実習！ トランスオミクスのデータ解析（6日目課題集）

小鍛治　俊也

* 目的

オミクスデータとUNIXコマンドやBioinfomaticsなどの情報技術を駆使して、Unbiasedな代謝制御ネットワークを構築する。

* 前半：ネットワーク情報の抽出

以下の２つのファイルを作成する

１．変動代謝物とその代謝酵素（ここでは責任代謝酵素と呼ぶ）今日

変動代謝物A Aの代謝酵素S1

変動代謝物A Aの代謝酵素S2

変動代謝物B Bの代謝酵素P1

… …

２．リン酸化した責任代謝酵素とそのキナーゼ（ここでは責任キナーゼと呼ぶ）明日

リン酸化した代謝酵素S1 S1とP1のキナーゼX

リン酸化した代謝酵素P1 S1とP1のキナーゼX

リン酸化した代謝酵素P1 P1のキナーゼY

… …

* 後半：ネットワーク情報の可視化

前半で作成したファイルを基に以下のような図を作成する。



* 一般的な注意
* 個別ソフトウェアやUNIXコマンドの解説は、web上にわかりやすいものが多数存在するので、それらを参照されたい。
* 課題資料中のコマンド例はコピー＆ペーストしてよい
* ただし、**コマンド中の’-’（ハイフン）やスペース等が正しくコピーされず、コマンドが動作しないことがある。**その際は当該部分を自分で書き直すこと。
* UNIX課題１：UNIXコマンドの初歩を体験する
* 準備

フォルダ”自分の名前”内にフォルダ”Transomics”を作成

→実習ウェブサイトから各種ファイルをフォルダ”Transomics”にダウンロード＆解凍

* 1. ディレクトリに関する基本コマンドを体験する

Finderより、[アプリケーション]🡪[ユーティリティ]🡪ターミナル　を選択すると、UNIXシェルが起動します。以下のコマンドを順に実行せよ。

現在のフォルダを確認

$ pwd

/home/yugi

フォルダ内のファイル一覧を確認。先程作成したTransomics という名のフォルダができていることを確認。

$ ls

Transomics

フォルダ'Transomics'に移動する

$ cd Transomics

$ pwd

/home/yugi/Transomics

フォルダ’transomics\_data’に移動する。

$ cd transomicsdata

もう一度フォルダ内のファイル一覧を確認。さきほどダウンロードしたファイルを確認。

$ ls

Day6\_AnswerFiles ipi.RAT.xrefs metabolome.txt phosphoproteome.txt

元のフォルダに戻る（一つ上の階層のフォルダに行く）

$ cd ../

$ pwd

/home/yugi/Transomics

補完機能を使って、先程と同じことをする（フォルダ'Transomics'に移動する）。

$ cd transo（ここでtabキーを押す。すると、ディレクトリ名・ファイル名の残り部分を補ってくれる）

$ cd transomics\_data

履歴を使って、先程と同じコマンドを呼び出す

$ （ここで↑キーを押すと、1つ前に実行したコマンドを表示してくれる）

$ cd transomics\_data

止まらなくなってしまったプロセス（≒コマンド）を止めたいときは Ctrl キーを押しながら c を押す(略して Ctrl-c または ^C と書く)。

$ yes

Y

Y

Y

^C

四則演算もできる（exprコマンド）。数値と演算記号の間にスペースを入れること。

$ expr 1 + 1

2

$ expr 1+1

1+1 # 数値と演算記号の間にスペースを入れないとこういう結果になる。

$ expr 2 - 1

1

$ expr 2 \\* 2 # 掛け算の演算子は ’\\*’

4

$ expr 12 / 6

2

$ expr 11 % 3 # ’%’ は剰余演算子（割り算の余りを求める）

2

lessコマンドでタンパク質ID変換表 ipi.RAT.xrefs の中身を読んでみる。lessコマンドでファイルを読むときは、f(forward)で先に進み、b (back) で前に戻る。gでファイル先頭に移動、Gでファイル末尾に移動。q (quit) で終了。

