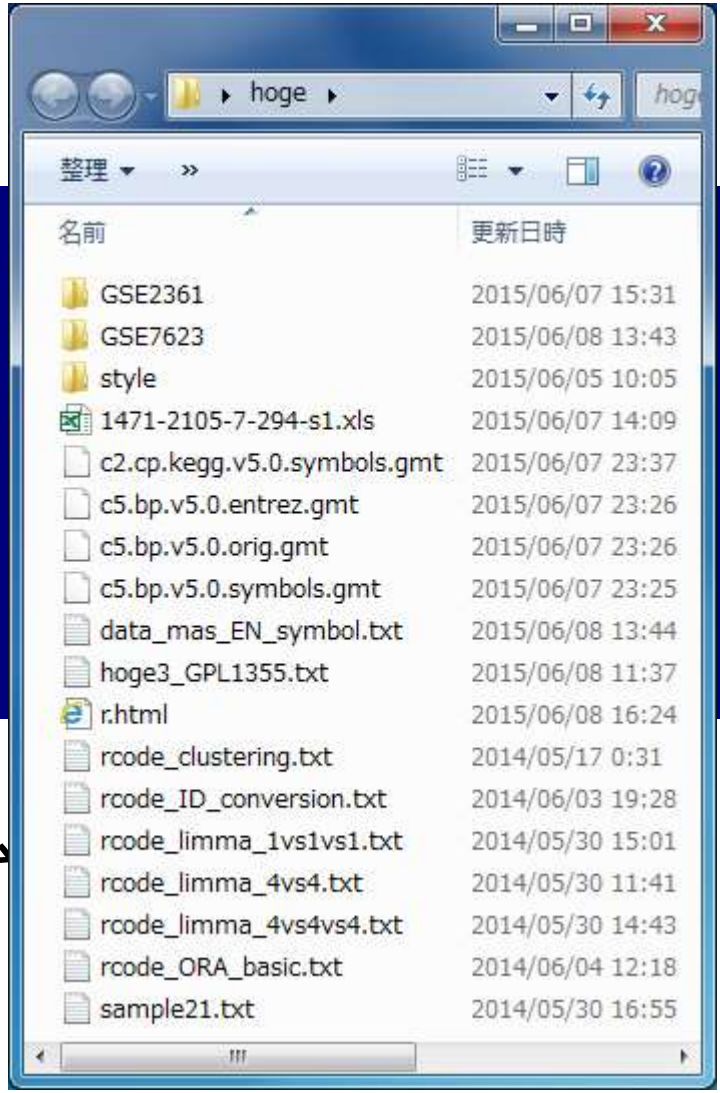
USBメモリ中のhogeフォルダをデスクトップにコピーしておいてください。



前回（5/26）のhogeフォルダがデスクトップに残っているかもしれないのでご注意ください。

機能ゲノム学 第4回

大学院農学生命科学研究科
アグリバイオインフォマティクス教育研究プログラム
門田幸二（かどた　こうじ）
kadota@iu.a.u-tokyo.ac.jp
http://www.iu.a.u-tokyo.ac.jp/~kadota/

講義予定

細胞中で発現している全転写物（トランスクリプトーム）の解析技術は、マイクロアレイから次世代シーケンサ（RNA-seq）に移行しつつあります。しかしRNA-seqデータ解析の多くは、マイクロアレイの知識を前提としています。本科目では、マイクロアレイデータを主な例として、各種トランスクリプトーム解析手法について解説します。

- 第1回（2015年5月12日）
  - 原理、各種データベース、生データ取得
  - 教科書の1.2節、2.2節周辺
- 第2回（2015年5月19日）
  - 遺伝子発現行列作成（データ正規化）
  - クラスタリング（データ変換や距離の定義など）
  - 教科書の3.2節周辺
- 第3回（2015年5月26日）
  - 実験デザイン、発現変動解析（多重比較問題）、M-A plot
  - 教科書の3.2節と4.2節周辺
- 第4回（2015年6月9日）
  - 機能解析（Gene Ontology解析やパスウェイ解析）

Contents

- デザイン行列の意味を理解（教科書p173–182）
  - limmaパッケージを用いた2群間比較のおさらい
  - limmaパッケージを用いた3群間比較（反復あり）
- 反復なし多群間比較（教科書p182–188）
  - limmaパッケージを用いた3群間比較（反復なし）
  - TCCパッケージ中のROKU法を用いた特異的発現遺伝子検出
- 機能解析（遺伝子セット解析）
  - 基本的な考え方
  - 前処理
    - MSigDBからの遺伝子セット情報（gmt形式ファイル）取得
    - ID変換（probe ID → gene symbol）
  - GSAパッケージを用いた遺伝子セット解析

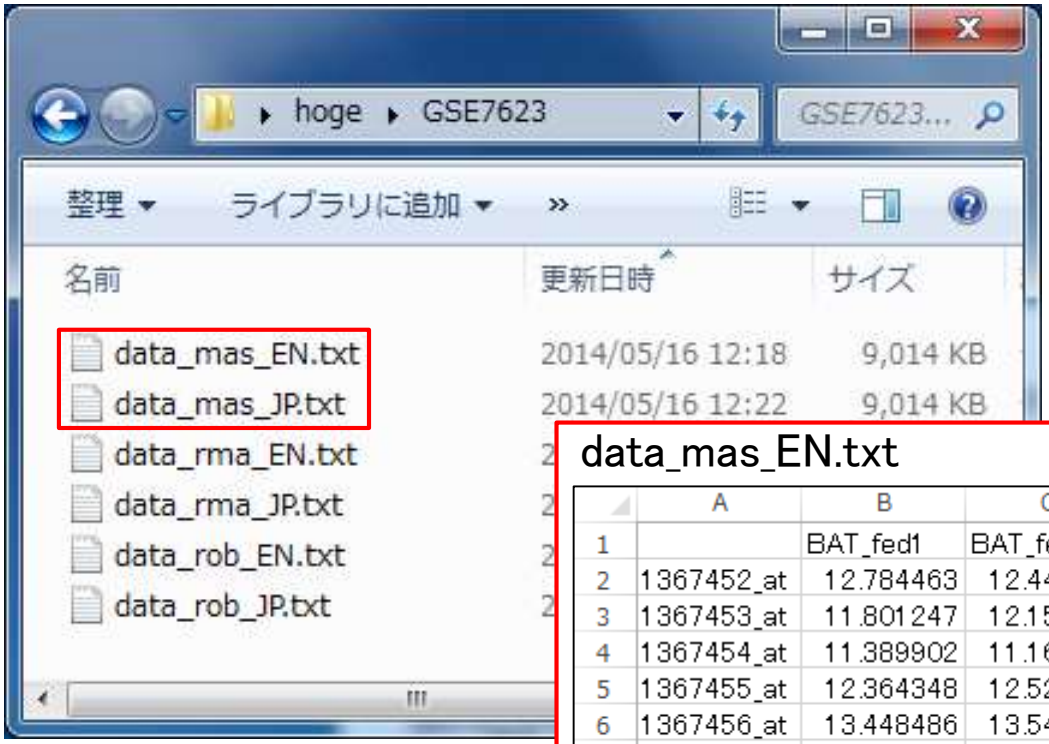
アレイデータ

hogeフォルダ中に3つの前処理法の実行結果ファイルがあります。
MAS5 (data_mas_*.txt)、RMA (data_rma_*.txt)、RMX (data_rob_*.txt)

- Affymetrix GeneChip
  - Ge et al., *Genomics*, **86**: 127–141, 2005
    - **GSE2361**、GPL96 (Affymetrix Human Genome U133A Array)、22,283 probesets
    - **ヒト36サンプル**：Heart (心臓)、Thymus (胸腺)、Spleen (脾臓)、Ovary (卵巣)、Kidney (腎臓)、Skeletal Muscle (骨格筋)、Pancreas (膵臓)、Prostate (前立腺)、…
  - Nakai et al., *Biosci Biotechnol Biochem.*, **72**: 139–148, 2008
    - **GSE7623**、GPL1355 (Affymetrix Rat Genome 230 2.0 Array)、31,099 probesets
    - **ラット24サンプル**：Brown adipose tissue (褐色脂肪組織; BAT)8サンプル、White adipose tissue (白色脂肪組織; WAT)8サンプル、Liver (肝臓; LIV)8サンプル
      - BAT 8サンプル：通常(BAT_fed) 4サンプル vs. 24時間絶食(BAT_fas) 4サンプル
      - WAT 8サンプル：通常(WAT_fed) 4サンプル vs. 24時間絶食(WAT_fas) 4サンプル
      - LIV 8サンプル：通常(LIV_fed) 4サンプル vs. 24時間絶食(LIV_fas) 4サンプル
  - Kamei et al., *PLoS One*, **8**: e65732, 2013
    - **GSE30533**、GPL1355 (Affymetrix Rat Genome 230 2.0 Array)、31,099 probesets
    - **ラット10サンプル**：全てLiver (肝臓)サンプル
    - iron-deficient diet (Iron_def) 5サンプル vs. control diet (Control) 5サンプル

アレイデータ

GSE7623 (Nakai et al., 2008)
の対数変換後のデータ



data_mas_EN.txt

	A	B	C	D	E	F	G	H	I
1		BAT_fed1	BAT_fed2	BAT_fed3	BAT_fed4	BAT_fas1	BAT_fas2	BAT_fas3	BAT_fa
2	1367452_at	12.784463	12.447082	12.805908	12.304718	12.589425	12.607532	11.815378	12.43
3	1367453_at	11.801247	12.152935	11.942227	11.968477	11.845375	11.681727	12.078672	12.04
4	1367454_at	11.389902	11.160757	11.145987	11.212088	11.540652	11.308877	11.49885	11.40
5	1367455_at	12.364348	12.529744	12.432574	12.604011	12.441991	12.249935	12.281827	12.19
6	1367456_at	13.448486	13.543046	13.552794	13.629799	13.36913	13.244278	13.424371	13.32
7	1367457_at	10.404028	10.69632	10.475078	10.45579	10.141921	10.290666	10.146529	10.2
8	1367458_at	9.9253387	10.244544	9.9720008	9.9576072	8.702884	9.3578792	9.2134367	9.499

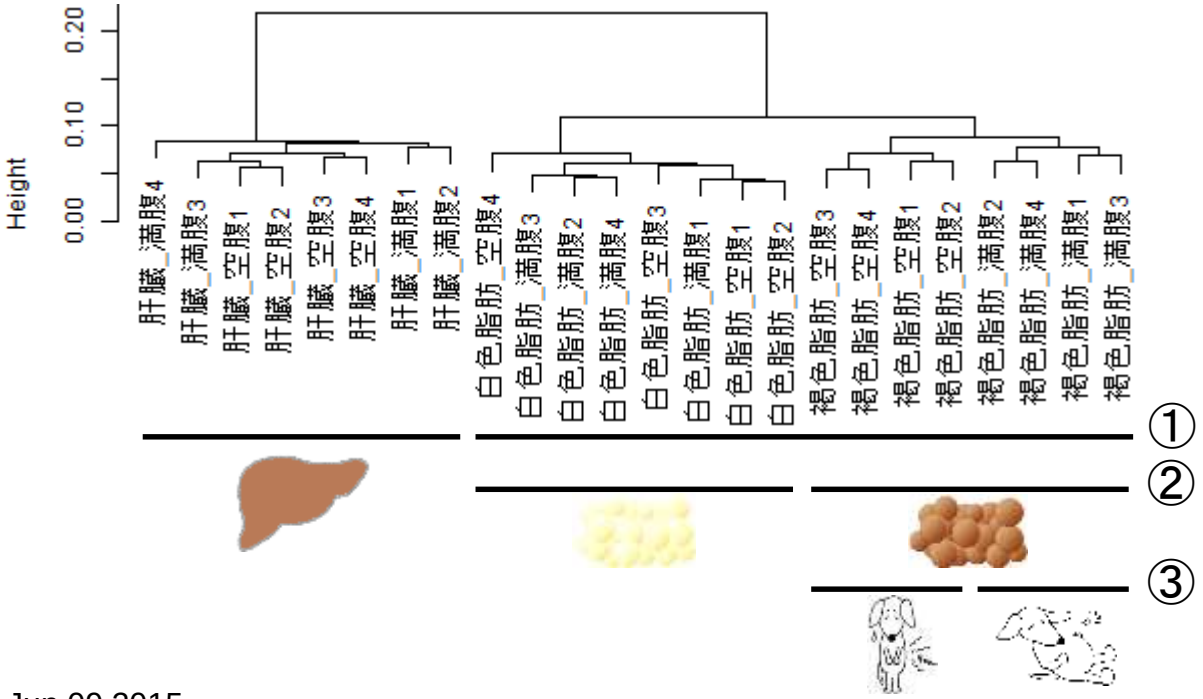
data_mas_JP.txt

	A	B	C	D	E	F	G
1		褐色脂肪_満腹1	褐色脂肪_満腹2	褐色脂肪_満腹3	褐色脂肪_満腹4	褐色脂肪_空腹1	褐色脂肪_空腹
2	1367452_at	12.7844634	12.44708219	12.80590758	12.30471769	12.58942538	12.6075319
3	1367453_at	11.80124704	12.15293493	11.94222741	11.96847729	11.84537542	11.6817274
4	1367454_at	11.38990178	11.16075717	11.14598707	11.21208786	11.54065185	11.3088766
5	1367455_at	12.36434768	12.52974368	12.43257392	12.60401124	12.44199125	12.2499348
6	1367456_at	13.44848649	13.54304603	13.55279359	13.62979898	13.36912977	13.2442783
7	1367457_at	10.40402803	10.69631952	10.47507777	10.4557902	10.14192076	10.2906657
8	1367458_at	9.9253387	10.244544	9.9720008	9.9576072	8.702884	9.3578792

2群間比較(limma)

GSE7623データを用い、様々な2群間比較を行い、クラスタリング結果とDEG検出結果の関係をみてみよう

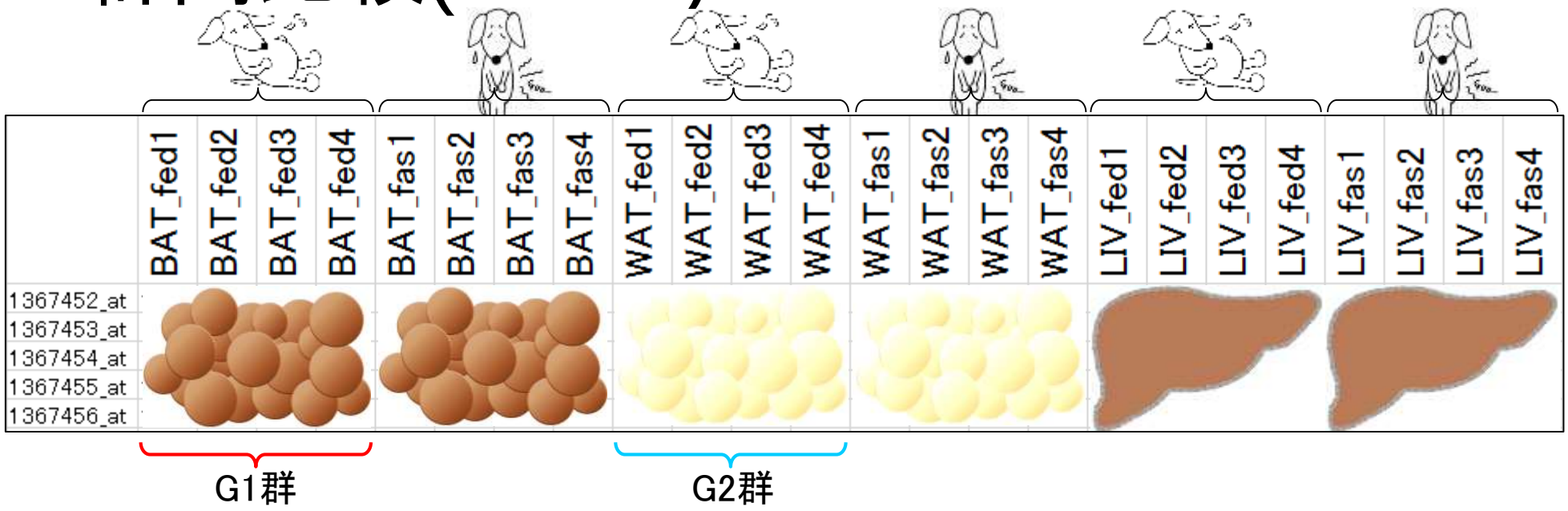
- Nakai et al., *Biosci Biotechnol Biochem.*, **72**: 139–148, 2008
  - GSE7623、GPL1355 (Affymetrix Rat Genome 230 2.0 Array)、31,099 probesets
  - ラット24サンプル：Brown adipose tissue (褐色脂肪組織; BAT) 8サンプル、White adipose tissue (白色脂肪組織; WAT) 8サンプル、Liver (肝臓; LIV) 8サンプル
    - BAT 8サンプル：通常 (BAT_fed) 4サンプル vs. 24時間絶食 (BAT_fas) 4サンプル
    - WAT 8サンプル：通常 (WAT_fed) 4サンプル vs. 24時間絶食 (WAT_fas) 4サンプル
    - LIV 8サンプル：通常 (LIV_fed) 4サンプル vs. 24時間絶食 (LIV_fas) 4サンプル

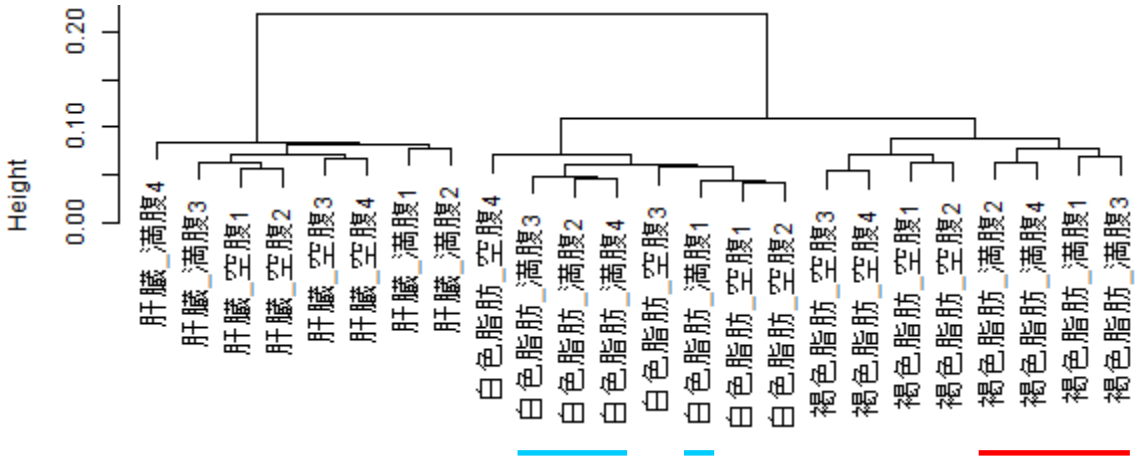


rcode_clustering_png.txtの実行結果。
①肝臓と脂肪間で大きく二つのクラスターに分かれている。
②脂肪の中でも白色脂肪と褐色脂肪に分かれている。
③褐色脂肪は空腹(24時間絶食)と満腹(通常)できれいに分かれている。

