncol(inputMatrix)){
        plot.pca(PCAresults, which="scores", pc.no=c(kPC1, kPC2), cex=0.5, markers="At2g14610")
    }
}

# 因子負荷量（loadings）から評価
par(mfrow=c(4,7))
for (kPC1 in 1:(ncol(inputMatrix)-1)){
    for (kPC2 in (kPC1+1):ncol(inputMatrix)){
        plot.pca(PCAresults, which="loadings", pc.no=c(kPC1, kPC2), cex=0.5,
                 markers=c("npr1.1", "sid2.2"))
    }
}
```

ex6-3

__（発展問題: k-means法はトレーニングコースでは解説していません）__coi1, dde2, jar1, jin1はジャスモン酸シグナル伝達経路、ein2-1, ein3はエチレンシグナル伝達経路、npr1-1, sid2-2はサリチル酸シグナル伝達経路に関わる遺伝子の変異体である。k-means法はデータをいくつのクラスターに分ければよいか見当が付いている場合によいクラスリング方法である。これらの変異体遺伝子発現プロファイルをk-means法を使って解析し、クラスター数の妥当性を考察せよ。kの数は3から始めなさい。同じ処理

例

```
kmeans(t(inputMatrix), centers=3)$cluster
```

を繰り返し、結果の安定性やクラスタリングのされ方を指標に結果を評価しなさい。

ヒント: centers引数、iter.max引数を調整することにより、結果が変化します。

ex6-4

Sato_A_thaliana-P_syringae_arvRpt2_6h_expRatio_full.txt（484遺伝子x22遺伝子型の実データセット）を使って同じ解析をし、より複雑なデータセットを使った場合のクラスタリング結果の違いを検討しなさい。

ex7: Clustering

演習問題1

データセットSato_A_thaliana-P_syringae_arvRpt2_6h_expRatio_small.txt(61遺伝子x8遺伝子型）を使って、ユークリッド距離を使った場合とコサイン係数を使った距離でクラスタリングした時の違いを調べなさい（クラスタリング結果の違いの可視化はdendextendライブラリーを使うのが便利である）。このデータはシロイヌナズナ変異体 にバクテリアを感染させた際の発現プロファイルを取り、野生型との比（log2)をとったものである。なお、コサイン係数での距離、クラスタリングは下記 のカスタム関数を用いてよい。また、heatmapおよびheatmap.2(gplotsライブラリー）では引数に Rowv=as.dendrogram(“行のクラスタリング結果”), Colv=as.dendrogram(“列のクラスタリング結果”)と指定することで任意のクラスタリング結果でヒートマップを描くことができる。

```
library(colorspace)
library(dendextend)
library(dendextendRcpp)
library(gplots)

inputMatrix   <- read.delim("~/data/MS/Sato_A_thaliana-P_syringae_arvRpt2_6h_expRatio_small.txt",
                 header=TRUE, row.name=1)
heatmapColors <- colorpanel(10, low="blue", mid="white", high="orange")

heatmap.2(as.matrix(inputMatrix),
          scale="none",           # 発現量比のスケーリング無し
          trace="none",           # heatmap.2デフォルトのトレースをキャンセル
          # ヒートマップのマス目設定
          sepcolor="black", colsep=0:ncol(inputMatrix), rowsep=0:nrow(inputMatrix), sepwidth=c(0.01, 0.01),
          density.info="none",    # ヒストグラム
          col=heatmapColors,
          cexRow=(0.2 + 1/log10(nrow(inputMatrix)))/3*2,
          RowSideColors=ifelse(rownames(inputMatrix)=="At2g14610", "magenta", "grey")
)

# コサイン係数（ベクトルの角度）でクラスタリング
# コサイン係数は関数が無いので自作する
cosine.coef <- function(x,y) {
    a <- sum(na.omit(x * y)) / sqrt( sum(na.omit(x)^2) * sum(na.omit(y)^2) )
    return(a)
}