$ less ipi.RAT.xrefs

catコマンドでファイルを表示する。

… …

… …

… …

REFSEQ\_INFERRED NP\_001178899 IPI00991262 F1MA97; （以下省略）

REFSEQ\_INFERRED NP\_001185509 IPI00991259 （以下省略）

headコマンドとtailコマンドでファイルの先頭と末尾を表示する。

$ head –n +2 ipi.RAT.xrefs

# Protein cross-references file for IPI rat release 3.87

SP A0A4Z3 IPI00197419 ENSRNOP00000007851 （以下省略）

$ tail –n -2 ipi.RAT.xrefs

REFSEQ\_INFERRED NP\_001178899 IPI00991262 F1MA97; （以下省略）

REFSEQ\_INFERRED NP\_001185509 IPI00991259 （以下省略）

cutコマンドでファイルの１列目だけ表示する。

$ cut -f1,2 ipi.RAT.xrefs

… …

… …

… …

REFSEQ\_INFERRED NP\_001178735

REFSEQ\_INFERRED NP\_001178880

REFSEQ\_INFERRED NP\_001178899

REFSEQ\_INFERRED NP\_001185509

ここまでに使った UNIX コマンドの使用法を調べてみる。下記は ls コマンドのマニュアル。

$ man ls

よく使うコマンド

* 既出

$ pwd # カレントディレクトリ（今いるディレクトリ）の表示

$ cd [dirname] # dirname という名前のディレクトリに移動

$ cd ../ # １つ上のディレクトリに移動

$ ls # そのディレクトリに入っているファイル一覧を表示

$ more [filename] # ファイルの中身を表示。スペースで進む、bで戻る(back)、qで終了(quit)。

* 基本

$ cd # ホームディレクトリに移動

$ cp fileA fileB # fileA という名前のファイルをfileB という名前で

コピー。

$ mv fileA fileB # fileA を fileB という名前に変更

$ ln -s fileA shortcutB # fileAのショートカット（UNIXでは「リンク」

と呼ぶ）を shortcutB という名前で作成

$ rm fileA # fileA を削除

$ cat fileA fileB ... fileZ # fileA ... fileZ を結合して表示。cat fileA は fileA のみを表示。

$ less fileA # fileAの中身を表示。スペースで進む。bで戻る。qでquit。

コマンド名とは裏腹に、moreコマンドの拡張版にあたる。

$ wc fileA # fileAの行数/単語数/バイト数を表示

$ head –n +5 fileA # fileAの先頭から5行を表示

$ tail –n -5 fileA # fileAの末尾から5行を表示

$ tail –n +5 fileA # fileAの5行目から末尾までを表示

$ diff fileA fileB # fileAとfileB の異なる部分（差分）を表示。

 fileAとfileB が同一の内容なら何も表示されない。

* UNIX課題2: パターンマッチ入門

メタボロームデータのファイル metabolome.txtについて、以下の演習問題に答えなさい。UNIXコマンドを組み合わせて解くこと。

1. metabolome.txtをExcelで開き、データの特徴を捉えなさい。右クリック→このアプリケーションで開く→Excelで開ける or DockのExcelのアイコンにDrag & Drop。

（1列目：インデックス、２列目：代謝物名、…、８列目以降：数値）

1. "glucose" を含む行を表示しなさい。grep コマンドを使うこと。
2. "Glucose"（先頭が大文字）を含む行を表示しなさい。
3. "glucose" を含まない行を表示しなさい。

（ヒント: grep の –v オプション。grep **–v** [検索したい文字列] [ファイル名] )

1. "glucose" または "Glucose" を含む行を表示しなさい。

（ヒント１: grep の -i オプション。大文字小文字を無視できれば良い。）

（ヒント２：複数の単語を検索したい場合は”Word1\|Word2”を指定する）

1. "glucose" および "Glucose" のどちらも含まない行を表示しなさい。

（ヒント: 一般に、複数のオプションを使いたい場合は次の例のように書けばよい。

 例: -aオプションと-bオプションを同時に使う → -abと書く）

1. インデックス１１３の代謝物を表示しなさい

（ヒント：grep ‘113’ metabolome.txtでは余計な行が含まれる。行先頭の”113”のみを検索したい場合は’^113’とする。^は行先頭を表す。ちなみに$は行末尾を表す。）