4 WAT_fed vs. 4 BAT_fed

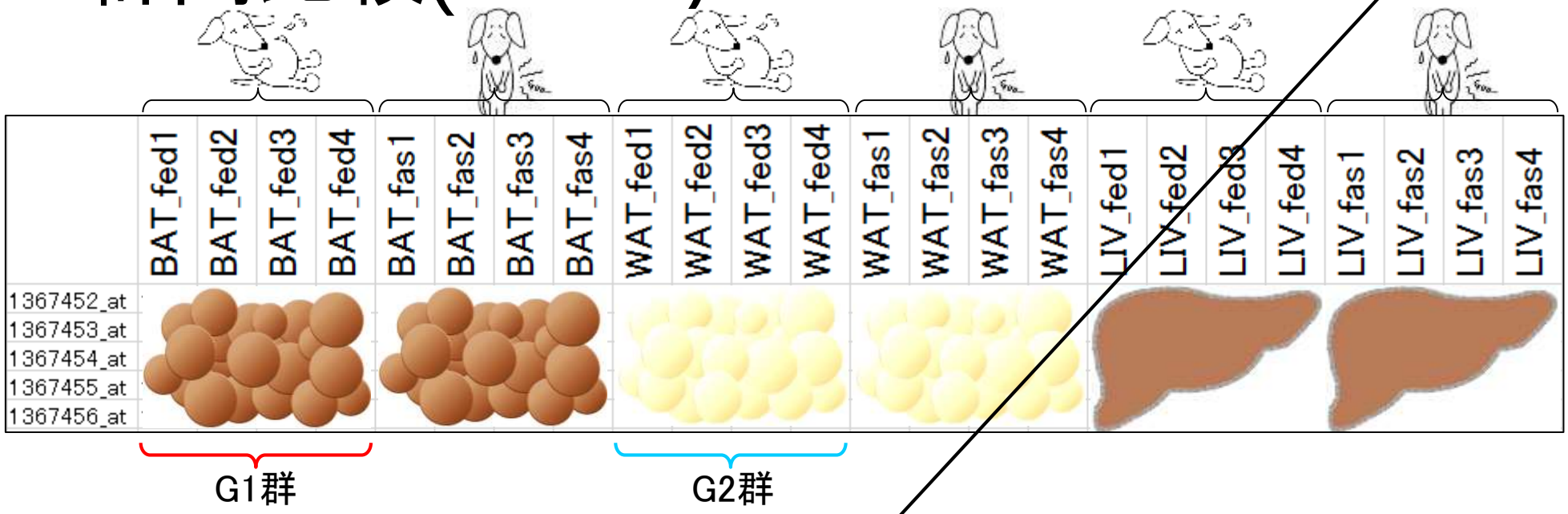
2群間比較(limma)





2群間比較(limma)

サブセット抽出のための情報はここで与えている



rcode_limma_4vs4.txt

```
#####################################################↓
###  4 BAT_fed samples vs. 4 WAT_fed samples  ###↓
#####################################################↓
in_f <- "data_mas_EN.txt"                  #入力ファイル名を指定してin_fに格納↓
out_f1 <- "hoge1.txt"                      #出力ファイル名を指定してout_f1に格納↓
out_f2 <- "hoge1.png"                      #出力ファイル名を指定してout_f2に格納↓
param_G1 <- 4                              #G1群のサンプル数を指定↓
param_G2 <- 4                              #G2群のサンプル数を指定↓
param_posi <- c(1:4, 9:12)                 #元の発現行列上での列番号を指定↓
param_FDR <- 0.05                          #DEG検出時のfalse discovery rate (FDR)閾値を指定↓
param_fig <- c(400, 380)                   #ファイル出力時の横幅と縦幅を指定(単位はピクセル)
↓
#必要なパッケージをロード↓
library(limma)                             #パッケージの読み込み↓
```

2群間比較(limma)

解析したいサブセットに正しくできていることがわかります

rcode_limma_4vs4.txt

```
##################################################
###  4 BAT_fed samples vs. 4 WAT_fed samples  ###
##################################################
in_f <- "data_mas_EN.txt"
out_f1 <- "hoge1.txt"
out_f2 <- "hoge1.png"
param_G1 <- 4
param_G2 <- 4
param_posi <- c(1:4, 9:12)
param_FDR <- 0.05
param_fig <- c(400, 380)

#必要なパッケージをロード
library(limma)

#入力ファイルの読み込みとラベル情報の作
data <- read.table(in_f, header=TRUE, r
data.cl <- c(rep(1, param_G1), rep(2, p
data <- data[,param_posi]
colnames(data)

#本番
#design <- model.matrix(~ as.factor(dat
design <- model.matrix(~data.cl)
fit <- lmFit(data, design)
```

R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/GSE7623"
> in_f <- "data_mas_EN.txt"                    #入力ファイ$
> out_f1 <- "hoge1.txt"                        #出力ファイ$
> out_f2 <- "hoge1.png"                        #出力ファイ$
> param_G1 <- 4                                #G1群のサン$
> param_G2 <- 4                                #G2群のサン$
> param_posi <- c(1:4, 9:12)                   #元の発現行$
> param_FDR <- 0.05                            #DEG検出時の$
> param_fig <- c(400, 380)                     #ファイル出$
>
> #必要なパッケージをロード
> library(limma)                               #パッケージ$
>
> #入力ファイルの読み込みとラベル情報の作成、そしてサ$
> data <- read.table(in_f, header=TRUE, row.names=1, $
> data.cl <- c(rep(1, param_G1), rep(2, param_G2))#G1$
> data <- data[,param_posi]                     #サブセット$
> colnames(data)                               #サブセット$
[1] "BAT_fed1" "BAT_fed2" "BAT_fed3" "BAT_fed4"
[5] "WAT_fed1" "WAT_fed2" "WAT_fed3" "WAT_fed4"
> param_posi
[1]  1  2  3  4  9 10 11 12
>
```

2群間比較(limma)

designオブジェクトが(実験)デザイン行列です。この行列の2列目がG1群とG2群がどれに相当するかを表すクラスラベル情報であることもわかります。

rcode_limma_4vs4.txt

```
#本番
#design <- model.matrix(~ as.factor(data.cl))#デザイン行列を作成した結果をde
design <- model.matrix(~data.cl)          #デザイン行列を作成した結果をdesignに
fit <- lmFit(data, design)                #モデル構築(ばらつきの程度を見積もっている)
out <- eBayes(fit)                        #検定(経験ベイズ)
p.value <- out$p.value[,ncol(design)]  #p
q.value <- p.adjust(p.value, method="BH")
ranking <- rank(p.value)                  #p
sum(q.value < param_FDR)                  #F
mean_G1 <- apply(as.matrix(data[,data.cl=
mean_G2 <- apply(as.matrix(data[,data.cl=
M <- mean_G2 - mean_G1                    #M
A <- (mean_G1 + mean_G2)/2                #M

#ファイルに保存(テキストファイル)
tmp <- cbind(rownames(data), data, M, A,
write.table(tmp, out_f1, sep="¥t", append
```

R Console

```
> design <- model.matrix(~data.cl)
> design
  (Intercept) data.cl
1           1       1
2           1       1
3           1       1
4           1       1
5           1       2
6           1       2
7           1       2
8           1       2
attr(,"assign")
[1] 0 1
> data.cl
[1] 1 1 1 1 2 2 2 2
> as.factor(data.cl)
[1] 1 1 1 1 2 2 2 2
Levels: 1 2
>
```

2群間比較(limma)

①limma実行後のp-value情報は、ベクトル形式ではなく行列形式になっていることに注意。そしてその列数は、デザイン行列designの列数と同じ。②out$p.value行列の2列目の情報が解析結果に相当

rcode_limma_4vs4.txt

```
#本番
#design <- model.matrix(~ as.factor(data.cl))#デザイン行列を作成
design <- model.matrix(~data.cl)          #デザイン行列を作成した結果をdesignに格納
fit <- lmFit(data, design)                #モデル構築(ばらつきの程度を見積もっている)
out <- eBayes(fit)                        #検定(経験ベイズ)
p.value <- out$p.value[,ncol(design)]     #p値をp.valueに格納
q.value <- p.adjust(p.value, method="BH")#q値をq.valueに格納
ranking <- rank(p.value)                  #p.valueでランキングした結果をrankingに格納
sum(q.value < param_FDR)                  #FI
mean_G1 <- apply(as.matrix(data[,data.cl=
mean_G2 <- apply(as.matrix(data[,data.cl=
M <- mean_G2 - mean_G1                    #M
A <- (mean_G1 + mean_G2)/2                #M

#ファイルに保存(テキストファイル)
tmp <- cbind(rownames(data), data, M, A,
write.table(tmp, out_f1, sep="¥t", append
```

R Console

```
> fit <- lmFit(data, design)                #モデル構築($
> out <- eBayes(fit)                        #検定(経験ベ$
> p.value <- out$p.value[,ncol(design)]     #p値をp.valu$
> head(out$p.value, n=5)
             (Intercept)     data.cl ①
1367452_at 9.548174e-11 0.60594552
1367453_at 3.516879e-11 0.09645379
1367454_at 1.402195e-10 0.16185158
1367455_at 2.986084e-11 0.37498101
1367456_at 7.686525e-12 0.10585764
> dim(out$p.value)
[1] 31099     2
> head(out$p.value[, 2], n=4)
②
1367452_at 1367453_at 1367454_at 1367455_at
0.60594552 0.09645379 0.16185158 0.37498101
>
```

2群間比較(limma)

①limma実行後のp-value情報は、ベクトル形式ではなく行列形式になっていることに注意。そしてその列数は、デザイン行列designの列数と同じ。しつこく示してるだけ

rcode_limma_4

```
#本番
#design <- model.matrix(~ as.factor(data.cl))#デザイン行列を作成した結果をdesignに格納
design <- model.matrix(~data.cl)          #デザイン行列を作成した結果をdesignに格納
fit <- lmFit(data, design)                #モデル構築(ばらつきの程度を見積もっている)
out <- eBayes(fit)                        #
p.value <- out$p.value[,ncol(design)]     #p
q.value <- p.adjust(p.value, method="BH")
ranking <- rank(p.value)                  #p
sum(q.value < param_FDR)                  #F
mean_G1 <- apply(as.matrix(data[,data.cl=
mean_G2 <- apply(as.matrix(data[,data.cl=
M <- mean_G2 - mean_G1                    #M
A <- (mean_G1 + mean_G2)/2                #M

#ファイルに保存(テキストファイル)
tmp <- cbind(rownames(data), data, M, A,
write.table(tmp, out_f1, sep="¥t", append
```

R Console

```
> head(out$p.value[, 2], n=4)
1367452_at 1367453_at 1367454_at 1367455_at
0.60594552 0.09645379 0.16185158 0.37498101
> design
  (Intercept) data.cl
1           1       1
2           1       1
3           1       1
4           1       1
5           1       2
6           1       2
7           1       2
8           1       2
attr(,"assign")
[1] 0 1
> dim(design)
[1] 8 2
> ncol(design)
[1] 2
>
```

Contents

- デザイン行列の意味を理解（教科書p173–182）
  - limmaパッケージを用いた2群間比較のおさらい
  - limmaパッケージを用いた3群間比較（反復あり）
- 反復なし多群間比較（教科書p182–188）
  - limmaパッケージを用いた3群間比較（反復なし）
  - TCCパッケージ中のROKU法を用いた特異的発現遺伝子検出
- 機能解析（遺伝子セット解析）
  - 基本的な考え方
  - 前処理
    - MSigDBからの遺伝子セット情報（gmt形式ファイル）取得
    - ID変換（probe ID → gene symbol）
  - GSAパッケージを用いた遺伝子セット解析

3群間比較(limma)

赤枠内では、分散分析(ANOVA)的な「比較するどこかの群間で発現変動している遺伝子」の同定について記述しています。「機能ゲノム学」では①の教科書に沿った内容で進めます。実験デザイン行列だとかコントラスト行列だとか様々な記述形式がありますが、②「応用1」の例題を一通り行うと感覚がつかめると思います。「農学生命情報科学特論I」のRNA-seqカウントデータの統計解析のところで説明する予定です。

(Rで)マイクロアレイデータ解析

(last modified 2015/05/25, since 2005)

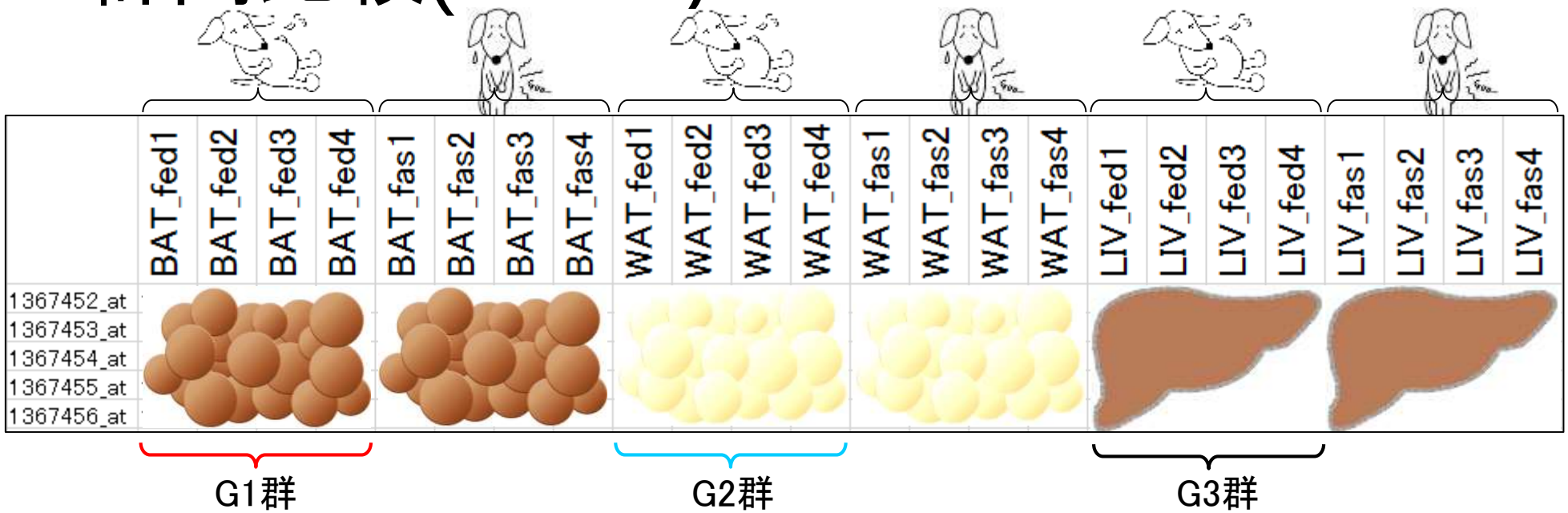
What's new?

- 門田幸二 著シリーズ Useful R 第7巻トランスクリプトーム解析刊行(共立出版)。マイクロアレイ解
  する最近の知見や、ROKU法 (Kadota et al., 2006)、WAD法 (Kadota et al., 2008)などについての解
  んで
  トー
- お知
  や講

- 解析 | 発現変動 | 2群間 | 対応なし | パターンマッチング法(last modified 20
- 解析 | 発現変動 | 2群間 | 対応あり | について(last modified 2009/11/11)
- 解析 | 発現変動 | 2群間 | 対応あり | SAM (Tusher 2001) (last modified 201
- 解析 | 発現変動 | 2群間 | 対応あり | SAM (Tusher 2001)+WADの重みづけ
- 解析 | 発現変動 | 2群間 | 対応あり | 時系列 | について(last modified 2013/6/8)
- 解析 | 発現変動 | 2群間 | 対応あり | 時系列 | maSigPro (Conesa 2006) (last modified 2013/6/2)
- 解析 | 発現変動 | 3群間 | 対応なし | について(last modified 2015/01/16)
- 解析 | 発現変動 | 3群間 | 対応なし | Mulcom (Isella 2011)(last modified 2013/12/06)
- 解析 | 発現変動 | 3群間 | 対応なし | 基礎 | limma (Ritchie 2015)(last modified 2015/06/08) NEW
- 解析 | 発現変動 | 3群間 | 対応なし | 応用1 | limma (Ritchie 2015)(last modified 2015/06/08) NEW
- 解析 | 発現変動 | 3群間 | 対応なし | 応用2 | limma (Smyth 2004)(last modified 2015/06/08) NEW
- 解析 | 発現変動 | 3群間 | 対応なし | 一元配置分散分析(One-way ANOVA)(last modified 2013/11/12)
- 解析 | 発現変動 | 3群間 | 対応なし | Kruskal-Wallis(クラスカル-ウォリス)検定 (last modified 2013/6/2)
- 解析 | 発現変動 | 多群間 | について(last modified 2013/6/2)
- 解析 | 発現変動 | 多群間 | SpeCond(Cavalli 2011) (last modified 2013/6/10)
- 解析 | 発現変動 | 多群間 | ROKU(Kadota 2006) (last modified 2014/05/30)

デザイン行列を理解しておかないと、多群間比較時に困ります

3群間比較(limma)



rcode_limma_4vs4vs4.txt

```
##########################################################################↓
###  4 BAT_fed samples vs. 4 WAT_fed samples vs. 4 LIV_fed samples ###↓
##########################################################################↓
in_f <- "data_mas_EN.txt"                  #入力ファイル名を指定してin_fに格納↓
out_f1 <- "hoge1.txt"                      #出力ファイル名を指定してout_f1に格納↓
out_f2 <- "hoge1.png"                      #出力ファイル名を指定してout_f2に格納↓
param_G1 <- 4                              #G1群のサンプル数を指定↓
param_G2 <- 4                              #G2群のサンプル数を指定↓
param_G3 <- 4                              #G2群のサンプル数を指定↓
param_posi <- c(1:4, 9:12, 17:20)          #元の発現行列上での列番号を指定↓
param_FDR <- 0.05                          #DEG検出時のfalse discovery rate (FDR)閾値を指定↓
↓
#必要なパッケージをロード↓
library(limma)                             #パッケージの読み込み↓
```

 教科書p180-182

3群間比較(limma)

解析したいサブセットに正しくできていることがわかります

rcode_limma_4vs4vs4.txt