# making a distance table between columns using uncentered Pearson correlation
cosine.table <- function(x) {
    numberOfPoints <- ncol(x)
    columnNames    <- colnames(x)
    distanceTable  <- matrix(data = NA, nrow = numberOfPoints, ncol = numberOfPoints,
                             dimnames = list( columnNames, columnNames )
                            )

    for ( i in 1:(numberOfPoints-1) ) {
        for ( j in (i+1):numberOfPoints ) {
            v1 <- x[ , i]
        v2 <- x[ , j]
        d  <- 1 - cosine.coef(v1, v2)
        distanceTable[i, j] <- d
        distanceTable[j, i] <- d
    }
    }

    for ( i in 1:numberOfPoints ) { distanceTable[i, i] <- 1 } # fill the diagonal
        return(distanceTable)
}

# コサイン係数の距離行列
# cosineDistanceTable <- as.dist(cosine.table(as.matrix((dat1))))

# 行のクラスタリング
```

```
rowClusters <- hclust(as.dist(cosine.table(as.matrix((t(inputMatrix))))))
# 列のクラスタリング
colClusters <- hclust(as.dist(cosine.table(as.matrix((inputMatrix)))))

heatmap.2(as.matrix(inputMatrix), Rowv=as.dendrogram(rowClusters), Colv=as.dendrogram(colClusters),
  scale="none", trace="none", sepcolor="black", colsep=0:ncol(inputMatrix), rowsep=0:nrow(inputMatrix),
  sepwidth=c(0.01, 0.01), density.info="none", col=heatmapColors,
  cexRow=(0.2 + 1/log10(nrow(inputMatrix)))/3*2,
  RowSideColors=ifelse(rownames(inputMatrix)=="At2g14610", "magenta", "grey"))


rowClusters1 <- as.dendrogram(hclust(as.dist(dist(as.matrix((inputMatrix))))))
rowClusters2 <- as.dendrogram(hclust(as.dist(cosine.table(as.matrix(t(inputMatrix))))))
rowDendrogramList <- dendlist(rowClusters1, rowClusters2)
png("tanglegram_row.png", width=480*4, height=480*2, res=200)
tanglegram(rowDendrogramList, common_subtrees_color_branches = TRUE, columns_width= c(10,3,10),
  lab.cex=0.6, lwd=2, main_left="Euclidian", main_right="Cosine")
dev.off()


colClusters1 <- as.dendrogram(hclust(as.dist(dist(as.matrix(t(inputMatrix))))))
colClusters2 <- as.dendrogram(hclust(as.dist(cosine.table(as.matrix((inputMatrix))))))
columnDendrogramList <- dendlist(colClusters1, colClusters2)
tanglegram(columnDendrogramList, common_subtrees_color_branches = TRUE)
```

Case study 1: Genome-based RNA-Seq pipeline

アラビドプシス(Arabidopsis thaliana)のRNA-seqを行った。ライブラリは2D sample (2days dark conditionで生育させた黄色芽生え)と2D2L sample (その後さらに2days light conditionで生育させた緑化芽生え）でそれぞれsampling duplicateを３つ用意した。シーケンスはpaired-end（インサートの両端を読む）で101bpシークエンスしたものを事前にpre-processingしている。 これらのリードをArabidopsis thalianaのゲノムにマッピングする。TopHatを用いてsplice-awareなマッピングを行う。

Data

Input reads

(ファイルは、~/data/KY/tophat/　にある)

- condition Dark, rep#1:　2D_1_R1.fastq, 2D_1_R2.fastq
- condition Dark, rep#2:　2D_2_R1.fastq, 2D_2_R2.fastq
- condition Dark, rep#3: 2D_3_R1.fastq, 2D_3_R2.fastq
- condition Light, rep#1: 2D2L_1_R1.fastq, 2D2L_1_R2.fastq
- condition Light, rep#2: 2D2L_2_R1.fastq, 2D2L_2_R2.fastq
- condition Light, rep#3: 2D2L_3_R1.fastq, 2D2L_3_R2.fastq

Reference

- （本来ならば、Arabidopsis thaliana genome and annotation (Ensembl) をiGenomes (http://tophat.cbcb.umd.edu/igenomes.html) からダウンロードする　ftp://ussd-ftp.illumina.com/Arabidopsis_thaliana/Ensembl/TAIR10/Arabidopsis_thaliana_Ensembl_TAIR10.tar.gz）
- 今回は、そのままでは実習時間内では計算時間がかかり過ぎるので、今回は演習用にあらかじめChr4のみのデータに限定したgenomeファイル(genome_chr4.fa)とアノテーションファイル(genes_chr4.gtf)およびbowtie2のindexファイル(genome_chr4.fa.*.bt2)を用意してある。（KY/tophat/ディレクトリ）

Software

- tophat (installed)
- cufflinks (installed)
- bowtie2 (installed)
- samtools (installed)

Setup

Setup environment

top_cuff ディレクトリをつくり、以下の解析はその下で作業しよう。