* UNIX課題3: パイプとリダイレクション、ファイル取り扱いの基本
* パイプとリダイレクション

メタボロームデータのファイル metabolome.txt について、以下の演習問題に答えなさい。UNIXコマンドを組み合わせて解くこと。

1. 以下のコマンドを実行しなさい。

$ grep CoA metabolome.txt > metabolome\_CoA.txt

（「Q1」のコマンドにおける '>' の意味はスライドを参照）

1. 「Q1」で作成したファイルの行数を数えなさい（ヒント：よく使うコマンド“wc”）。

~~ CoA (Acetyl CoAなど)の数がわかる

1. diffコマンドを以下のように用いて、「Q1」のファイルと模範解答ファイルが一致しているか、答え合わせしなさい。模範解答と一致していれば、何も出力されない。

$ diff metabolome\_CoA.txt Day6\_AnswerFiles/metabolome\_CoA.txt

1. 以下のコマンドを実行しなさい。

$ grep CoA metabolome.txt | wc

（「Q4」のコマンドにおける '|' の意味はスライド参照。）

* ファイル取り扱いの基本（2p前の“よく使うコマンドを参照”）
1. metabolome\_CoA.txt をコピーしたファイルを作成しなさい。コピーのファイル名は metabolome\_CoA\_copy.txt とする。コピーファイルの存在をlsで確認。
2. metabolome\_CoA\_copy.txt のファイル名を変更しなさい。変更後のファイル名は metabolome\_CoA\_cp.txt とする。ファイルの名前の変更をlsで確認。
3. metabolome\_CoA\_cp.txt を削除しなさい。ファイルの削除をlsで確認。
* UNIX課題4: awk

以下の演習問題に答えなさい。

1. 以下の (i) - (vii) は、awk コマンドを使って metabolome.txt を列ごとに処理した例である。それぞれのコマンドの説明として最も適当なものを下記の 1 ～ 7 から選びなさい。
2. $ awk '{ print }' metabolome.txt
3. $ awk '{ print $1 }' metabolome.txt

$ cut –f1 metabolome.txt

1. $ awk '{ print $2 }' metabolome.txt

$ awk -F"\t" '{ print $2 }' metabolome.txt

1. $ awk -F"\t" '{ print $4"\t"$2 }' metabolome.txt

$ cut –f4,2 metabolome.txt

1. $ awk -F"\t" '$2 ~ /CoA/{ print }' metabolome.txt
2. $ awk '$1 == “16”, $1 == “24” { print }' metabolome.txt
3. $ awk -F"\t" '($80 > 1){ print }' metabolome.txt
4. $ awk '$2 == “ATP” { print }' metabolome.txt
5. 4列目、2列目の順に表示する。出力形式は [4列目][タブ文字][2列目]。cutコマンドで同じことをしようとすると、出力が[2列目][タブ文字][4列目]になってしまう。
6. すべての行を表示する
7. 80列目の値が 1 より大きい行のみ表示する
8. 1列目を表示する。変数 $1 は1列目を指す。cutコマンドの-fオプションを用いても同じことができる。
9. 2列目を表示する。-F"\t" オプションを指定すると、タブ文字を列の区切りとして認識する。指定しないと区切り文字はタブ文字とスペース文字どちらも。
10. 2列目に文字列 "CoA" を含む行のみ表示する
11. 1列目が ”16” の行から、1列目が ”24” の行までの範囲を表示する。“”で囲まれていると文字列として認識される。
12. 2列目が文字列 "ATP" である行のみ表示する
* UNIX課題5: sortとuniq

リン酸化タンパク質データphosphoproteome.txtの各行は、リン酸化残基を持つペプチド（Not タンパク質）を表す。これについて、以下の演習問題に答えなさい。1, 2行目が見出し行であることに注意すること。

1. phosphoproteome.txtをExcelで開き、データの特徴を捉えなさい。

（1列目：酵素番号、２列目：タンパク質ID、…、９列目以降：数値）

1. このファイルの1-2行目は、各列の内容を説明する「見出し行」であってデータそのものではない。このため、1-2行目の存在がデータ処理の邪魔になることがある。1-2行目を表示せず、3行目以降表示したいときはどうすればよいか答えなさい。

（ヒント: head –n +N = 先頭N行、tail –n -N末尾N行、tail –n +N=先頭N行から末尾まで、head –n -N = 先頭から末尾N+1行まで）