```
################################################################################
###  4 BAT_fed samples vs. 4 WAT_fed samples vs. 4 LIV_fed samples ###
################################################################################

in_f <- "data_mas_EN.txt"                          #入力
out_f1 <- "hoge1.txt"                              #出力
out_f2 <- "hoge1.png"                              #出力
param_G1 <- 4                                      #G1群
param_G2 <- 4                                      #G2群
param_G3 <- 4                                      #G2群
param_posi <- c(1:4, 9:12, 17:20)                  #元の
param_FDR <- 0.05                                  #DEG

#必要なパッケージをロード
library(limma)                                     #パッ

#入力ファイルの読み込みとラベル情報の作成、
data <- read.table(in_f, header=TRUE, row.na
data.cl <- c(rep("G1", param_G1), rep("G2",
data <- data[,param_posi]                          #サブ
colnames(data)                                     #サブ

#本番
#design <- model.matrix(~ 0 + as.factor(data
design <- model.matrix(~ 0 + data.cl)  #デザ
colnames(design) <- levels(as.factor(data.c
```

R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/GSE7623"
> list.files(pattern="mas_EN")
[1] "data_mas_EN.txt"
> in_f <- "data_mas_EN.txt"                        #入力ファイ$
> out_f1 <- "hoge1.txt"                            #出力ファイ$
> param_G1 <- 4                                    #G1群のサン$
> param_G2 <- 4                                    #G2群のサン$
> param_G3 <- 4                                    #G2群のサン$
> param_posi <- c(1:4, 9:12, 17:20)                #元の発現行$
> param_FDR <- 0.05                                #DEG検出時の$
>
> #必要なパッケージをロード
> library(limma)                                   #パッケージ$
>
> #入力ファイルの読み込みとラベル情報の作成、そしてサ$
> data <- read.table(in_f, header=TRUE, row.names=1, $
> data.cl <- c(rep("G1", param_G1), rep("G2", param_G$
> data <- data[,param_posi]                        #サブセット$
> colnames(data)                                   #サブセット$
 [1] "BAT_fed1" "BAT_fed2" "BAT_fed3" "BAT_fed4"
 [5] "WAT_fed1" "WAT_fed2" "WAT_fed3" "WAT_fed4"
 [9] "LIV_fed1" "LIV_fed2" "LIV_fed3" "LIV_fed4"
>
```

3群間比較(limma)

実験デザイン行列には様々が記述の仕方があって難解ですが、慣れ以外にはありません。

```
#本番↓
#design <- model.matrix(~ 0 + as.factor(data
design <- model.matrix(~ 0 + data.cl)  #デサ
colnames(design) <- levels(as.factor(data.c
fit <- lmFit(data, design)             #モテ
contrast <- makeContrasts(             #比較
    G1vsG2 = G1 - G2,                  #比較
    G1vsG3 = G1 - G3,                  #比較
    G2vsG3 = G2 - G3,                  #比較
    levels = design)                   #比較
fit2 <- contrasts.fit(fit, contrast)   #モテ
out <- eBayes(fit2)                    #検定
p.value <- out$p.value                 #p値
q.value <- apply(p.value, MARGIN=2, p.adjust
ranking <- apply(p.value, MARGIN=2, rank)#p
```

R Console

```
> design <- model.matrix(~ 0 + data.cl)
> design
   data.clG1 data.clG2 data.clG3
1          1         0         0
2          1         0         0
3          1         0         0
4          1         0         0
5          0         1         0
6          0         1         0
7          0         1         0
8          0         1         0
9          0         0         1
10         0         0         1
11         0         0         1
12         0         0         1
attr(,"assign")
[1] 1 1 1
attr(,"contrasts")
attr(,"contrasts")$data.cl
[1] "contr.treatment"

> data.cl
 [1] "G1" "G1" "G1" "G1" "G2" "G2" "G2" "G2" "G3" "G3"
[11] "G3" "G3"
> |
```

3群間比較(limma)

levels関数を用いて合理的にグループラベル情報を抽出し、デザイン行列designの列名情報を変更し取扱いやすくしています。

rcode_limma_4vs4vs4.txt

```
#本番↓
#design <- model.matrix(~ 0 + as.factor(data.cl))#デザイン行列を作成した結果をdes
design <- model.matrix(~ 0 + data.cl)  #デサ
colnames(design) <- levels(as.factor(data.c
fit <- lmFit(data, design)                #モテ
contrast <- makeContrasts(                #比較
    G1vsG2 = G1 - G2,                     #比較
    G1vsG3 = G1 - G3,                     #比較
    G2vsG3 = G2 - G3,                     #比較
    levels = design)                      #比較
fit2 <- contrasts.fit(fit, contrast)      #モテ
out <- eBayes(fit2)                       #検定
p.value <- out$p.value                    #p値
q.value <- apply(p.value, MARGIN=2, p.adjust
ranking <- apply(p.value, MARGIN=2, rank)#p
```

R Console

```
> as.factor(data.cl)
 [1] G1 G1 G1 G1 G2 G2 G2 G2 G3 G3 G3 G3
Levels: G1 G2 G3
> levels(as.factor(data.cl))
[1] "G1" "G2" "G3"
> colnames(design) <- levels(as.factor(data.cl))
> design
   G1 G2 G3
1   1  0  0
2   1  0  0
3   1  0  0
4   1  0  0
5   0  1  0
6   0  1  0
7   0  1  0
8   0  1  0
9   0  0  1
10  0  0  1
11  0  0  1
12  0  0  1
attr(,"assign")
[1] 1 1 1
attr(,"contrasts")
attr(,"contrasts")$data.cl
[1] "contr.treatment"
```

3群間比較(limma)

デザイン行列の列名を変更して取扱いやすくしておかないと、この部分での指定時にややこしいことになる。ここでは3種類の2群間比較を行うようにしている。ここで作成しているのは、コントラスト行列というもの。「比較したいものを作成した行列」という位置づけ。

rcode_limma_4vs4vs4.txt

```
#本番
#design <- model.matrix(~ 0 + as.factor(data.cl))#デザイン行列を作成した結果
design <- model.matrix(~ 0 + data.cl)  #デザイン行列を作成した結果をdesignに
colnames(design) <- levels(as.factor(data.cl))#デザイン行列の列名を付与
fit <- lmFit(data, design)                    #モデル構築(ばらつきの程度を見積もっ
contrast <- makeContrasts(                    #比較したい2群の情報を作成
    G1vsG2 = G1 - G2,                         #比較したい2群の情報を作成
    G1vsG3 = G1 - G3,                         #比較したい2群の情報を作成
    G2vsG3 = G2 - G3,                         #比較したい2群の情報を作成
    levels = design)                          #比較したい2群の情報を作成
fit2 <- contrasts.fit(fit, contrast)          #モデル構築
out <- eBayes(fit2)                           #検定(経験ベイズ)
p.value <- out$p.value                        #p値
q.value <- apply(p.value, MARGIN=2, p.adjust
ranking <- apply(p.value, MARGIN=2, rank)#p
```

R Console

```
> fit <- lmFit(data, design)                    #モデル構築($
> contrast <- makeContrasts(                    #比較したい2$
+     G1vsG2 = G1 - G2,                         #比較したい2$
+     G1vsG3 = G1 - G3,                         #比較したい2$
+     G2vsG3 = G2 - G3,                         #比較したい2$
+     levels = design)                          #比較したい2$
> contrast
       Contrasts
Levels G1vsG2 G1vsG3 G2vsG3
    G1      1      1      0
    G2     -1      0      1
    G3      0     -1     -1
> |
```

教科書p180-182

3群間比較(limma)

3種類の2群間比較を行うようにしたコントラスト行列contrastを入力としているので、DEG検出結果として31,099行×3列からなるp-value行列が得られることになる。

rcode_limma_4vs4vs4.txt

```
#本番
#design <- model.matrix(~ 0 + as.factor(data.cl))#デザイン行列を作成した結果
design <- model.matrix(~ 0 + data.cl)  #デザイン行列を作成した結果をdesignに
colnames(design) <- levels(as.factor(data.cl))#デザイン行列の列名を付与
fit <- lmFit(data, design)             #モデル構築(ばらつきの程度を見積もっている)
contrast <- makeContrasts(             #比較したい2群の情報を作成
    G1vsG2 = G1 - G2,                  #比較したい2群の情報を作成
    G1vsG3 = G1 - G3,                  #比較したい2群の情報を作成
    G2vsG3 = G2 - G3,                  #比較したい2群の情報を作成
    levels = design)                   #比較したい2群の情報を作成
fit2 <- contrasts.fit(fit, contrast)   #モデル構築
out <- eBayes(fit2)                    #検定
p.value <- out$p.value                 #p値
q.value <- apply(p.value, MARGIN=2, p.adjust
ranking <- apply(p.value, MARGIN=2, rank)#p
```

R Console

```
> fit2 <- contrasts.fit(fit, contrast)   #モデル構築
> out <- eBayes(fit2)                    #検定(経験ベ$
> head(out$p.value, n=5)
            Contrasts
                 G1vsG2      G1vsG3     G2vsG3
  1367452_at 0.57066249 0.005780086 0.01612904
  1367453_at 0.07035758 0.010713438 0.30569612
  1367454_at 0.22337752 0.001491958 0.01373208
  1367455_at 0.34340818 0.004630791 0.02635020
  1367456_at 0.09121355 0.887708741 0.07149528
> dim(out$p.value)
[1] 31099     3
>
```

3群間比較(limma)

apply関数を用いて列ごと（MARGIN=2）にq-valueを計算している。

rcode_limma_4vs4vs4.txt

```
#本番
#design <- model.matrix(~ 0 + as.factor(data.cl))#デザイン行列を作成した結果をdesignに格納
design <- model.matrix(~ 0 + data.cl)  #デザイン行列を作成した結果をdesignに格納
colnames(design) <- levels(as.factor(data.cl))#デザイン行列の列名を付与
fit <- lmFit(data, design)             #モデル構築(ばらつきの程度を見積もっている)
contrast <- makeContrasts(             #比較したい2群の情報を作成
    G1vsG2 = G1 - G2,                  #比較したい2群の情報を作成
    G1vsG3 = G1 - G3,                  #比較したい2群の情報を作成
    G2vsG3 = G2 - G3,                  #比較したい2群の情報を作成
    levels = design)                   #比較したい2群の情報を作成
fit2 <- contrasts.fit(fit, contrast)   #モデル構築
out <- eBayes(fit2)                    #検定(経験ベイズ)
p.value <- out$p.value                 #p値をp.valueに格納
q.value <- apply(p.value, MARGIN=2, p.adjust, method="BH")#q値をq.valueに格納
ranking <- apply(p.value, MARGIN=2, rank)#p.valueでランキングした結果をrankingに格納
```

R Console

```
> p.value <- out$p.value                        #p値をp.valu$
> q.value <- apply(p.value, MARGIN=2, p.adjust, metho$
> head(q.value, n=4)
            Contrasts
                G1vsG2      G1vsG3     G2vsG3
  1367452_at 0.7409725 0.019218956 0.04689295
  1367453_at 0.2034828 0.031828163 0.45396064
  1367454_at 0.4238707 0.006279388 0.04125726
  1367455_at 0.5520626 0.016024589 0.06949329
>
```

教科書p180-182

3群間比較(limma)

FDR 1%という閾値を満たす遺伝子数は、G1 vs. G2が2,418個、G1 vs. G3が8,119個、そしてG2 vs. G3が7,471個という結果。

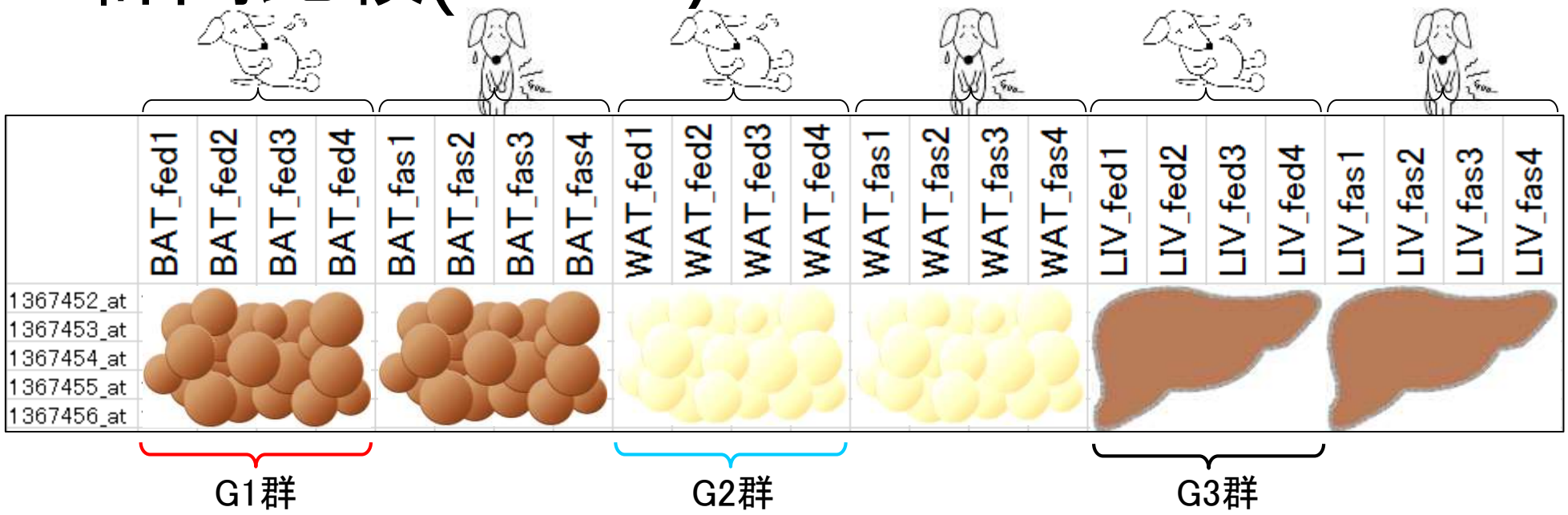
```
p.value <- out$p.value                    #p値をp.valueに格納
q.value <- apply(p.value, MARGIN=2, p.adjust, method="BH")#q値をq.valueに格納
ranking <- apply(p.value, MARGIN=2, rank)#p.valueでランキングした結果をrankingに格納

#ファイルに保存(テキストファイル)
tmp <- cbind(rownames(data), data, p.value, q.value, ranking)#入力データの右側にDEG検出結果を結合したものをtmpに格納
write.table(tmp, out_f1, sep="¥t", append=F, quote=F, row.names=F)#tmpの中身を指定したファイル名で保存

# ベン図
sum(q.value[,1] < 0.01)
sum(q.value[,2] < 0.01)
sum(q.value[,3] < 0.01)
vennDiagram(decideTests(out, adjust.method=
```

R Console

```
> #ファイルに保存(テキストファイル)
> tmp <- cbind(rownames(data), data, p.value, q.value$
> write.table(tmp, out_f1, sep="\t", append=F, quote=$
>
> # ベン図
> sum(q.value[,1] < 0.01)
[1] 2418
> sum(q.value[,2] < 0.01)
[1] 8119
> sum(q.value[,3] < 0.01)
[1] 7471
> vennDiagram(decideTests(out, adjust.method="BH", p.$
>
```

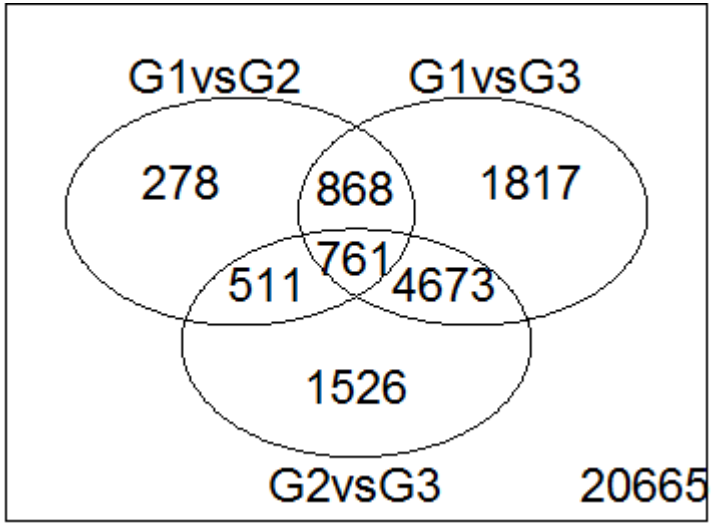
3群間比較(limma)

G1vsG2のDEG数が他に比べて少ないので妥当



R Console

```
> # ベン図
> sum(q.value[,1] < 0.01)
[1] 2418
> sum(q.value[,2] < 0.01)
[1] 8119
> sum(q.value[,3] < 0.01)
[1] 7471
> vennDiagram(decideTests(out, adj$
>
```



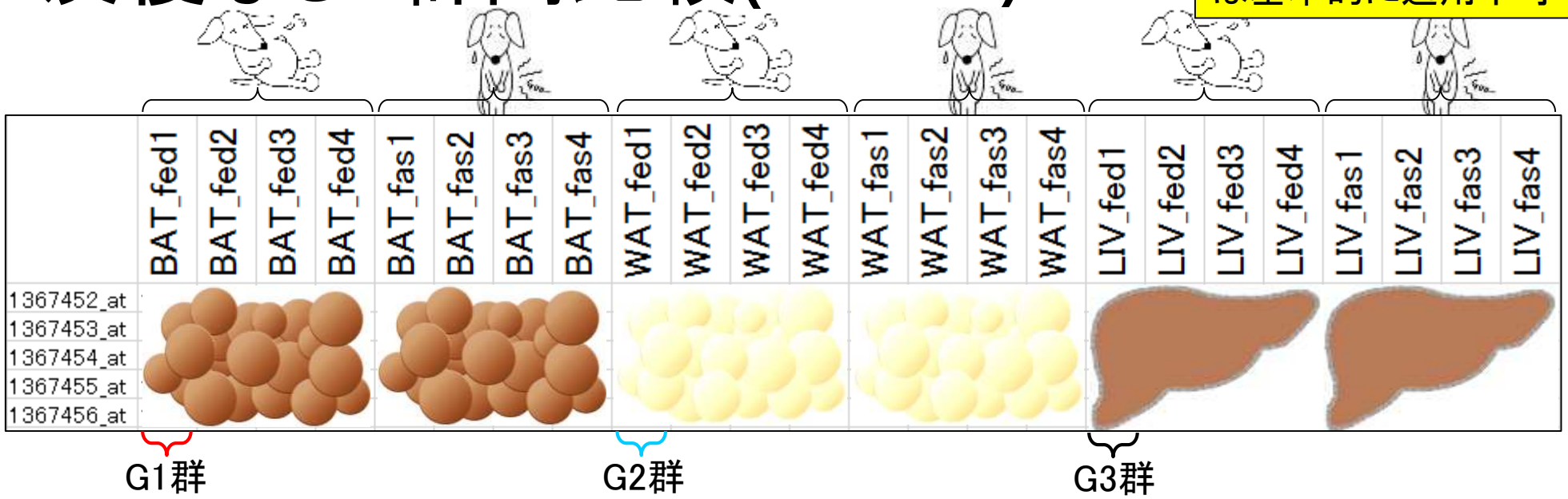
Contents

- デザイン行列の意味を理解（教科書p173–182）
  - limmaパッケージを用いた2群間比較のおさらい
  - limmaパッケージを用いた3群間比較（反復あり）
- 反復なし多群間比較（教科書p182–188）
  - limmaパッケージを用いた3群間比較（反復なし）
  - TCCパッケージ中のROKU法を用いた特異的発現遺伝子検出
- 機能解析（遺伝子セット解析）
  - 基本的な考え方
  - 前処理
    - MSigDBからの遺伝子セット情報（gmt形式ファイル）取得
    - ID変換（probe ID → gene symbol）
  - GSAパッケージを用いた遺伝子セット解析