```
$ mkdir top_cuff
$ cd top_cuff
```

Sequence reads

''less'' などのコマンドで、 2D_1_R1.fastq の内容を確認する。

注）Pre-processing済みであることが分かる。

Run tophat

TopHatを実行。

2D_1_R1.fastq

2D_1_R2.fastq

```
$ tophat -p 4 -G genes.gtf  -o 2D_1 genome.fa 2D_1_R1.fastq 2D_1_R2.fastq
```

*-p は使うCPU coreを指定するオプション。使用するコンピュータのスペックに合わせて。

- オススメ： --transcriptome-index　オプションは指定した方が良い。初回に作製したbowtie2 indexが2回目以降使い回せる。複数ライブラリを解析する際は大幅に時間の節約になる。
- 今回はpaired-endのデータを用いるが、single readでの解析もできる。

同様に他のもの計６サンプルをマッピング。

Inspect Results

計算が終わったら、どのようなファイルが生成されたか確認する。

```
$ ls -l C1_tophat_out/
```

```
$ $ ls -la 2D_1
total 92072
-rw-r--r--   1 kyamaguc  staff   3761633  3  2 17:14 accepted_hits.bam
-rw-r--r--   1 kyamaguc  staff       557  3  2 17:14 align_summary.txt
-rw-r--r--   1 kyamaguc  staff      5372  3  2 17:14 deletions.bed
-rw-r--r--   1 kyamaguc  staff      2850  3  2 17:14 insertions.bed
-rw-r--r--   1 kyamaguc  staff    407733  3  2 17:14 junctions.bed
drwxr-xr-x  30 kyamaguc  staff      1020  3  2 17:14 logs
-rw-r--r--   1 kyamaguc  staff       176  3  2 17:12 prep_reads.info
-rw-r--r--   1 kyamaguc  staff  33862783  3  2 17:14 unmapped.bam
```

prep_reads.infoやalign_summary.txtの中身を''less''で確認しよう。

accepted_hits.bam がアライメント結果だ。中身を''samtools''で確認しよう。

```
$ samtools view 2D_1/accepted_hits.bam |less
```

IGV

IGV で可視化しよう。

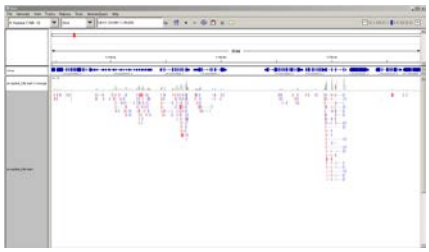
IGVでbamファイルを読むためには、インデクシングをしなければいけない。sort => indexing の段階をふむ。