1. phosphoproteome.txtの3行目以降の各行には、リン酸化ペプチドのIDと各時刻のリン酸化強度値が含まれている。これらにリン酸化された酵素は全部でいくつあるか数えなさい。

（ヒント：1~2行を削除した上で、cutまたはawkで１列目のEC番号を取り出し、全部で何行あるか数える。ただし、複数回登場するEC番号があるので、重複を排除したうえで行数をカウントすること。4つのパイプとsortとuniqを使う。）

1. それぞれの酵素について何個のリン酸化ペプチドが得られたか数えなさい。uniqコマンドの-cオプションを使うと容易である。
* 余談：メタボロームとは？

細胞や臓器、血液など生体試料中に含まれる代謝物質（糖やアミノ酸、脂質など）の総体、もしくはその網羅的計測データを指す。異なる条件下に置いた細胞から得たメタボロームデータを比較することにより、当該細胞の代謝調節メカニズムの解明が可能であるほか、薬効バイオマーカー探索などにも用いられている。有名なデータベースとしては、HMDB (Human Metabolome DataBase) や、本実習で用いるKEGG (Kyoto Encyclopedia of Genes and Genomes)がある。

* トランスオミクスStep (i): 変動代謝物を同定する

代謝物データmetabolome.txtから変動代謝物を抜き出し、 KEGG Mapper で利用可能な形式に変換する。

利用可能なファイルの例

C00001 red

C00002 blue

… …

Q1. 代謝物の変動を表す指標S1 / S0 (Fig. S3参照) はmetabolome.txt の 80 列目に記載されている。80列目だけ表示しなさい。

Q2. metabolome.txtの1-2行目は、各列の内容を説明する「見出し行」であってデータそのものではない。このため、1-2行目の存在がデータ処理の邪魔になることがある。1-2行目を表示せず、3行目以降表示したいときはどうすればよいか答えなさい。

Q3. 欠損点などの理由によりS1 / S0を計算できなかった代謝物については、当該箇所(80列目)に ”N.D.”と記載されている。“N.D." の代謝物の行を除いたデータを measured\_metabolite.txt に出力しなさい。1-2行目の見出しは出力しなくてよい。

（ヒント：S1 / S0 はmetabolome.txt の 80 列目に記載されている。ここが”N.D.”になっていない行を出力すればよい。awkでは == が一致を、 != が不一致を表す。”の有無に要注意。区切り文字としてタブ文字”\t”を指定する必要があるので注意。）

Q4. Q3 で作成したmeasured\_metabolite.txtの内容が、模範解答ファイル‘Day6\_AnswerFiles/Step\_i/measured\_metabolite.txt’と一致しているか確認しなさい。（ヒント: diffコマンドを使う。UNIX課題3のQ3参照。）

Q5. KEGGデータベース内にIDが存在しない代謝物については、KEGGのIDを記載する箇所(4列目)に ”-”と記載されている。4列目が “-" となっている代謝物をmeasured\_metabolite.txt から取り除いた結果残るデータを mappable\_metabolite.txt に出力しなさい。

Q6. Q5 で作成したmappable\_metabolite.txtの内容が、模範解答ファイル‘Day6\_AnswerFiles/Step\_i/mappable\_metabolite.txt’と一致しているか確認しなさい。

Q7. S1 / S0 > 1の代謝物は「増加」である。mappable\_metabolite.txtを元にして、[増加した代謝物 ID][タブ文字][“red”] という形式のファイル metabolite\_mapping.txt をつくりなさい。代謝物のIDは4列目に格納されている。また、”red”は綴りそのままの文字列である。（ヒント：タブ文字は”\t”で表現。”の有無に要注意）

Q8. Q7 で作成したmetabolite\_mapping.txtの内容が、模範解答ファイル‘Day6\_AnswerFiles/Step\_i/metabolite\_mapping**\_first**.txt’と一致しているか確認しなさい。

Q9. S1 / S0 < -1の代謝物は「減少」である。[減少した代謝物 ID][タブ文字][“blue”] という形式で、metabolite\_mapping.txtに追加書き込みしなさい。‘blue’は綴りそのままの文字列である。（ヒント：ファイルへの追加書き込みには > ではなく >> を用いる）

Q10. -1 < S1/S0 < 1の代謝物は「変動なし」である。[変動なしの代謝物 ID][tab][“green”] という形式で、metabolite\_mapping.txtに追加書き込みしなさい。