教科書 § 4.2.2

反復なし3群間比較(limma)

(biological) replicatesがないデータの場合limmaは基本的に適用不可



rcode_limma_1vs1vs1.txt

```
##########################################################################↓
###  1 BAT_fed sample vs. 1 WAT_fed sample vs. 1 LIV_fed sample ###↓
##########################################################################↓
in_f <- "data_mas_EN.txt"                  #入力ファイル名を指定してin_fに格納↓
out_f1 <- "hoge1.txt"                      #出力ファイル名を指定してout_f1に格納↓
param_G1 <- 1                              #G1群のサンプル数を指定↓
param_G2 <- 1                              #G2群のサンプル数を指定↓
param_G3 <- 1                              #G2群のサンプル数を指定↓
param_posi <- c(1, 9, 17)                  #元の発現行列上での列番号を指定↓
param_FDR <- 0.05                          #DEG検出時のfalse discovery rate (FDR)閾値を指定↓
```

教科書 § 4.2.2

反復なし3群間比較(limma)

(biological) replicatesがないデータの場合limmaは基本的に適用不可

```
##################################################################
###  1 BAT_fed sample vs. 1 WAT_fed sample vs. 1 LIV_fed sample ###
##################################################################
in_f <- "data_mas_EN.txt"                #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge1.txt"                    #出力ファイル名を指定してout_f1に格納
param_G1 <- 1                            #G1群のサンプル数を指定
param_G2 <- 1                            #G2群のサンプル数を指定
param_G3 <- 1                            #G3群のサンプル数を指定
param_posi <- c(1, 9, 17)                #元の発現行列上での列番号を指定
param_FDR <- 0.05                        #DEG検出時のfalse discovery rate (FDR)閾値を指

#必要なパッケージをロード
library(limma)                           #パッケージの読み込み

#入力ファイルの読み込みとラベル情報の作成、そ
data <- read.table(in_f, header=TRUE, row.name
data.cl <- c(rep("G1", param_G1), rep("G2", pa
data <- data[,param_posi]                #サブセ
colnames(data)                           #サブセ

#本番
#design <- model.matrix(~ 0 + as.factor(data.c
design <- model.matrix(~ 0 + data.cl)  #デザイ
colnames(design) <- levels(as.factor(data.cl))
fit <- lmFit(data, design)               #モデル
contrast <- makeContrasts(               #比較し
    G1vsG2 = G1 - G2,                    #比較し
    G1vsG3 = G1 - G3,                    #比較し
    G2vsG3 = G2 - G3,                    #比較し
    levels = design)                     #比較し
fit2 <- contrasts.fit(fit, contrast)     #モデル
out <- eBayes(fit2)                      #検定(
p.value <- out$p.value                   #p値を
q.value <- apply(p.value, MARGIN=2, p.adjust,
ranking <- apply(p.value, MARGIN=2, rank)#p.va
```

R Console

```
> #design <- model.matrix(~ 0 + as.factor(data.cl))#デザイ$
> design <- model.matrix(~ 0 + data.cl)   #デザイン行列を作$
> colnames(design) <- levels(as.factor(data.cl))#デザイン$
> fit <- lmFit(data, design)                #モデル構築(ばら$
> contrast <- makeContrasts(                #比較したい2群の$
+     G1vsG2 = G1 - G2,                     #比較したい2群の$
+     G1vsG3 = G1 - G3,                     #比較したい2群の$
+     G2vsG3 = G2 - G3,                     #比較したい2群の$
+     levels = design)                      #比較したい2群の$
> fit2 <- contrasts.fit(fit, contrast)    #モデル構築
> out <- eBayes(fit2)                     #検定(経験ベイズ)
Error in ebayes(fit = fit, proportion = proportion, stdev.$
  No residual degrees of freedom in linear model fits
>
```

教科書 § 4.2.3

反復なし多群間比較(TCC)

ROKU入出力のイメージ。出力は入力の対応する位置に「特異的組織でない部分には0、特異的高発現が1、特異的低発現が−1」と返す。①出力の右側は全体的な組織特異度の高さでランキングした結果を返す。黒枠内の遺伝子（gene2, 7, 8）は組織特異的遺伝子ではない。

入力

	tissue1	tissue2	tissue3	tissue4	tissue5	tissue6	tissue7	tissue8	tissue9	tissue10
gene1	0.00	0.00	0.00	0.00	0.00	0.00	9.00	0.00	0.00	0.00
gene2	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00
gene3	0.02	0.41	0.41	0.38	9.60	0.49	0.44	0.16	0.21	0.52
gene4	3.95	4.12	4.20	9.20	3.84	3.97	4.23	3.80	8.60	3.64
gene5	8.06	7.93	3.00	7.82	7.75	8.42	8.06	7.75	7.88	8.26
gene6	5.20	5.00	8.50	5.10	4.84	4.78	5.00	1.30	5.00	4.89
gene7	8.20	9.30	8.00	6.90	8.60	7.30	8.00	6.70	8.20	7.50
gene8	1.20	2.10	4.80	2.00	3.50	2.50	3.65	0.30	3.10	3.63

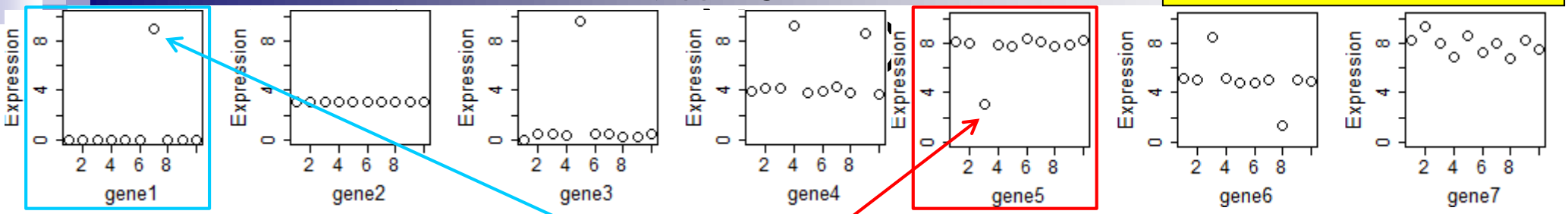
①

出力

	tissue1	tissue2	tissue3	tissue4	tissue5	tissue6	tissue7	tissue8	tissue9	tissue10	modH	ranking
gene1	0	0	0	0	0	0	1	0	0	0	0.000	1
gene2	0	0	0	0	0	0	0	0	0	0	3.322	8
gene3	0	0	0	0	1	0	0	0	0	0	0.768	2
gene4	0	0	0	1	0	0	0	0	1	0	1.718	5
gene5	0	0	−1	0	0	0	0	0	0	0	1.492	3
gene6	0	0	1	0	0	0	0	−1	0	0	1.645	4
gene7	0	0	0	0	0	0	0	0	0	0	2.952	6
gene8	0	0	0	0	0	0	0	0	0	0	3.032	7

教科書 § 4.2.3

高発現は1、低発現は-1



入力

	tissue1	tissue2	tissue3	tissue4	tissue5	tissue6	tissue7	tissue8	tissue9	tissue10
gene1	0.00	0.00	0.00	0.00	0.00	0.00	9.00	0.00	0.00	0.00
gene2	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00
gene3	0.02	0.41	0.41	0.38	9.60	0.49	0.44	0.16	0.21	0.52
gene4	3.95	4.12	4.20	9.20	3.84	3.97	4.23	3.80	8.60	3.64
gene5	8.06	7.93	3.00	7.82	7.75	8.42	8.06	7.75	7.88	8.26
gene6	5.20	5.00	8.50	5.10	4.84	4.78	5.00	1.30	5.00	4.89
gene7	8.20	9.30	8.00	6.90	8.60	7.30	8.00	6.70	8.20	7.50
gene8	1.20	2.10	4.80	2.00	3.50	2.50	3.65	0.30	3.10	3.63

出力

	tissue1	tissue2	tissue3	tissue4	tissue5	tissue6	tissue7	tissue8	tissue9	tissue10	modH	ranking
gene1	0	0	0	0	0	0	1	0	0	0	0.000	1
gene2	0	0	0	0	0	0	0	0	0	0	3.322	8
gene3	0	0	0	0	1	0	0	0	0	0	0.768	2
gene4	0	0	0	1	0	0	0	0	1	0	1.718	5
gene5	0	0	-1	0	0	0	0	0	0	0	1.492	3
gene6	0	0	1	0	0	0	0	-1	0	0	1.645	4
gene7	0	0	0	0	0	0	0	0	0	0	2.952	6
gene8	0	0	0	0	0	0	0	0	0	0	3.032	7

反復なし多群間比較(TCC)

入出力のイメージは、この例題実行結果。modH列の値が、「データ変換後のエントロピー値」に相当。（2015.04.21の講義でほんの少しだけ触れました）

(Rで)マイクロアレイデータ解析

(last modified 2015/05/25, since 2005)

What's

- 門田
- お知

- 解析 | 発現変動 | 2群間 | 対応あり | 時系列 | maSigPro (Conesa_2006) (last modified 2013/6/2)
- 解析 | 発現変動 | 3群間 | 対応なし | について(last modified 2015/01/16)
- 解析 | 発現変動 | 3群間 | 対応なし | Mulcom (Isella_2011)(last modified 2013/12/06)
- 解析 | 発現変動 | 3群間 | 対応なし | limma (Smyth_2004)(last modified 2014/02/03)
- 解析 | 発現変動 | 3群間 | 対応なし | 一元配置分散分析(One-way ANOVA) (last modified 2013/11/12)
- 解析 | 発現変動 | 3群間 | 対応なし | Kruskal-Wallis(クラスカル-ウォリス) 検定 (last modified 2013/6/2)
- 解析 | 発現変動 | 多群間 | について(last modified 2013/6/2)
- 解析 | 発現変動 | 多群間 | SpeCond(Cavalli_2011) (last modified 2013/6/10)
- 解析 | 発現変動 | 多群間 | ROKU(Kadota_2006) (last modified 2014/05/30)
- 解析 | 発現変動 | 多群間 | Sprent's non-parametric method(Ge_2005) (last modified 2009/07/31)
- 解析 | 発現変動 | 多群間 | Schug's H(

解析 | 発現変動 | 多群間 | ROKU (Kadota_2006)

TCCパッケージで提供しているROKU法(Kadota et al., 2006)を用いて、遺伝子発現行列中の遺伝子を全体的な組織特異性の度合いでランキングします。出力ファイル中の"modH"列の数値は、「ROKU論文中のAdditional file 1 (Suppl.xls)の"H(x')"列の数値」と対応しています。つまり、データ変換後のエントロピー値です。"ranking"列は、modHの値でランキングした結果です。"ranking"列で昇順にソートすることで、全体的な組織特異性の度合いでランキングしていることになります。つまり、上位が「(どの組織で特異的かはこのスコアだけでは分からないが)組織特異性が高い遺伝子」ということになります。残りの結果は「1:特異的高発現、-1:特異的低発現、0:その他」からなる「外れ値行列」です。例えば、組織AとBで1, それ以外の組織で0を示す遺伝子(群)は「AとB特異的高発現遺伝子」と判断します。

「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し、以下をコピペ

1. サンプルデータ21のsample21.txtの場合:

log2変換後のデータであるという前提です。

```
in_f <- "sample21.txt"                    #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.txt"                      #出力ファイル名を指定してout_fに格納

#必要なパッケージをロード
library(TCC)                              #パッケージの読み込み

#入力ファイルの読み込み
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="")#in_fで指定したファ
```

参考

Sequence logosは、あるポジションに特定の塩基が濃縮されている状態をうまく表すために、エントロピーを内部的に計算しているだけです。水色の枠内がエントロピー。

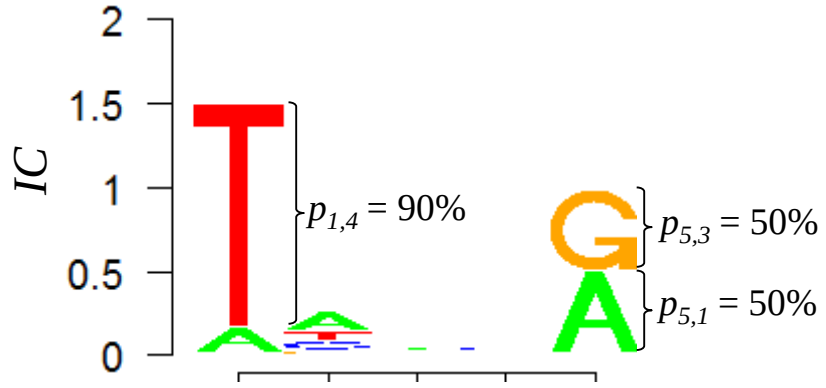
Sequence logos

- position iの情報量 $IC_i = \log_2(N) - H(x_i)$

$\log_2(N) = 2$

position i

	1	2	3	4	5	…
配列 1	T	A	C	G	G	…
配列 2	T	A	A	C	G	…
配列 3	T	G	T	A	G	…
配列 4	A	C	T	T	A	…
配列 5	T	T	G	G	A	…
配列 6	T	C	A	A	G	…
配列 7	T	A	C	T	A	…
配列 8	T	T	G	C	A	…
配列 9	T	A	A	C	A	…
配列10	T	A	C	T	G	…



IC	1.53	0.24	0.03	0.03	1.00	…

x_{ij}	1	2	3	4	5	…
Aの数 (j=1)	1	5	3	2	5	…
Cの数 (j=2)	0	2	3	3	0	…
Gの数 (j=3)	0	1	2	2	5	…
Tの数 (j=4)	9	2	2	3	0	…
$\Sigma_j\ x_{ij}$	10	10	10	10	10	

j

p_{ij}	1	2	3	4	5	…
1	0.1	0.5	0.3	0.2	0.5	…
2	0.0	0.2	0.3	0.3	0.0	…
3	0.0	0.1	0.2	0.2	0.5	…
4	0.9	0.2	0.2	0.3	0.0	…
Σ_j	1.0	1.0	1.0	1.0	1.0	

$-p_{ij}\log_2(p_{ij})$	1	2	3	4	5	…
1	0.33	0.50	0.52	0.46	0.50	…
2	0.00	0.46	0.52	0.52	0.00	…
3	0.00	0.33	0.46	0.46	0.50	…
4	0.14	0.46	0.46	0.52	0.00	…
$H = \Sigma_j$	0.47	1.76	1.97	1.97	1.00	

Schneider and Stephens, *Nucleic Acids Res.*, **18**(20): 6097-6100, 1990

 参考

組織特異的発現遺伝子検出にエントロピーを最初に応用したのはSchugら(2005)。

エントロピー

- 遺伝子iのエントロピー $H(x_i) = -\sum_{j=1}^{N} p_{ij} \log_2(p_{ij}),\ \text{where}\ p_{ij} = x_{ij} / \sum_{j=1}^{N} x_{ij}$

x_{ij}

j	遺伝子1	遺伝子2	遺伝子3	遺伝子4	遺伝子5	...
組織1	1	5	6	4	10	...
組織2	0	2	6	4	10	...
組織3	0	1	6	4	10	...
組織4	9	2	6	4	10	...
組織5	0	4	6	10	4	
組織6	0	6	6	4	10	
組織7	0	3	6	4	10	
組織8	0	5	6	4	10	
$\Sigma_j\ x_{ij}$	10	28	48	38	74	

p_{ij}

	遺伝子1	遺伝子2	遺伝子3	遺伝子4	遺伝子5	...
1	0.10	0.18	0.13	0.11	0.14	...
2	0.00	0.07	0.13	0.11	0.14	...
3	0.00	0.04	0.13	0.11	0.14	...
4	0.90	0.07	0.13	0.11	0.14	...
5	0.00	0.14	0.13	0.26	0.05	
6	0.00	0.21	0.13	0.11	0.14	
7	0.00	0.11	0.13	0.11	0.14	
8	0.00	0.18	0.13	0.11	0.14	
Σ_j	1.00	1.00	1.00	1.00	1.00	

$-p_{ij} \log_2(p_{ij})$

	遺伝子1	遺伝子2	遺伝子3	遺伝子4	遺伝子5	...
1	0.33	0.44	0.38	0.34	0.39	...
2	0.00	0.27	0.38	0.34	0.39	...
3	0.00	0.17	0.38	0.34	0.39	...
4	0.14	0.27	0.38	0.34	0.39	...
5	0.00	0.40	0.38	0.51	0.23	
6	0.00	0.48	0.38	0.34	0.39	
7	0.00	0.35	0.38	0.34	0.39	
8	0.00	0.44	0.38	0.34	0.39	
Σ_j	0.47	2.83	3.00	2.90	2.96	

組織特異的遺伝子は低いエントロピー

そうでないものは高い値

N:組織数(jの数) = 8

Hの取りうる範囲:$0 \leqq H \leqq \log_2 N \rightarrow 0 \leqq H \leqq 3$

Schug et al., *Genome Biol.*, **6**: R33, 2005

hoge – GSE2361フォルダ中のデータを用いてROKUを実行してみよう。

ROKUを実行

- Affymetrix GeneChip
  - Ge et al., *Genomics*, **86**: 127–141, 2005
    - **GSE2361**、GPL96 (Affymetrix Human Genome U133A Array)、22,283 probesets
    - **ヒト36サンプル**：Heart (心臓)、Thymus (胸腺)、Spleen (脾臓)、Ovary (卵巣)、Kidney (腎臓)、Skeletal Muscle (骨格筋)、Pancreas (膵臓)、Prostate (前立腺)、…

BMC Bioinformatics. 2006 Jun 12;7:294.

ROKU: a novel method for identification of tissue-specific genes.

Kadota K[1], Ye J, Nakai Y, Terada T, Shimizu K.

Author information

Abstract

BACKGROUND: One of the important goals of mi
expression is considerably higher or lower in some
identifying such tissue-specific genes.