```
$ samtools sort accepted_hits.bam accepted_hits.sorted
# => accepted_hits.sorted.bam ができる
$ samtools index accepted_hits.sorted.bam
# => accepted_hits.sorted.bam.bai ができる
```

1. IGVを立上げる。
2. 左上のプルダウンメニューからA.thaliana(TAIR10)を選ぶ。

3.メニュー　File > Load from File ... => accepted_hits.sorted.bam を選択

1. 第４染色体の適当な場所を指定し、適当にズームアップする。



X:1.14kb 近辺

Statistics

マップ率（インプットのリードの何%がリファレンスにマップされたか）を調べよう。

Run cufflinks

```
$ cufflinks -o 2D_1 -G genes.gtf accepted_hits.bam
```

- tophatと同じフォルダーに出力させておくのが良いだろう。

less コマンドで確認

```
-rw-r--r--   1 kyamaguc  staff   3761633  3  2 17:14 accepted_hits.bam
-rw-r--r--   1 kyamaguc  staff       557  3  2 17:14 align_summary.txt
-rw-r--r--   1 kyamaguc  staff      5372  3  2 17:14 deletions.bed
-rw-r--r--   1 kyamaguc  staff    422318  3  2 17:24 genes.fpkm_tracking
-rw-r--r--   1 kyamaguc  staff      2850  3  2 17:14 insertions.bed
-rw-r--r--   1 kyamaguc  staff    577476  3  2 17:24 isoforms.fpkm_tracking
-rw-r--r--   1 kyamaguc  staff    407733  3  2 17:14 junctions.bed
drwxr-xr-x  30 kyamaguc  staff      1020  3  2 17:14 logs
-rw-r--r--   1 kyamaguc  staff       176  3  2 17:12 prep_reads.info
-rw-r--r--   1 kyamaguc  staff         0  3  2 17:24 skipped.gtf
-rw-r--r--   1 kyamaguc  staff   8075446  3  2 17:24 transcripts.gtf
-rw-r--r--   1 kyamaguc  staff  33862783  3  2 17:14 unmapped.bam
```

新たにgenges.fpkm_tracking, isoforms.fpkm_trackingなどのファイルができている。

これらを他（2D_2, 2D_3, 2D2L_1, 2D2L_2, 2D2L_3）を含めて、全6sampleに関して行う。

Run cuffmerge

mergeするgtfファイルリストassemble.txtを作成する。以下を参考に自分のマシンに対応したパスで指定

```
~/top_cuff/2D_1/transcripts.gtf
~/top_cuff/2D_2/transcripts.gtf
~/top_cuff/2D_3/transcripts.gtf
~/top_cuff/2D2L_1/transcripts.gtf
~/top_cuff/2D2L_2/transcripts.gtf
~/top_cuff/2D2L_3/transcripts.gtf
```

cuffmergeを実行

```
$ cuffmerge -p 4 -s genome.fa -g genes.gtf assemblies.txt
```

*genome.fa, genes.gtfはパスを指定すること

merged_asmフォルダー下にmerged.gtfファイルが作成された。

less コマンドで確認

Run cuffdiff

GTFに記載の情報のみの解析なら、Run cufflinks, Run cuffmergeの部分はやる必要はない。