[ヒント：awkでは、複数の条件の論理積(AND)をとるときは条件A && 条件Bで表現]

Q11. Q7-Q10 で作成したmetabolite\_mapping.txtの内容が、模範解答ファイル‘Day6\_AnswerFiles/Step\_i/metabolite\_mapping**\_final**.txt’と一致しているか確認しなさい。

間違えている場合は、Q7からやり直し。履歴（↑）を使うと楽。

Q12. KEGG Mapper (http://www.genome.jp/kegg/tool/map\_pathway2.html) を開き、以下の手順で代謝経路図を表示しなさい。

1. metabolite\_mapping.txt の内容を KEGG Mapper の入力フォームにコピー＆ペーストする
2. KEGG Mapper にて、Search against 欄に Rattus novegicus を意味する 'rno' を入力（∵Fao細胞はラット肝がん由来）。
3. Use uncolored diagram をチェックして Exec ボタンを押す。
4. 'rno01100 Metabolic pathways - Rattus norvegicus (rat)' をクリックする
* UNIX課題6: 変数

以下の (i) - (vii) は、変数 i に数値を代入するコマンドと、変数 i の中身を表示するコマンド（あるいは表示しようとしたものの思い通りに動かなかったコマンド）の実行例である。それぞれのコマンドの説明として適当なものを下記の 1 ～ 7 から選びなさい。「変数」についての詳しい説明は「変数」「シェル」等で検索のこと。

1. $ i=1
2. $ i = 1

-bash: i: コマンドが見つかりません

1. $ echo $i

1

1. $ echo $ij

（何も出力されない）

$ echo ${i}j

1j

1. $ echo '${i}j'

${i}j

1. $ i=$(ls)

$ echo $i

Day6\_AnswerFiles\_2017 metabolome.txt mmc2.xlsx mmc4.xlsx phosphoproteome.txt posterior.txt rpk.txt

1. 変数名、代入記号、値の間にスペースを挟むとエラーになる
2. シングルクオートで囲まれた文字列中に含まれる変数名は、文字列としてそのまま出力される
3. 変数に値を代入するコマンドは [変数名]=[値] と書く
4. コマンドの実行結果そのものを変数として扱いたいときは、$( command ) のように書く
5. 変数名の直後に文字列を書く場合、変数名と当該文字列の区切りを明確にするため、変数名は { } で囲まなければならない。
6. 変数に格納されている値を呼び出す際には、$[変数名] と書く。echoは引数を表示するだけの関数、意外とよく使う。
* UNIX課題7: for
1. 以下のようにseqコマンドを使うと、整数を順に表示することを確認しなさい。

$ seq 10

1

2

3

4

5

6

7

8

9

10

1. 以下を実行し、1から10までの整数がforループごとにiに代入されることを確認せよ

$ for i in $(seq 10) ; do echo $i ; done

1

2

3

4

5

6

7

8

9

10

1. ２つ目のコマンドを埋めて、1から10までの整数を順に足し合わせるコマンドを完成させなさい。（ヒント：for文を使って、sumを更新しつつ表示する）

$ sum=0

$ for i in $(seq 10) ; do sum=$(expr $この空欄を埋める + $i) ; echo $sum ; done　←わからない人はこの部分をコピペ

1

3

6

10

15

21

28

36

45

55

1. metabolite\_mapping.txtと**for文**を用いて、増加した代謝物IDのみを黒色で表示するKEGG Mapper用テキストを表示せよ。

（ヒント：for文のinを工夫する。増加代謝物は何色であったか。echoにおいて\t（タブ文字）を用いる場合はオプション-eを）

$ for aCPD in $(この空欄を埋める | cut –f1) ; do echo -e "$aCPD\tblack"; done　←わからない人はこの部分をコピペ

C00026 (tab) black

C00041 black

C00074 black

…

C08262 black

C02990 black

C16169 black

* トランスオミクスStep (ii): 責任代謝酵素を同定する

「ある代謝物質を基質または産物とする酵素」のことをYugi et al. (2014) では「責任代謝酵素 (RME: Responsible Metabolic Enzyme)」と定義した。責任代謝酵素について以下の問いに答えなさい。