RESULTS: We describe a method, ROKU, which
many tissues and thousands of genes. ROKU ran
Shannon entropy and detects tissues specific to e
evaluated the capacity for the detection of various
We observed that ROKU was superior to a conver
according to overall tissue specificity and to detect
objective tissues.

CONCLUSION: ROKU is useful for the detection
is also directly applicable to the selection of diagn

Analysis of real data

To further investigate the validity of our method (ROKU), we applied the method to a public gene expression matrix consisting of 36 normal human tissues and 22,283 probesets [5]. Briefly, ROKU (1) processes each probeset expression vector and makes a processed vector $\boldsymbol{x}'$, (2) calculates the entropy $H(\boldsymbol{x}')$, and (3) assigns specific tissues to each probeset whose observations are detected to be 'outliers' (see Methods). We compared the performance of ROKU to that of Schug's method, which directly uses the original/non-processed vector $\boldsymbol{x}$ for measuring the entropy $H(\boldsymbol{x})$ [4]. The two entropy scores ($H(\boldsymbol{x}')$ and $H(\boldsymbol{x})$) for all probesets are available in the additional file [see 1].

Additional file 1. Full information analyzed by ROKU for dataset of Ge et al. (2005). For the original gene expression matrix, an outlier matrix (consisting of 1 for over-expressed outliers, -1 for under-expressed outliers, and 0 for non-outliers) is provided. It also contains two entropy scores measured by ROKU and Schug's method and their ranks.

Format: XLS Size: 8.1MB Download file

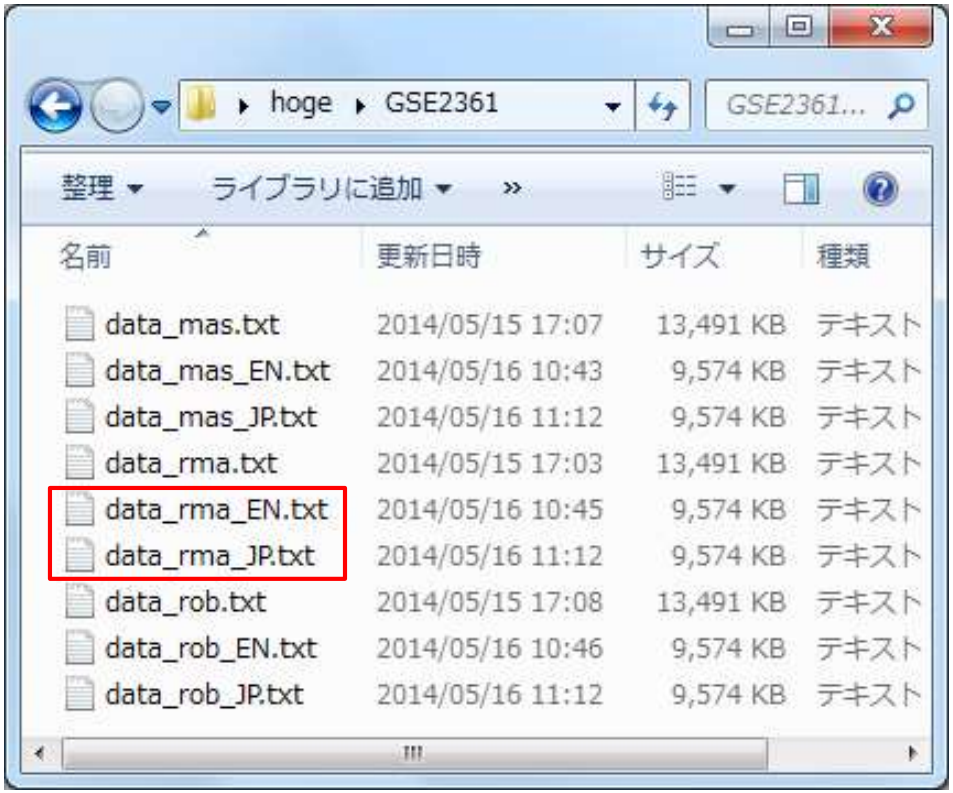
This file can be viewed with: Microsoft Excel Viewer

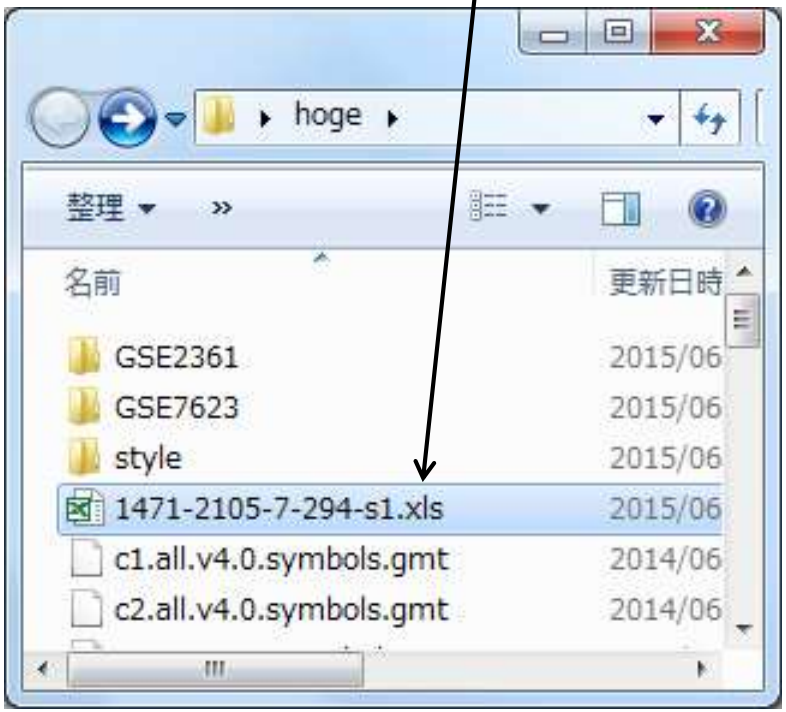
OPEN DATA

ROKUを実行

赤枠のRMA前処理後のデータを入力としてROKUを実行した結果(の一部)が、基本的にROKU論文のAdditional file 1と同じです。

- Affymetrix GeneChip
  - Ge et al., *Genomics*, **86**: 127–141, 2005
    - **GSE2361**、GPL96 (Affymetrix Human Genome U133A Array)、22,283 probesets
    - **ヒト36サンプル**:Heart (心臓)、Thymus (胸腺)、Spleen (脾臓)、Ovary (卵巣)、Kidney (腎臓)、Skeletal Muscle (骨格筋)、Pancreas (膵臓)、Prostate (前立腺)、…





教科書 § 4.2.3

ROKUを実行

テンプレートスクリプトをテキストエディタにコピペして、入力ファイル名部分を変更し、R Console画面上でコピペ。約2分かかります。

解析 | 発現変動 | 多群間 | ROKU (Kadota_2006)

TCCパッケージで提供しているROKU法(Kadota et al., 2006)を用いて、遺伝子発現行列中の遺伝子を全体的な組織特異性の度合いでランキングします。出力ファイル中の"modH"列の数値は、「ROKU論文中のAdditional file 1(Suppl.xls)の"H(x')"列の数値」と対応しています。つまり、データ変換後のエントロピー値です。"ranking"列は、modHの値でランキングした結果です。"ranking"列で昇順にソートすることで、全体的な組織特異性の度合いでランキングしていることになります。つまり、上位が「(どの組織で特異的かはこのスコアだけでは分からないが)組織特異性が高い遺伝子」ということになります。残りの結果は「1:特異的高発現、-1:特異的低発現、0:その他」からなる「外れ値行列」です。例えば、組織AとBで1, それ以外の組織で0を示す遺伝子(群)は「AとB特異的高発現遺伝子」と判断します。

「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し、以下をコピペ

1. サンプルデータ21のsample21.txtの場合:

log2変換後のデータであるという前提です。

```
in_f <- "sample21.txt"
out_f <- "hoge1.txt"

#必要なパッケージをロード
library(TCC)

#入力ファイルの読み込み
data <- read.table(in_f, header=

#本番
hoge <- ROKU(data)
outlier <- hoge$outlier
modH <- hoge$modH
ranking <- hoge$rank

#ファイルに保存
tmp <- cbind(rownames(data), out
write.table(tmp, out_f, sep="\t
```

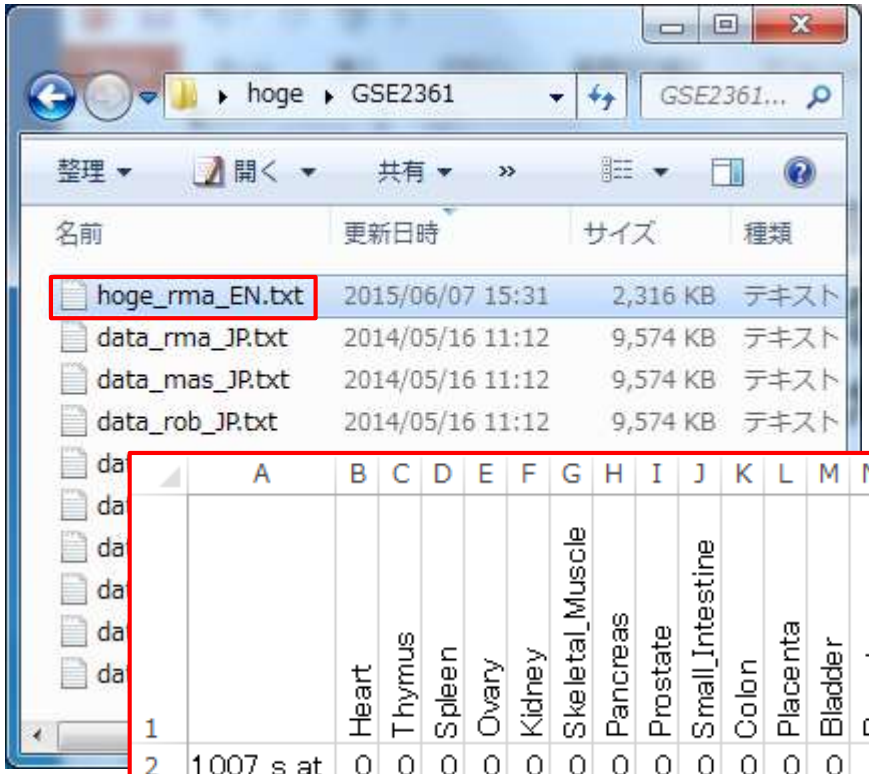
```
in_f <- "data_rma_EN.txt"          #入力ファイル名を指定してin_fに格納
out_f <- "hoge_rma_EN.txt"         #出力ファイル名を指定してout_fに格納

#必要なパッケージをロード
library(TCC)                       #パッケージの読み込み

#入力ファイルの読み込み
data <- read.table(in_f, header=TRUE, row.names=1, sep="¥t", quote="")#in_f

#本番
hoge <- ROKU(data)                 #ROKUを実行した結果をhogeに格納
outlier <- hoge$outlier            #外れ値行列をoutlierに格納
modH <
rankin
#ファイ
tmp <-
write.
```