```
cuffdiff -p 4 merged.gtf -o 2D_vs_2D2L \
~/top_cuff/2D_1/accepted_hits.bam,~/top_cuff/2D_2/accepted_hits.bam,~/top_cuff/2D_3/accepted_hits.bam \
~/top_cuff/2D2L_1/accepted_hits.bam,~/top_cuff/2D2L_2/accepted_hits.bam,~/top_cuff/2D2L_3/accepted_hits.bam
```

2D_vs_2D2L ディレクトリに結果が出力されるので確認してみよう。

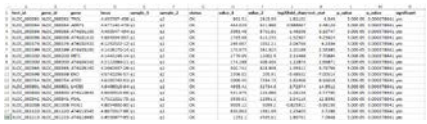
Explore the results

gene level での発現変動に興味があるので、見るべき結果ファイルは、

- gene_exp.diff
- genes.fpkm_tracking

tab区切りテキストなので、Excelに読み込ませることが可能。中身を確認しよう。 Excelのソート機能、フィルター機能を活用しよう。

Q: How many genes are differentially expressed?



Q: Scatter plot やMA plotを書いてみよう

Links

- http://tophat.cbcb.umd.edu/ |TopHat
- http://cufflinks.cbcb.umd.edu/ |CuffLinks

Notes

参考

Trapnell, C. et al. Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks. Nat Protoc 7, 562–578 (2012).

Case study 2: Transcript-based RNA-seq pipeline

de novo assembly and differential expression analysis

updated: September 5, 2015

Pipeline overview

transcript base のRNA-seq解析の基本的なパイプラインを学ぶ。イルミナMiSeqのショートリードを用いて、de novo RNA-seq アセンブリと得られたコンティグの簡易アノテーション、発現量推定と発現比較解析までのプロトコルを具体的なスクリプトやコマンドを追って説明する。

実験デザイン：シロイヌナズナ *Arabidopsis thaliana* を明条件(L)と暗条件(D)で育て、それぞれ3個体からRNAを抽出して (three biological replicates)、illumina TruSeq kitでRNA-seqライブラリを作製し、イルミナシーケンサーMiSeqでシーケンスした(片側76base シーケンス)。明暗 (L vs D) の条件の差で発現の異なる遺伝子を同定したい。

モデル植物シロイヌナズナはゲノムシーケンスは既知だが、ここではあえてゲノム情報は使わず、de novo RNA-seqのアプローチで解析する。

```
[Strategy]
1. de novo assembly to build transcriptome reference
   tool: Trinity

2. mapping reads to transcriptome reference
   tool: bowtie2

3. abundance estimation
   tool: eXpress

4. differential expression analysis
   tool: edgeR

5. annotation of reference sequences
   tool: blastx
```

Setup

Software

本解析に必要なソフトウェアは以下の通り。すべて演習用のMacにインストール済み。

- bowtie2, eXpress, edgeR, MS Excel, NCBI BLAST

Data set

```
~/data/EX/practice2/
|-- Data
|   |-- Trinity.fasta
|   |-- blastx_results.txt
|   `-- TAIR10_pep_20110103_representative_gene_model_updated
|
|-- IlluminaReads
|   |-- D1_R1.fastq
|   |-- D2_R1.fastq
|   |-- D3_R1.fastq
|   |-- L1_R1.fastq
|   |-- L2_R1.fastq
```

```
|   `-- L3_R1.fastq
|-- Scripts
|   |-- compile_results.rb
|   `-- merge_express_results.rb
```

de novo RNA-seq assembly using Trinity

Trinity でRNA-seq readsをde novo assembling.

(注：TrinityはLinux上でしか稼働しないので、本講習ではskipする。)

Input readsの準備

```
$cat *.fastq > left_all.fq
```

Run Trinity (example)

```
# prepare input reads
$ cat *.R1.fastq > left_all.fq
$ cat *.R2.fastq > right_all.fq

# Run Trinity
$ Trinity --seqType fq --left left_all.fq --right right_all.fq
          --CPU 8 --max_memory 20G
```

Result: ''Trinity.fasta''

Quality assessment

Trinityソフトウェアに含まれる `TrinityStats.pl` でassembly statisticsをチェック。

```
$TRINITY_HOME/util/TrinityStats.pl Trinity.fasta
```

Mapping Illumina Reads to Trinity contigs using bowtie2

bowtie2 を使ってリードをTrinity.fasta にマッピングする。

Build bowtie2 index

まず、reference (Trinity.fasta) をindexing。この作業は一度やればよい。

```
$ bowtie2-build Trinity.fasta Trinity.fasta
```

Mapping

```
$ bowtie2 -x Trinity.fasta -U IlluminaReads/D1_R1.fastq -p 8 -a -S D1.sam
```

bowtie2の実行が終わると、mapping rateなどのサマリーが表示されるので保存しておくとよいだろう。

(例)