1. 多くのデータベースには、APIといわれるインターフェイスが存在する。以下のURLをブラウザ等に打ち込み、KEGGデータベースの「クエン酸 (KEGG COMPOUND ID: C00158) 」の項を閲覧せよ。代謝物データには名前・分子式・分子量・代謝経路、代謝酵素などの情報が含まれていることが確認できる。

URL: [http://rest.kegg.jp/get/cpd:C00158](http://rest.kegg.jp/get/cpd%3AC00158)

1. curl [URL] は、 URL にあるウェブページの内容を標準出力に書き出すコマンドである。以下のコマンドにより、KEGGデータベースにアクセスし、「クエン酸 (KEGG COMPOUND ID: C00158) 」のデータを取得し、データの内容を確認せよ。

$ curl http://rest.kegg.jp/get/cpd:C00158 > C00158.txt

$ less C00158.txt

1. KEGGデータベースの上記のAPI（get）以外にも様々なAPIを提供している。API“link”はある項目に対して関連のある項目を網羅的に出力する。以下のコマンドにより、KEGGデータベースにアクセスし、「クエン酸 (KEGG COMPOUND ID: C00158) 」に関わる全代謝酵素（ec.X.X.X.X）のデータを出力せよ。C00158.txtと比較すると、ENZYMEの項と同じことがわかる。

$ curl http://rest.kegg.jp/link/ec/cpd:C00158

cpd:C00158 ec:4.2.1.3

cpd:C00158 ec:4.1.3.6

cpd:C00158 ec:6.3.2.54

cpd:C00158 ec:6.3.2.38

cpd:C00158 ec:6.3.1.17

cpd:C00158 ec:2.3.3.3

cpd:C00158 ec:2.8.3.10

cpd:C00158 ec:2.3.3.8

cpd:C00158 ec:2.3.3.1

cpd:C00158 ec:2.3.3.16

cpd:C00158 ec:6.2.1.18

（順番は変わっている可能性あり）

1. では、変動代謝物に関わる代謝酵素の抜き出しを行う。まず、変動代謝物のデータをまとめたファイルmetab\_rme.txtを作成し、lessで確認せよ。metabolite\_mapping.txtから変動代謝物を抜き出し、それらをcurlを含むfor文にわたすことで作成可能である。最後にソートすること（Q1とUNIX課題7\_Q4参照）。

（ヒント：grepによる複数単語の検索はUNIX課題2\_Q5のヒント2参照。検索単語の除外を用いても良い。）

$ for aCPD in $(この空欄を埋める | cut –f1) ; do curl http://rest.kegg.jp/link/ec/cpd: この空欄を埋める; done | sort> metab\_rme.txt　←わからない人はこの部分をコピペ

1. （不安であれば）模範解答ファイル‘Day6\_AnswerFiles/Step\_ii/ metab\_rme.txt’と一致しているかを確認せよ。
2. metabolite\_rme.txtから代謝酵素のみを取り出し、重複をなくし、空白行を削除したファイルrme.txtを作成せよ。（ヒント：空白行の削除にはgrep –vを用いる。空白行は”^$”「行先頭^の次の文字が行末尾$」で表現）

$ この空欄を埋める | この空欄を埋める | この空欄を埋める | grep –v ‘^$’ > rme.txt　←わからない人はこの部分をコピペ

1. 作成したrme.txtの内容が、模範解答ファイル‘Day6\_AnswerFiles/Step\_ii/rme.txt’と一致しているか確認しなさい。（注：KEGGの更新により、微妙に一致しないこともある）
2. 黒色の責任代謝酵素を KEGG Mapper で視覚化するための元ファイル、rme\_mapping.txtを以下のコマンドで作成しなさい。echoコマンドに –e オプションを付けると” \t”はタブとして扱われる。（ヒント：file Aの中身を表示するコマンドはcat fileA）

$ for aEC in $(この空欄を埋める) ; do echo -e “この空欄を埋める" ; done > rme\_mapping.txt　←わからない人はこの部分をコピペ

1. metabolite\_mapping.txt と rme\_mapping.txt を一つのファイルにまとめ（fig2\_mapping.txt）、KEGG Mapper を使って変動代謝物と責任代謝酵素を同時に視覚化しなさい。（ヒント：ファイルの結合にはcat fileA fileB > fileCを用いる）