R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/GSE2361"
> list.files(pattern="rma")
[1] "data_rma.txt"    "data_rma_EN.txt" "data_rma_JP.txt"
>
```

ROKUを実行

ROKU論文のAdditional file 1と若干結果が違いますが、Rやパッケージのバージョンの違い、ROKU内部で用いている階乗計算部分が原著論文では近似式(Staring)でしたが、TCCに移植する際に別の関数に置換したことなどが原因と考えられます。



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM
1		Heart	Thymus	Spleen	Ovary	Kidney	Skeletal_Muscle	Pancreas	Prostate	Small_Intestine	Colon	Placenta	Bladder	Breast	Uterus	Thyroid	Skin	Salivary_Gland	Trachea	Cerebellum	Brain	Fetal_Brain	Adrenal_Gland	Bone_Marrow	Amygdala	Caudate_Nucleus	Corpus	Hippocampus	Thalamus	Pituitary_Gland	Spinal_Cord	Testis	Liver	Stomach	Lung	Fetal_Lung	Fetal_Liver	modH	ranking
2	1007_s_at	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4.81	20121
3	1053_at	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	4.54	7074
4	117_at	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	4.71	15317
5	121_at	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3.97	1419
6	1255_g_at	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	0	1	0	1	0	0	0	0	0	4.13	2045
7	1294_at	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4.85	21291
8	1316_at	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4.66	12535
9	1320_at	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	4.57	8350
10	1405_i_at	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	4.19	2329
11	1431_at	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	2.66	76
12	1438_at	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	4.57	8065
13	1487_at	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4.74	16878

Tips

ROKUの詳細について記述しています。また、リクエストのあった①WAD法の数式や、②前処理法とDEG検出法の組合せのガイドラインについてもこのPDFに記載あり。教科書p170にも記載あり。

門田　幸二のホームページ

名前　門田　幸二(かどた　こうじ)

所属

身分

研究分野

研究テーマ

トランスクリプトー
どによって得られ
応用を目指しま

講演など(上記講義以外)(last modified: 2015.04.23)

34. 門田幸二,「Rで塩基配列解析:ゲノム解析からトランスクリプトーム解析まで」, HPCI講習会・バイオインフォマティクス実習コース, 産総研・臨海副都心センター(東京), 2016.03.03-04
33. 門田幸二,「演題未定」, 2015.10.21-23

32. 門田幸二,「Rでゲノム・トランスクリプトーム解析:CpG解析から機能解析まで」, HPCI講習会・バイオインフォマティクス実習コース, 産総研・臨海副都心センター(東京), 2015.03.05-06
31. 門田幸二,「
2014・中級
30. 門田幸二,「
析」, 仙台国
29. 門田幸二,「
代シークエン
28. 門田幸二,「
リーズ, イル
27. 門田幸二,「
シークエン
26. 門田幸二,「
産総研・臨

13. 門田幸二,「トランスクリプトーム解析におけるバイオインフォマティクス要素技術～私の相場観～」, 日本バイオインフォマティクス学会・第2回アグリバイオインフォマティクス研究会第1部, 琉球大学(沖縄), 2010.09.17
12. 門田幸二,「マイクロアレイデータ解析結果の正しい?!解釈について」, 東京大学大学院農学生命科学研究科アグリバイオインフォマティクス教育研究プログラム・マイクロアレイデータ解析講習会, 東京大学(東京), 2009.11.20 (第1回), 2009.11.24 (第2回)
11. 門田幸二,「トランスクリプトームデータの解析戦略とその周辺」, 東京大学大学院農学生命科学研究科第36回アグリバイオインフォマティクス セミナー, 東京大学(東京), 2009.10.21
10. 門田幸二,「マイクロアレイを用いた遺伝子発現解析」, 基礎生物学研究所 バイオインフォマティクス・トレーニングコース 2009, 基礎生物学研究所(愛知), 2009.8.19-21 (第1回), 2009.09.08-10 (第2回)
9. 門田幸二,「感度・特異度・再現性高く発現変動遺伝子を検出するための推奨ガイドライン」, 東京大学ILSI Japan寄付講座「機能性食品ゲノミクス」公開シンポジウム・食品の機能予測とニュートリゲノミクス, 東京大学(東京), 2009.5.13
8. 門田幸二,「マイクロアレイ解析の話:発現変動遺伝子検出あたりを中心に」, 日本バイオインフォマティクス学会・第5回九州地域部会講習会, 九州大学(福岡), 2009.02.26-27
7. 門田幸二,「トランスクリプトーム解析手法の開発」, アグリバイオインフォマティクス成果報告シンポジウム, 東京大学(東京), 2008.12.8

Contents

- デザイン行列の意味を理解（教科書p173–182）
  - limmaパッケージを用いた2群間比較のおさらい
  - limmaパッケージを用いた3群間比較（反復あり）
- 反復なし多群間比較（教科書p182–188）
  - limmaパッケージを用いた3群間比較（反復なし）
  - TCCパッケージ中のROKU法を用いた特異的発現遺伝子検出
- 機能解析（遺伝子セット解析）
  - 基本的な考え方
  - 前処理
    - MSigDBからの遺伝子セット情報（gmt形式ファイル）取得
    - ID変換（probe ID → gene symbol）
  - GSAパッケージを用いた遺伝子セット解析

機能解析

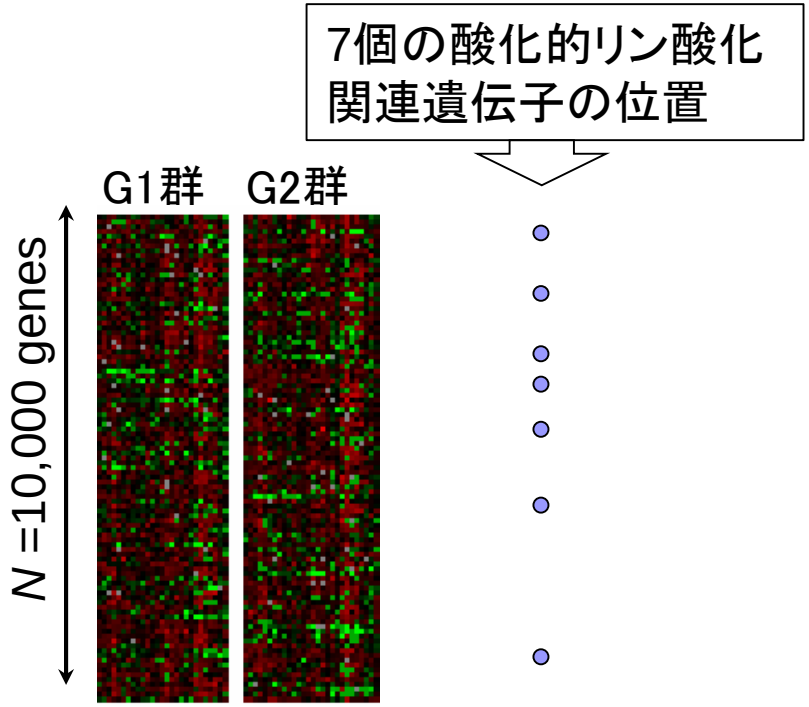
機能解析の実体は、遺伝子セットの発現変動解析。発現に差のある遺伝子セットを探したい、ということ

- Gene Ontology (GO)解析(発現に差のあるGO termを探索)
  - 基本3カテゴリ(Cellular Component (CC), Molecular Function (MF), Biological Process (BP))のどれでも可能
    - 例:肝臓の空腹状態 vs. 満腹状態のGO(BP)解析の結果、「脂肪酸β酸化」関連GO term (GO:0006635)が動いていることが分かった
- パスウェイ解析(発現に差のあるパスウェイを探索)
  - KEGG Pathway, BioCarta, Reactome pathway databaseのどれでも可能
    - 例:酸化的リン酸化パスウェイ関連遺伝子セットが糖尿病患者で動いていた
- モチーフ解析(発現に差のあるモチーフを探索)
  - 同じ3'-UTR microRNA結合モチーフをもつ遺伝子セット
  - 同じ転写因子結合領域(TATA-boxなど)をもつ遺伝子セット
    - 例:TATA-boxをもつ遺伝子セットがG1群 vs. G2群比較で動いていた
- ...

機能解析（遺伝子セット解析）

酸化的リン酸化関連遺伝子セットが変動しているかどうかを調べたい、という問題を考える。

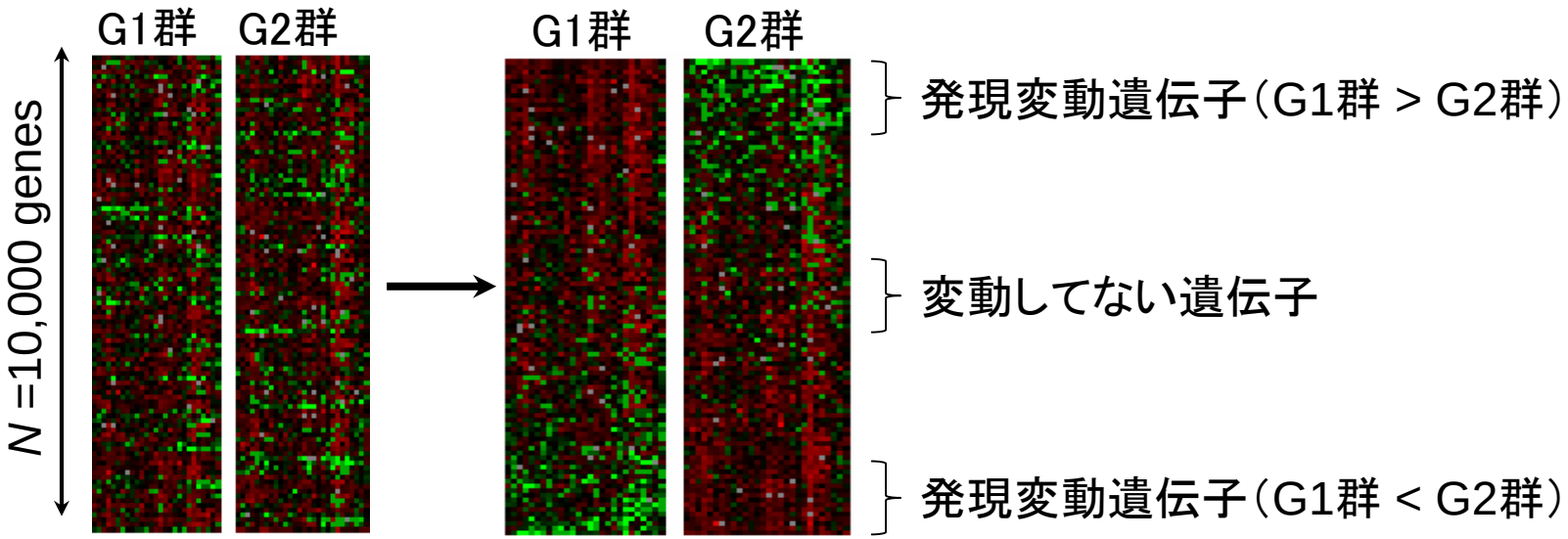
- 発現変動遺伝子セット解析手法（2群間比較用がほとんど）
  - *N* =10,000個の遺伝子からなる2群間比較用データ
  - この中に、XXX関連遺伝子が*n*個含まれている
    - 例：酸化的リン酸化（=XXX）関連遺伝子が7（=*n*）個含まれている



基本はデフォルトでやるようですが、様々な選択肢があります

機能解析（遺伝子セット解析）

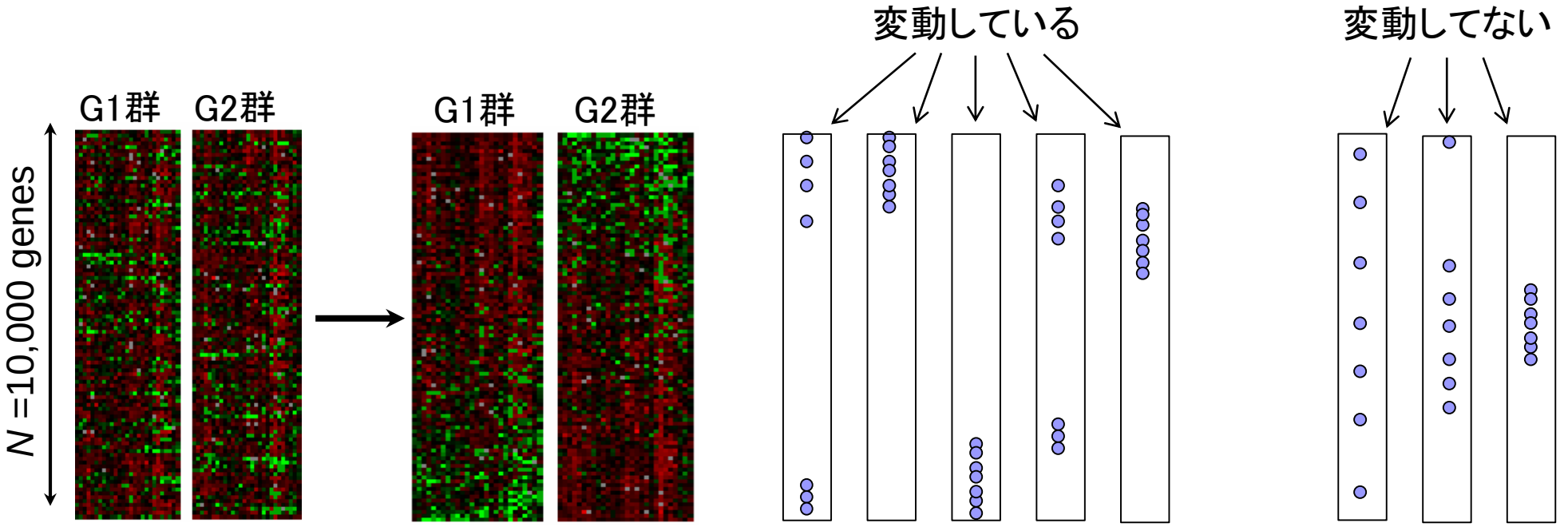
- 遺伝子ごとの発現変動の度合いを数値化
  - 例：t-統計量、log2（G2/G1）、相関係数、…



どうやって偏りを評価するのか?

機能解析(遺伝子セット解析)

- 発現変動順にソート後の酸化的リン酸化関連遺伝子セットのステレオタイプな分布

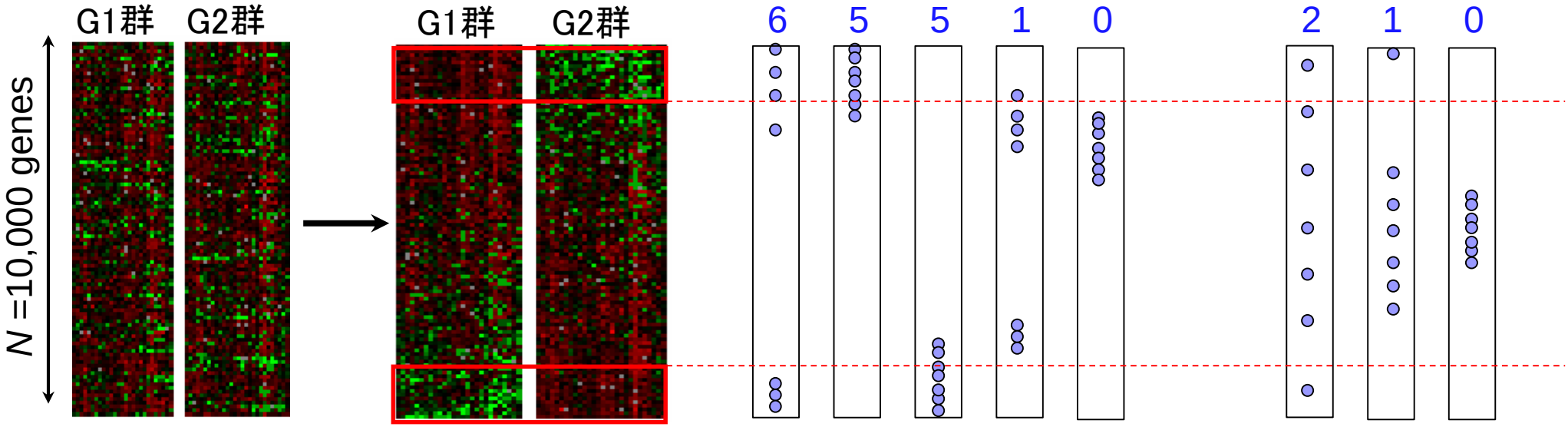


機能解析（遺伝子セット解析）

基本的な考え方は、「全遺伝子」と「上位のサブセット」のみで、調べたい遺伝子セットの割合が不変という帰無仮説のもとで検定

- Over-Representation Analysis (ORA)
  - 何らかの手段で決めた上位X(=1500)個のうち、x個が酸化的リン酸化関連遺伝子であった

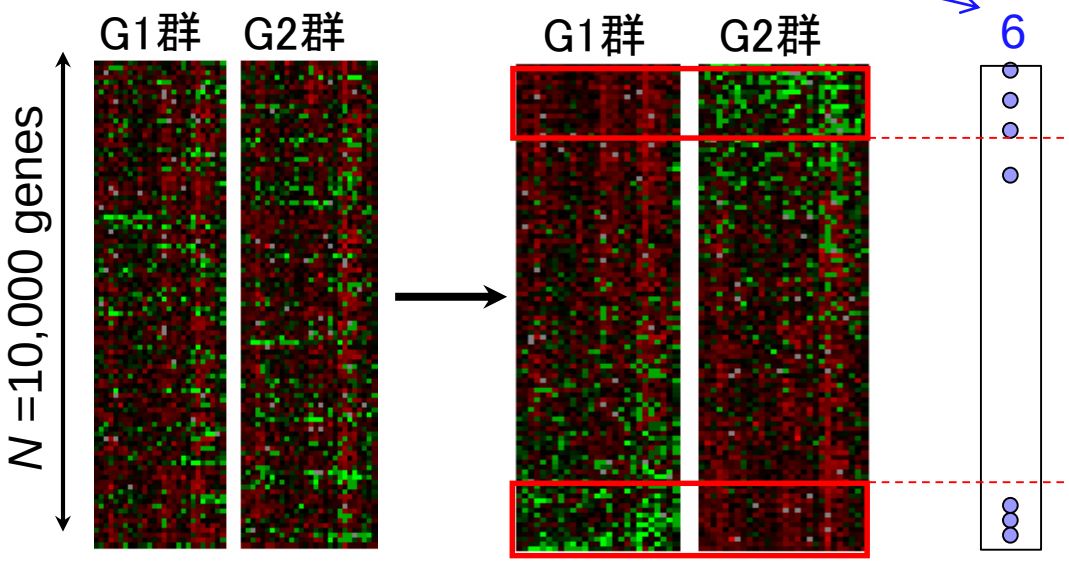
酸化的リン酸化関連遺伝子セット（n=7）が変動していない場合：$x/n \fallingdotseq X/N$ (= 1500/10000)
酸化的リン酸化関連遺伝子セット（n=7）が変動している場合：$x/n >> X/N$ (= 15%)



機能解析（遺伝子セット解析）

2×2分割表（contingency table）に基づく方法。超幾何検定やカイ二乗検定が利用されます。

- Over-Representation Analysis (ORA)
  - 何らかの手段で決めた上位*X*(=1500)個のうち、*x*個が酸化的リン酸化関連遺伝子であった



XXX=酸化的リン酸化関連遺伝子セット

	XXX	XXX以外	計
non-DEG数	1	8500-1	N-X
DEG数	6	1500-6	X
計	n	N-n	

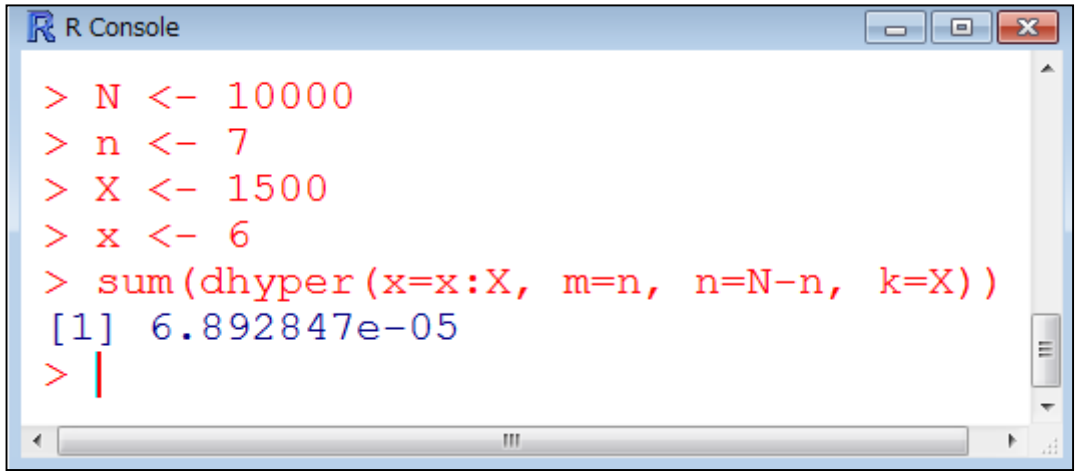
機能解析(超幾何検定)

DEGとして1500個抽出したとき、酸化的リン酸化関連遺伝子が6個以上含まれる確率として算出

- *N*=10000個の遺伝子発現データ中にXXX=酸化的リン酸化関連遺伝子は*n*=7個含まれていた。上位*X*=1500個の発現変動遺伝子(DEG)の中に*x*=6個の酸化的リン酸化関連遺伝子が含まれていた
  - 帰無仮説:酸化的リン酸化関連遺伝子の割合はDEGとnon-DEG間で差がない

	XXX	XXX以外	計
non-DEG数	1	8500-1	8500
DEG数	6	1500-6	1500
計	7	9993	10000

	XXX	XXX以外	計
non-DEG数	n-x	$(N$-$n)$-$(X$-$x)$	N-X
DEG数	x	X-x	X
計	n	N-n	N



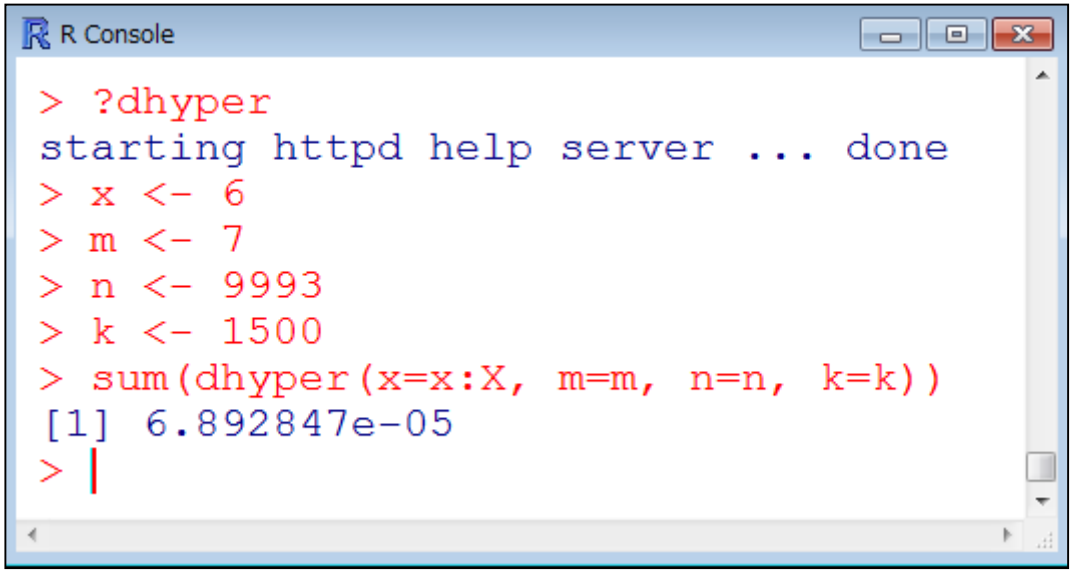
機能解析（超幾何検定）

?dhyperマニュアル中の一般的な説明に置き換えるとこんな感じです

- *m*=7個の白いボールと*n*=9993個の黒いボールが入った箱があります（トータルで*N*=*m*+*n*=10,000個）。この中から*k*=1500個ランダムに取り出したときに*x*=6個以上白いボールが含まれる確率を計算しなさい。

	白	黒	計
箱の中	1	9993−(1500−6)	8500
箱の外	6	1500−6	1500
計	7	9993	10000

	白	黒	計
箱の中	m-x	n-(k-x)	m+n-k
箱の外	x	k-x	k
計	m	n	N



機能解析(カイ二乗検定)

DEGとして1500個抽出したとき、酸化的リン酸化関連遺伝子が6個以上含まれる確率として算出

R Console

```
> N <- 10000
> n <- 7
> X <- 1500
> x <- 6
> data <- matrix(c((n-x), (N-n)-(X-x), x, (X-x)), ncol=2, byrow=T)
> data
     [,1] [,2]
[1,]    1 8499
[2,]    6 1494
> chisq.test(data)