```
382799 reads; of these:
  382799 (100.00%) were unpaired; of these:
    21064 (5.50%) aligned 0 times
    322103 (84.14%) aligned exactly 1 time
```

```
    39632 (10.35%) aligned >1 times
94.50% overall alignment rate
```

D1_R1.fastq D2_R1.fastq D3_R1.fastq L1_R1.fastq L2_R1.fastq L3_R1.fastq ６つのシーケンスファイルすべてについて、同様にマッピングを行なう。

```
$ bowtie2 -x Trinity.fasta -U IlluminaReads/D2_R1.fastq -p 8 -a -S D2.sam
$ bowtie2 -x Trinity.fasta -U IlluminaReads/D3_R1.fastq -p 8 -a -S D3.sam
$ bowtie2 -x Trinity.fasta -U IlluminaReads/L1_R1.fastq -p 8 -a -S L1.sam
$ bowtie2 -x Trinity.fasta -U IlluminaReads/L2_R1.fastq -p 8 -a -S L2.sam
$ bowtie2 -x Trinity.fasta -U IlluminaReads/L3_R1.fastq -p 8 -a -S L3.sam
```

samファイルの中身を確認する。

Abundance estimation using eXpress

eXpress はSAM/BAM fileを読み込んで、コンティグごとにリードをカウントする。multiple mapなどのマッピング結果のあいまいさを考慮して、真のカウントをEMアルゴリズムで推定する。

```
$ express -o L1 Trinity.fasta L1.sam
```

L1.sam　の解析結果が、L1 ディレクトリ以下に保存される。 `results.xprs` にカウント推定結果が出力されている。

多の5つのサンプルも同様に処理。

```
$ express -o L2 Trinity.fasta L2.sam
$ express -o L3 Trinity.fasta L3.sam
$ express -o D1 Trinity.fasta D1.sam
$ express -o D2 Trinity.fasta D2.sam
$ express -o D3 Trinity.fasta D3.sam
```

''results.xprs'' の中身を確認する。

Differential expression analysis using edgeR

Prepare count matrix

このあとの解析がしやすいように、サンプルごとに別のファイルに記録されているeXpressのカウントデータを、ひとつのファイルにまとめる。edgeRは、FPKMでなくカウントデータを入力としなければいけない。各、results.xprsファイルの、est_countsカラムを抜き出す。この作業にはやや煩雑なテキストデータ処理を要するので、筆者が用意したRubyスクリプト `merge_express_results.rb` を使って欲しい。

merge_express_results.rb

```
(使い方)
$ruby merge_express_result.rb dir1 dir2 dir3 ...

(例)
$ruby merge_express_result.rb D1 D2 D3 L1 L2 L3 > eXpress_est_count_merged.txt
```

Scatter plot

複雑な統計計算で発現変動解析をあれこれ行なう前に、scatter plotを描くなどの、簡単なデータチェックをしておく。

以下、R環境で。

```
> dat <- read.delim("eXpress_est_count_merged.txt", comment.char="#", row.name=1)

(example of scatter plot)
> plot(dat$D1 + 1, dat$D2+1, log="xy")

(example of all-vs-all scatter plot)
> pairs(dat, log="xy")

(example of comparison between D1vsD2 and D1vsL1)
> par(mfrow=c(1,2))
> plot(dat$D1 + 1, dat$D2+1, log="xy")
> plot(dat$D1 + 1, dat$L1+1, log="xy")
```

edgeR: data import

```
> library(edgeR)

> category <- c("D", "D", "D", "L", "L", "L")

> D <- DGEList(dat, group=group)           # import table into edgeR

> D <- calcNormFactors(D, method="TMM")    # TMM normalization

> D$samples
   group lib.size norm.factors
D1     D   361691    0.9436719
D2     D   311297    1.0367666
D3     D   410178    0.8524095
L1     L   455588    0.9706589
L2     L   378548    1.0408683
L3     L   349357    1.1868267
```

TMM normalization

```
# dump normalized count data
> D.cpm.tmm <- cpm(D, normalized.lib.size=T)
> write.table(D.cpm.tmm, file="cpm.tmm.txt", sep="\t", quote=F)
```

Differential expression analysis

```
>D <- estimateCommonDisp(D)               # estimate common dispersion
> D$common.dispersion
[1] 0.05574236

>D <- estimateTagwiseDisp(D)              # estimate tagwise dispersion
> summary(D$tagwise.dispersion)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.000871 0.000871 0.026900 0.059340 0.067680 1.029000

>de <- exactTest(D, pair=c("D", "L"))      # significance test to find differentially expressed genes
>topTags(de)                               # view the most significant genes


# dump DE analysis result
> de.sorted <- topTags(de, n=nrow(de$table))
> write.table(de.sorted$table, "de.txt", sep="\t", quote=F)
```

Result evaluation

有意に発現変動している遺伝子はいくつあるのか。例えば、Lで高発現し (logFC > 0)、FDR < 0.01 の遺伝子の数は、以下で求め

られる。

```
> sum(de.sorted$table$FDR<0.01 & de.sorted$table$logFC > 0)
```

これに答えるためには、edgeRの command `decideTestsDGE` も便利。

```
使い方例
> summary(decideTestsDGE(de, p.value=0.05))
   [,1]
-1    49
0  25903
1    269
```

plotSmear (edgeRに含まれるMA描画ツール) でMA plotを描いてみよう。

```
de.names <- row.names(D[decideTestsDGE(de, p.value=0.01) !=0, ])
plotSmear(D, de.tags=de.names)
```

MDS plotを行なうと、ライブラリ間の発現パターンの類似性をおおざっぱにとらえることができる。

```
plotMDS(D)
```

Quick annotation of Trinity contigs using BLAST

Trinityで得られたコンティグそれぞれがどのような遺伝子をコードしているだろうか？BLASTによる相同性検索はおおまかなアノテーションを行なうのに便利な手法である。ここでは、シロイヌナズナのタンパク質データベースを検索することにより、各コンティグがシロイヌナズナのどのタンパク質に対応するかを調べる。今回は、シロイヌナズナのシーケンスがqueryとなるので、シロイヌナズナのタンパク質データベースに対して検索をかけるとほぼ100%ヒットする。非モデル生物のde novo RNAseqでは、de novoアセンブリで得られたコンティグをqueryに、近縁種やモデル生物のタンパク質データベースや、nrデータベースに対して検索をかけることになる。

Build BLAST DB

国際コンソーシアムの運営するシロイヌナズナデータベース TAIRから、シロイヌナズナのタンパク質アミノ酸配列セットをダウンロードする。

ダウンロード

- ftp://ftp.arabidopsis.org/home/tair/Genes/TAIR10_genome_release/TAIR10_blastsets/TAIR10_pep_20110103_representative_gene_model_updated

（このファイルは、~/data/EX/practice2/Data/ ディレクトリにもコピーしておきました）。

ダウンロードしたファイルを''TAIR10.pep''の名前に変更。

```
$mv TAIR10_pep_20110103_representative_gene_model_updated TAIR10.pep
```

BLAST DBをビルド。