        Pearson's Chi-squared test with Yates' continuity
        correction

data:  data
X-squared = 22.2032, df = 1, p-value = 2.453e-06

 警告メッセージ:
In chisq.test(data) :  カイ自乗近似は不正確かもしれません
>
```

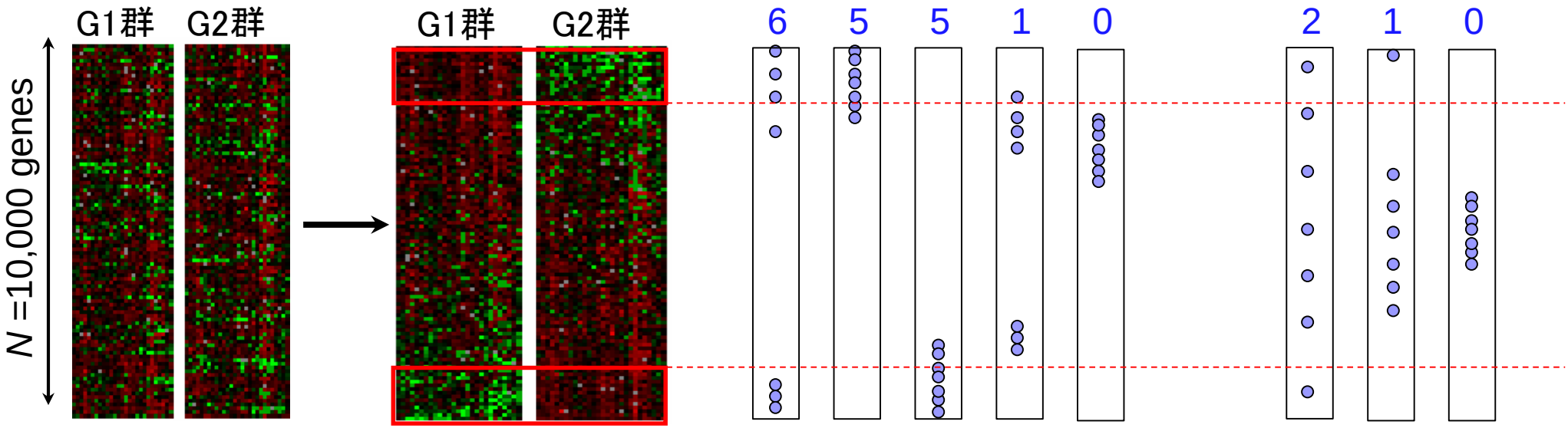
	XXX	XXX以外	計
non-DEG数	1	8500-1	8500
DEG数	6	1500-6	1500
計	7	9993	10000

	XXX	XXX以外	計
non-DEG数	n-x	$(N$-$n)$-$(X$-$x)$	N-X
DEG数	x	X-x	X
計	n	N-n	N

機能解析（遺伝子セット解析）

上位1500個のうち、酸化的リン酸化関連遺伝子が7個中4つ以上含まれていればp < 0.05で検出可能ということを意味する。

- Over-Representation Analysis (ORA)



p-value	*x*=6	*x*=5	*x*=4	*x*=3	*x*=2	*x*=1	*x*=0
超幾何検定	6.89E-05	0.0012	0.0121	0.0737	0.2834	0.6795	1.0000
カイ二乗検定	2.45E-06	0.0003	0.0095	0.1247	0.6337	0.6337	0.5603
Fisher test	6.89E-05	0.0012	0.0121	0.0737	0.2834	1.0000	0.6039

p < 0.05を灰色で示した

機能解析（遺伝子セット解析）

GenMAPPは比較的有名だと思います

- Over-Representation Analysis (ORA)
  - GenMAPP (Dahlquist et al., *Nature Genet.,* **31**: 19-20, 2002)
  - FatiGO (Al-Shahrour et al., *Bioinformatics*, **20**: 578-580, 2004)
  - GOstat (Beissbarth et al., *Bioinformatics*, **20**: 1464-1465, 2004)
  - GOFFA (Sun et al., *BMC Bioinformatics*, **7 Suppl 2**: S23, 2006)
  - agriGO (Du et al., *Nucleic Acids Res.*, **38**: W64-W70, 2010)
  - …

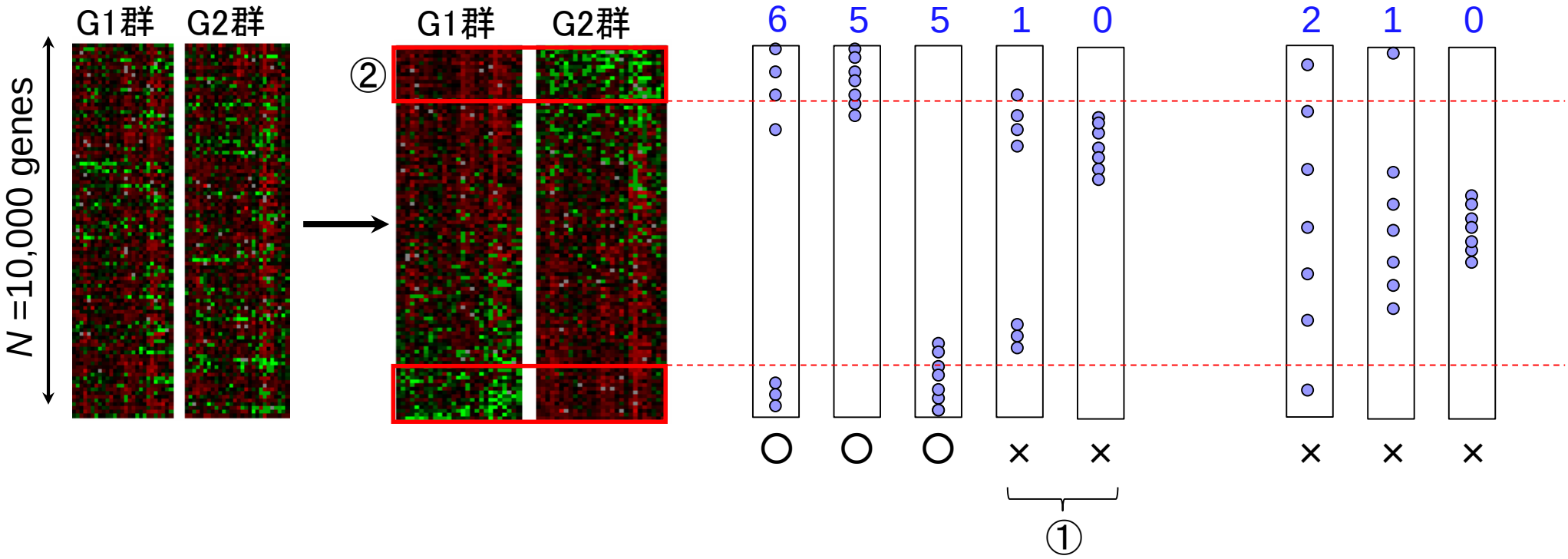
第1世代（ORA）の短所

① 全体的には動いているものの、個々の発現変動の度合いが弱い場合に検出困難
② 上位X個のX次第で結果が変わる
③ 情報量低下（発現変動の度合い → カウント情報）

もちろん分割表ベースの方法(ORA)ではない第2世代以降の方法があります。代表例はGene Set Enrichment Analysis (GSEA)。

	XXX	XXX以外	計
non-DEG数	n-x	$(N$-$n)$-$(X$-$x)$	N-X
DEG数	x	X-x	X
計	n	N-n	N

③



第2世代（FCS）

もちろん分割表ベースの方法(ORA)ではない第2世代以降の方法があります。代表例はGene Set Enrichment Analysis (GSEA)。

- Functional Class Scoring (FCS)
  1. 遺伝子ごとの統計量を算出（発現変動の度合いを数値化）
     例：*t*-統計量、log（G2/G1）、相関係数、…
  2. 目的の遺伝子セットXXX（=酸化的リン酸化関連遺伝子）の偏りを何らかの方法で評価
     - *t*検定（XXX中の遺伝子群の統計量 vs. それ以外の遺伝子群の統計量）
     - Wilcoxon rank sum test （XXX中の遺伝子群の発現変動の順位 vs. それ以外）
     - XXX中の*n*個の遺伝子群の何らかの要約統計量S_{XXX}を計算しておき、*N*個の全遺伝子の中からランダムに*n*個を抽出して同じ統計量を計算する（例えば10万回）。10万回のうちS_{XXX}「以上」（大きければ大きいほど発現変動していることを意味する場合；その逆のときは「以下」）だった回数（例えばj回）に基づいてp値（= j / 100,000）を算出（いわゆるgene set permutationというアプローチ）
     - 本来のG1群 vs. G2群のラベル情報を用いて得られたXXX中の*n*個の遺伝子群の何らかの要約統計量S_{XXX}を計算しておく。ランダムにラベル情報を入れ替えて、同じ統計量を計算することを何回も繰り返して*p*値を算出（いわゆるPhenotype permutationというアプローチ）

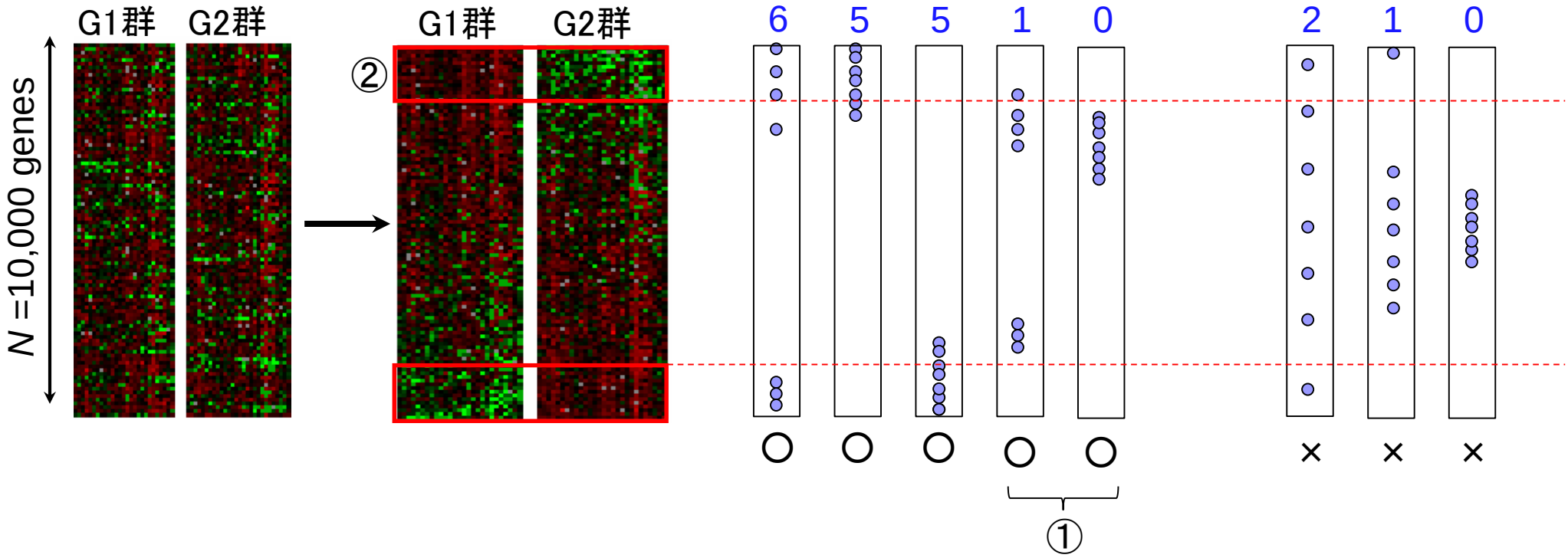
第2世代（FCS）

遺伝子ごとのlog比で考えると、遺伝子を等価に取り扱うのではなく、log比そのものを足し込むことで、発現変動の大きなものと小さなものを考慮するようなイメージ

- 第一世代(ORA)の欠点が改善
  - ① 全体的には動いているものの、個々の発現変動の度合いが弱い場合に検出困難
  - ② 上位X個のX次第で結果が変わる
  - ③ 情報量低下（発現変動の度合い → カウント情報）

	XXX	XXX以外	計
non-DEG数	n-x	$(N$-$n)$-$(X$-$x)$	N-X
DEG数	x	X-x	X
計	n	N-n	N

③



最も有名なのはGSEAです

第2世代(FCS)

- Functional Class Scoring (FCS)
  - **GSEA** (Subramanian et al., *PNAS*, **102**: 15545–15550, 2005)
  - PAGE (Kim and Volsky, *BMC Bioinformatics*, **6**: 144, 2005)
  - sigPathway (Tian et al., *PNAS*, **102**: 13544–13549, 2005)
  - GSA (Efron and Tibshirani, *Ann. Appl. Stat.*, **1**: 107–129, 2007)
  - GeneTrail (Backes et al., *Nucleic Acids Res.*, **35**: W186–W192, 2007)
  - SAM-GS (Dinu et al., *BMC Bioinformatics*, **8**: 242, 2007)
  - …

突っ込みどころは満載だが、そんなことをいってもしょうがない

遺伝子セット解析の課題

- （知識ベースの解析法なので）解析対象がアノテーションの情報の豊富な生物種に限定
  - それ以外の生物種は、まずは地道にアノテーション情報を増やしていくことが先決（ではないだろうか）
  - アノテーションの解像度を上げる努力も大事
- アノテーション情報の信頼度が高いとはいえない
  - なんらかのGO termがついていたとしても、その大部分のevidence codeが自動でつけられたもの（IEA, inferrred from electronic annotations）である…
- 遺伝子セット間の独立性の問題
  - 「数百個程度の遺伝子セットの中から、比較するサンプル間で動いている遺伝子セットはどれか？」という解析を遺伝子セット間の独立性を仮定して調べるが、そもそも独立ではない（GO term間の親子関係などから明らか）
    → いくつくらいの遺伝子セットが動いているのか？という問いに答えるすべがない
- 評価に用いられる「よく研究されているデータセット」は答えが完全に分かっているものではない（the actual biology is never fully known!）
  - “感度が高い”と謳っているだけの方法は…（全部の遺伝子セットが動いている → 感度100%）

どの遺伝子セットにどの遺伝子が所属しているかというgmt形式ファイルの取得が第一歩

遺伝子セット解析おさらい

- Gene Ontology (GO)解析(発現に差のあるGO termを探索)
  - 基本3カテゴリ(Cellular component (CC), Molecular Function (MF), Biological Process (BP))のどれでも可能
    - 例:肝臓の空腹状態 vs. 満腹状態のGO(BP)解析の結果、「脂肪酸β酸化」関連GO term (GO:0006635)が動いていることが分かった
- パスウェイ解析(発現に差のあるパスウェイを探索)
  - KEGG, BioCarta, Reactome pathway databaseのどれでも可能
    - 例:酸化的リン酸化パスウェイ関連遺伝子セットが糖尿病患者で動いていた
- モチーフ解析(発現に差のあるモチーフを探索)
  - 同じ3'-UTR microRNA結合モチーフをもつ遺伝子セット
  - 同じ転写因子結合領域(TATA-boxなど)をもつ遺伝子セット
    - 例:TATA-boxをもつ遺伝子セットがG1群 対 G2群比較で動いていた
- …

Contents

- デザイン行列の意味を理解（教科書p173–182）
  - limmaパッケージを用いた2群間比較のおさらい
  - limmaパッケージを用いた3群間比較（反復あり）
- 反復なし多群間比較（教科書p182–188）
  - limmaパッケージを用いた3群間比較（反復なし）
  - TCCパッケージ中のROKU法を用いた特異的発現遺伝子検出
- 機能解析（遺伝子セット解析）
  - 基本的な考え方
  - 前処理
    - MSigDBからの遺伝子セット情報（gmt形式ファイル）取得
    - ID変換（probe ID → gene symbol）
  - GSAパッケージを用いた遺伝子セット解析

MSigDBは、Molecular Signature Databaseの略。様々な遺伝子セット解析を行うためのgmt形式ファイルをダウンロード可能です。

MSigDB ver. 5.0

- H: hallmark gene sets (50 gene sets)
- c1: positional gene sets (326 gene sets)
  - ヒト染色体の位置ごとの遺伝子セットリストファイル (326 gene sets)
- c2: curated gene sets (4,725 gene sets)
  - **CGP**: chemical and genetic perturbations (3,395 gene sets)
  - **CP**: canonical pathways (1,330 gene sets)
  - **CP:BIOCARTA**: BioCarta gene sets (217 gene sets)
  - **CP:KEGG**: KEGG gene sets (186 gene sets)
  - **CP:REACTOME**: Reactome gene sets (674 gene sets)
- c3: motif gene sets (836 gene sets)
  - **MIR**: microRNA targets (221 gene sets)
  - **TFT**: transcription factor targets (615 gene sets)
- c4: computational gene sets (858 gene sets)
  - **CGM**: cancer gene neighborhoods (427 gene sets)
  - **CM**: cancer modules (431 gene sets)
- c5: gene ontology (GO) gene sets (1,454 gene sets)
  - **BP**: biological process (825 gene sets)
  - **CC**: cellular component (233 gene sets)
  - **MF**: molecular function (396 gene sets)
- c6: oncogenic signatures gene sets (189 gene sets)
- c7: immunologic signatures gene sets (1,910 gene sets)

発現変動と関連するKEGGパスウェイを調べたいとき ← CP:KEGG

発現変動と関連するBP中のGO termsを調べたいとき ← BP

MSigDB ver. 4.0

2015年4月にver. 4.0から5.0になったようです。劇的な違いはないようです。

- c1: positional gene sets (326 gene sets)
  - ヒト染色体の位置ごとの遺伝子セットリストファイル (326 gene sets)
- c2: curated gene sets (4,722 gene sets)
  - **CGP**: chemical and genetic perturbations (3,402 gene sets)
  - **CP**: canonical pathways (1,320 gene sets)
  - **CP:BIOCARTA**: BioCarta gene sets (217 gene sets)
  - **CP:KEGG**: KEGG gene sets (186 gene sets)
  - **CP:REACTOME**: Reactome gene sets (674 gene sets)
- c3: motif gene sets (836 gene sets)
  - **MIR**: microRNA targets (221 gene sets)
  - **TFT**: transcription factor targets (615 gene sets)
- c4: computational gene sets (858 gene sets)
  - **CGM**: cancer gene neighborhoods (427 gene sets)
  - **CM**: cancer modules (431 gene sets)
- c5: gene ontology (GO) gene sets (1,454 gene sets)
  - **BP**: biological process (825 gene sets)
  - **CC**: cellular component (233 gene sets)
  - **MF**: molecular function (396 gene sets)
- c6: oncogenic signatures gene sets (189 gene sets)
- c7: immunologic signatures gene sets (1,910 gene sets)

発現変動と関連するKEGGパスウェイを調べたいとき ← CP:KEGG

発現変動と関連するBP中のGO termsを調べたいとき ← BP

MSigDB

遺伝子セット解析を行うためのgmt形式ファイルのダウンロード方法はこちら

(Rで)マイクロアレイデータ解析

(last modified 2015/05/25, since 2005)

- 解析 | 発現変動 | 時系列 | non-periodic genes | maSigPro (Conesa 2006) (last modified 2009/8/3)
- 解析 | 発現変動 | 時系列 | non-periodic genes | SAM (Tushe 2001) (last modified 2009/8/3)
- 解析 | 機能解析 | 遺伝子オントロジー(GO)解析 | について ① modified 2015/06/07) NEW
- 解析 | 機能解析 | 遺伝子オントロジー(GO)解析 | GAGE (Lu 2009)(last modified 2014/06/01)
- 解析 | 機能解析 | 遺伝子オントロジー(GO)解析 | GSA (Efron 2007)(last modified 2014/06/03)推奨

解析 | 機能解析 | 遺伝子オントロジー(GO)解析 | について NEW

機能解析の実体は遺伝子セット解析です。遺伝子セット解析としてGO termを利用するのがGO解析です。

R用:

- globalt
- SAFE:
- topGO
- pcot2:
- Catego
- GSA (
- dCoxS
- GAGE
- GOSen
- Camer
- RamiG
- LCT:

遺伝子セットDB系:

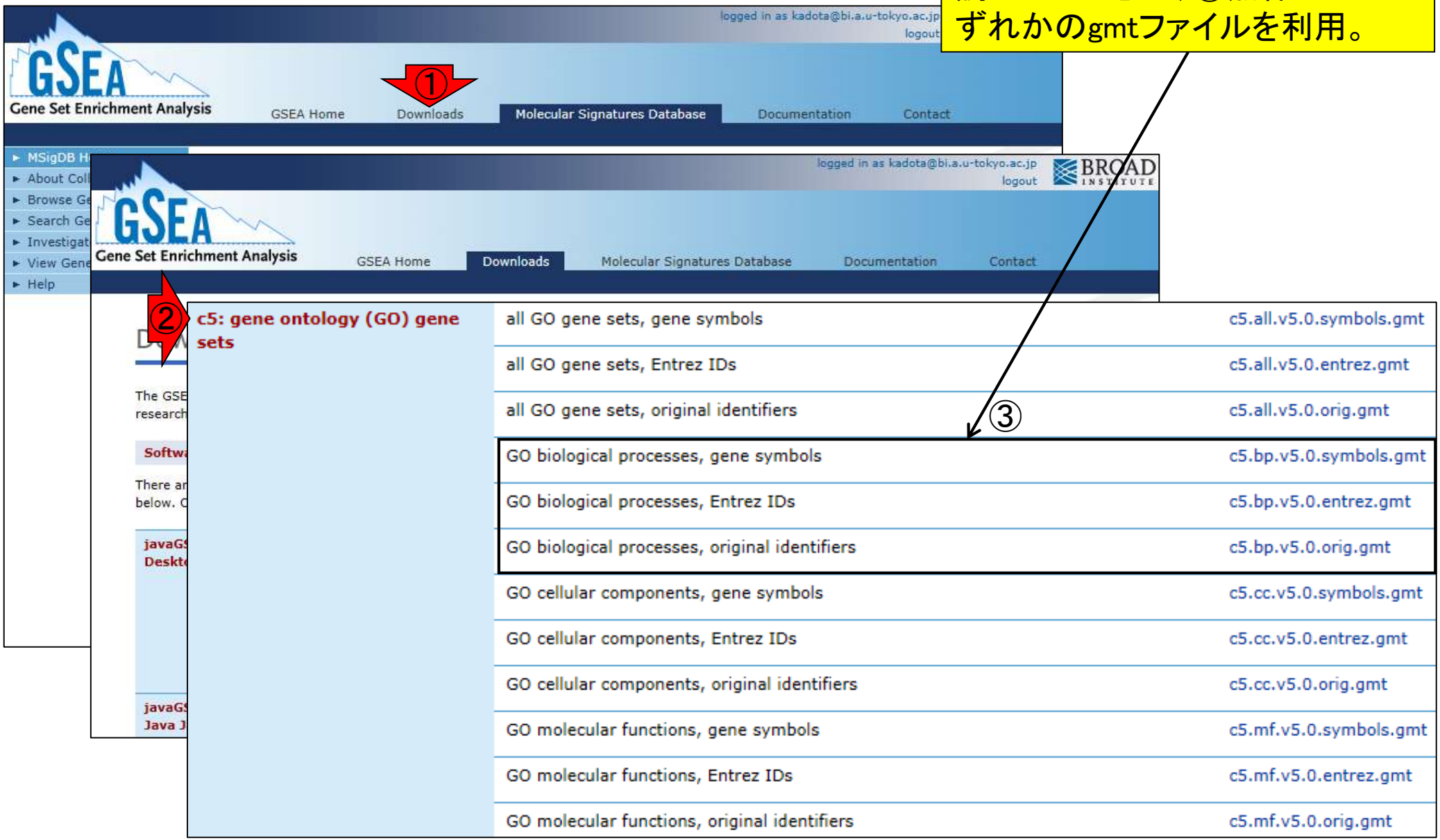
- MSigDB: Subramanian et al., PNAS, 2005

GSEAに代表される発現変動遺伝子セット解析は、基本的にGSEAの開発者らが作成した様々な遺伝子セット情報を収めた Molecular Signatures Database (MSigDB)からダウンロードした.gmt形式ファイルを読み込んで解析を行います。それゆえ、自分がどの遺伝子セットについて機能解析を行いたいのかを予め決めておく必要がありますが、GO解析の場合はbiological process (BP)が一般的なようです。2015/06/07現在のバージョンは5.0です。gmt形式ファイルの基本的なダウンロード方法は以下の通りです:

1. Molecular Signatures Database (MSigDB) ②
   「register」のページで登録し、遺伝子セットをダウンロード可能な状態にする。
2. Molecular Signatures Database (MSigDB)の
   「Download gene sets」の"Download"のところをクリックし、Loginページで登録したe-mail addressを入力。
3. これでMSigDBのダウンロードページに行けるので、目的に応じたgmtファイルをダウンロードしておく。
   「c5: gene ontology gene sets」の「bp: biological process」を解析する場合:c5.bp.v5.0.symbols.gmt
   「c5: gene ontology gene sets」の「cc: cellular components」を解析する場合:c5.cc.v5.0.symbols.gmt
   「c5: gene ontology gene sets」の「mf: molecular functions」を解析する場合:c5.mf.v5.0.symbols.gmt

MSigDB

②発現変動と関連するbiological processes (BP)中のGO termsを調べたいときは、③黒枠内のいずれかのgmtファイルを利用。



gmtファイル

基本的にどれを使っても自由だが、利用するRパッケージがどの入力形式を受け付けるかにも依存する。経験上gene symbolsを使っておけば間違いないので、門田は*.symbols.gmtをいつも利用しています。

	A	B	C	D				
1	TRNA_PROCESSING	http://wwv	ADAT1	TRNT1	FAR			
2	REGULATION_OF_BIOLOGICAL_QUALI	http://wwv	DLC1	ALS2	SLC			
3	DNA_METABOLIC_PROCESS	http://wwv	XRCC5	XRCC4	RAD51C	XRCC3	XRCC2	XRC
4	AMINO_SUGAR_METABOLIC_PROCESS	http://wwv	UAP1	CHIA	GNPDA1	GNE	CSGALNAC	CHS
5	BIOPOLYMER_CATABOLIC_PROCESS	http://wwv	BTRC	HNRNPD	USE1	RNASEH1	RNF217	ISG2
6	RNA_METABOLIC_PROCESS	http://wwv	HNRNPF	HNRNPD	SYNCRIP	MED24	PQBP	MED

	A	B	C	D	E	F	G	
1	TRNA_PROCESSING	http://wwv	23536	51095	10667	4234	6301	
2	REGULATION_OF_BIOLOGICAL_QUALI	http://wwv	10395	57679	84679	5743	5742	
3	DNA_METABOLIC_PROCESS	http://wwv	7520	7518	5889	7517	7516	
4	AMINO_SUGAR_METABOLIC_PROCESS	http://wwv	6675	27159	10007	10020	55790	
5	BIOPOLYMER_CATABOLIC_PROCESS	http://wwv	8945	3184	55850	246243	154214	
6	RNA_METABOLIC_PROCESS	http://wwv	3185	3184	10492	9862	6096	

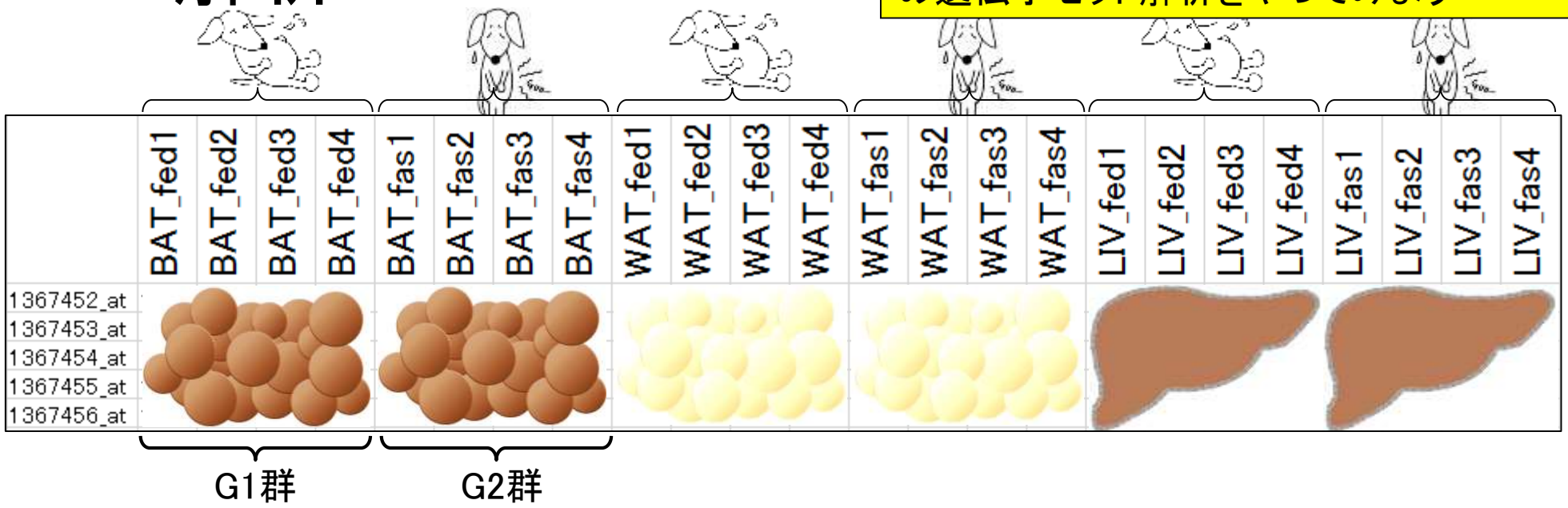
c5.bp.v5.0.symbols.gmt
c5.bp.v5.0.entrez.gmt
c5.bp.v5.0.orig.gmt

	A	B	C	D	E	F	G	
1	TRNA_PROCESSING	http://wwv	ADAT1	TRNT1	FARS2	METTL1	SARS	AARS
2	REGULATION_OF_BIOLOGICAL_QUALI	http://wwv	DLC1	ALS2	SLC9A7	PTGS2	PTGS1	MPV
3	DNA_METABOLIC_PROCESS	http://wwv	XRCC5	XRCC4	RAD51C	XRCC3	XRCC2	XRC
4	AMINO_SUGAR_METABOLIC_PROCESS	http://wwv	UAP1	CHIA	GNPDA1	GI		
5	BIOPOLYMER_CATABOLIC_PROCESS	http://wwv	BTRC	HNRPD	USE1	RI		
6	RNA_METABOLIC_PROCESS	http://wwv	HNRPF	HNRPD	SYNCRIP	M		

1列目：遺伝子セット名
2列目：URL
3列目以降：gene ID or symbol

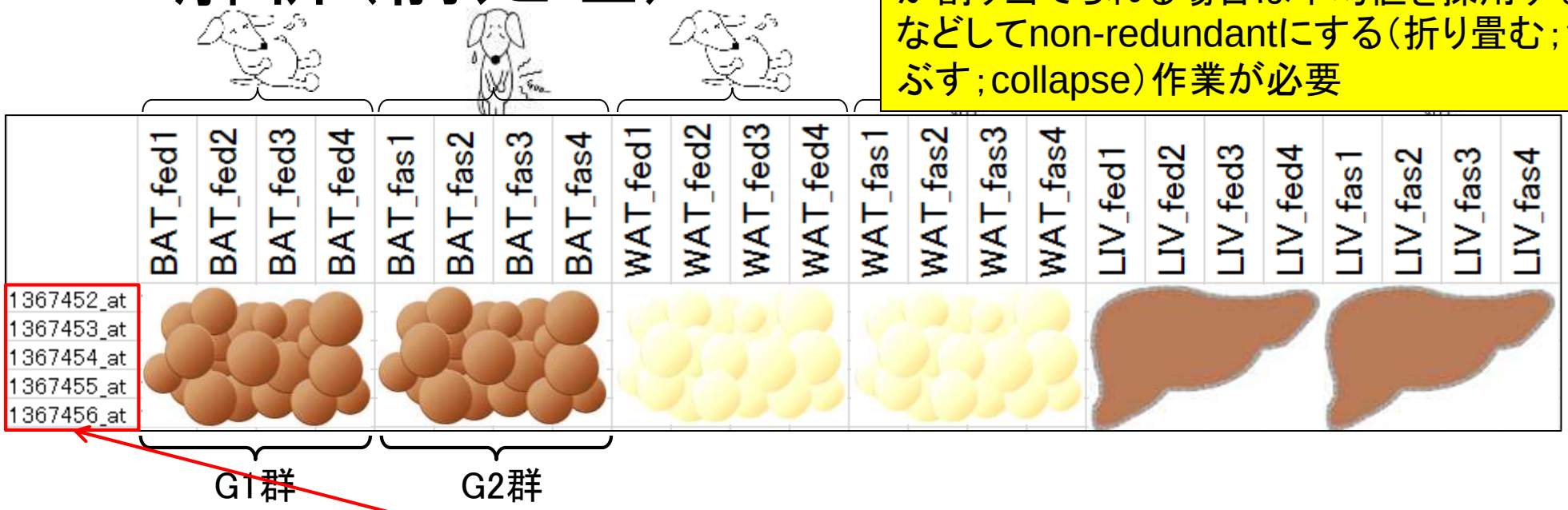
GO解析

GSE7623 (Nakai et al., 2008)の**対数変換後**のデータを入力として、BAT_fed vs. BAT_fasの遺伝子セット解析をやってみよう



GO解析（前処理）

プローブIDとgene symbolの対応付けを行い、同じgene symbolに複数のプローブIDが割り当てられる場合は平均値を採用するなどしてnon-redundantにする（折り畳む；つぶす；collapse）作業が必要



	A	B	C	D	E	F	G	
1	TRNA_PROCESSING	http://www	ADAT1	TRNT1	FARS2	METTL1	SARS	AARS
2	REGULATION_OF_BIOLOGICAL_QUALI	http://www	DLC1	ALS2	SLC9A7	PTGS2	PTGS1	MPV
3	DNA_METABOLIC_PROCESS	http://www	XRCC5	XRCC4	RAD51C	XRCC3	XRCC2	XRC
4	AMINO_SUGAR_METABOLIC_PROCESS	http://www	UAP1	CHIA	GNPDA1	GNE	CSGALNAC	CHS
5	BIOPOLYMER_CATABOLIC_PROCESS	http://www	BTRC	HNRNPD	USE1	RNASEH1	RNF217	ISG2
6	DNA_METABOLIC_PROCESS	http://www	HNRNPF	HNRNPD	SYNCRIP	MED24	PORR	MED

Contents

- デザイン行列の意味を理解（教科書p173–182）
  - limmaパッケージを用いた2群間比較のおさらい
  - limmaパッケージを用いた3群間比較（反復あり）
- 反復なし多群間比較（教科書p182–188）
  - limmaパッケージを用いた3群間比較（反復なし）
  - TCCパッケージ中のROKU法を用いた特異的発現遺伝子検出
- 機能解析（遺伝子セット解析）
  - 基本的な考え方
  - 前処理
    - MSigDBからの遺伝子セット情報（gmt形式ファイル）取得
    - ID変換（probe ID → gene symbol）
  - GSAパッケージを用いた遺伝子セット解析

教科書p70-71

ID変換

遺伝子発現データは、公共DBのGEOからGSE7623というIDで取得したものだった。ここから、プローブIDとgene symbolの対応付けを行うためのアノテーションファイルを取得可能

- イントロ | 発現データ取得 | GEOquery(Davis_2007) (last modified 2013/08/20)
- イントロ | アノテーション情報取得 | 公共DB(GEO)から (last modified 2013/08/18)
- イントロ | アノテーション情報取得 | GEOquery(Davis_2007) (last modified 2014/06/03)推
- イントロ | アノテーション情報取得 | Rのパッケージ*.dbから (last modified 2014/06/02)
- イントロ | プローブ配列情報取得 | Rのパッケージから (last modified 2013/08/16)
- イントロ | トランスクリプ
- イントロ | トランスクリプ
- イントロ | Affymetrix CE
- 正規化(cDNA or two-c
- 正規化 | Stanford型 (or
- 正規化 | Stanford型 (or
- 正規化 | Agilent microa
- 正規化 | Agilent microa
- 正規化 | Illumina Bead
- 正規化 | Illumina Bead

① ②

イントロ | アノテーション情報取得 | GEOquery (Davis_2007)

公共DBGene Expression Omnibus (GEO)に登録されているアレイ(Platform)のアノテーション情報をGEOqueryというRパッケージを用いてゲットするやり方を示します。

「ファイル」-「ディレクトリ

1. Affymetrix Rat Genome