```
$ makeblastdb -in TAIR10.pep -dbtype prot -parse_seqids
```

（BLAST検索例）

```
$ blastx -query Trinity.fasta -db TAIR10.pep -num_threads 8 -max_target_seqs 1 \
       -evalue 1.0e-8 -outfmt 6 > blastx_results.txt
```

上の例では、トップヒットだけをテーブル形式で出力している。（参考のため結果ファイルをDataディレクトリに保存しておいた。）

Compile results

モデル生物の場合、大半の遺伝子に詳細なアノテーションがついているので、それらの情報と紐づけするとさらに利便性は上がる。シロイヌナズナの場合、各遺伝子のfunctional annotationは以下のファイルにまとめられており、TAIRのウェブサイトからダウンロードすることができる。

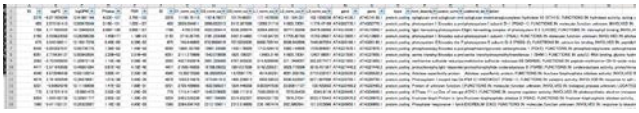
```
ftp://ftp.arabidopsis.org/home/tair/Genes/TAIR10_genome_release/TAIR10_functional_descriptions
```

これらの、DE analysis, annotation data, をひとつのテーブルにまとめると、見やすく、またさらなる下流解析を行なうのにも便利である。その処理には簡単なプログラミングが必要である。今回は、compile_results.rb (Scriptsディレクトリに含まれる) を使って下さい。

使い方

```
$ruby compile_results.rb > result_merged.txt
```

結果をMS Excelで吟味しよう。

(例) 

主なUNIXコマンド早見表

コマンド	由来等	説明	実行例
ls	LiSt	ディレクトリの内容をリスト表示	**ls** *directory*
pwd	Print Working Directory	現在のディレクトリを確認	**pwd**
cd	Change Directory	ディレクトリを移動	**cd** *directory*
cp	CoPy	ファイルコピー	**cp** *file1 file2*
mv	MoVe	ファイル移動	**mv** *file1 file2*
rm	ReMove	ファイル削除	**rm** *file*
chmod	Change MODe	ファイルの権限を変更	**chmod** *u+x file*
ln -s	LiNk	シンボリックリンクを作成	**ln -s** *file directory*
cat	ConcATinate	ファイルの内容を一括表示	**cat** *file*
head	HEAD	ファイルの先頭10行を表示	**head** *file*
tail	TAIL	ファイルの末尾10行を表示	**tail** *file*
less	antonym of more (*1)	ファイルの内容を表示	**less** *file*
wc	Word Count	ファイルの行数・単語数をカウント	**wc** *file*
grep	Global Regular Expression Print	ファイル内のパターン検索	**grep** *pattern file*
mkdir	MaKe DIRectory	ディレクトリを作成	**mkdir** *directory*
rmdir	ReMove DIRectory	ディレクトリを削除	**rmdir** *directory*
man	MANual	コマンドのマニュアルを表示	**man** *command*
echo	ECHO	引数の文字列を表示	**echo** *string*
jobs	JOBS	実行中のジョブをID付きで表示	**jobs**
fg	ForeGround	ジョブをフォアグラウンドで実行	**fg** *%jobid*
bg	BackGround	ジョブをバックグラウンドへ移行	**bg** *%jobid*
top	TOP	マシンの稼働状況をリアルタイム表示	**top**
ssh	Secure SHell	暗号化された通信経路で別のマシンにログイン	**ssh** *username@hostname*
scp	Secure CoPy	暗号化された通信経路で別のマシンにファイルをコピー	**scp** *file1 username@hostname:file2*
ftp	File Transfer Protocol	別のマシンにファイルを転送	**ftp** *hostname*
curl	Client for URLs	ウェブサーバからファイルを取得	**curl** *url*
gzip		ファイルの圧縮および解凍	**gzip** *file* **gzip** *-d file.gz*
bzip2		ファイルの圧縮および解凍	**bzip2** *file* **bzip2** *-d file.bz2*
tar	Tape ARchive	ファイルをまとめる	**tar** *cvf file.tar* **tar** *xvf file.tar*
vi	VIsual editor	テキストエディタ(vi)を起動	**vi**

*1 「more」というファイル表示コマンドも存在する。moreにはできない逆スクロールができるページャとして「less」と名付けられた。圧縮ファイルの表示なども可。