```
out_f <- "hoge1.txt
param <- "GPL1355"

#必要なパッケージをロ
library(GEOquery)

#前処理
data <- getGEO(para

#本番
out <- data@dataTab
write.table(out, ou
```

3. "GSE"から始まるIDをたよりにアレイのID情報を内部的に入手してアノテーション情報を取得したい場合:

GSE7623 (Nakai et al., BBB, 2008)は1種類のアレイ(GPL1355)しか使っていないので、1つのファイルのみ生成されます。出力ファイルの情報に相当するoutオブジェクト中の、自分がほしいprobe ID列とgene symbol列が1列目と11列目に存在することがあらかじめ分かっているという前提です。colnames(out)でわかります。アノテーション情報のバージョンが異なりうるため若干違うかもしれませんがhoge3_GPL1355.txtと酷似したものができていると思います。

```
param1 <- "GSE7623"                          #入手したいGEO IDを指定
param2 <- "hoge3_"                           #出力ファイル名の最初の部分を指定
param_posi <- c(1, 11)                       #outオブジェクト中のID列とgene symb

#必要なパッケージをロード
library(GEOquery)                            #パッケージの読み込み

#前処理
data <- getGEO(param1)                       #指定したGEO IDのデータを取得
sapply(data, annotation)                     #用いられたアレイ情報(GPL ID)を表示

#本番
hoge <- sapply(data, annotation)             #用いられたアレイ情報(GPL ID)をhoge
for(i in 1:length(hoge)){                    #hogeの要素数(用いられたアレイ数)分
    out_f <- paste(param2, hoge[i], ".txt", sep="")#出力ファイル名を作成し
    out <- data[[i]]@featureData@data        #アノテーション情報抽出結果をoutに格
    colnames(out)                            #確認してるだけです
```

教科書p70-71

ID変換

プローブIDとgene symbolからなるアノテーションファイルを取得できています。確認時は2分程度で終わりましたが、hogeフォルダにhoge3_GPL1355.txtを一応置いてあります。

3. "GSE"から始まるIDをたよりにアレイのID情報を内部的に入手してアノテーション情報を取得したい場合:

GSE7623 (Nakai et al., BBB, 2008)は1種類のアレイ(GPL1355)しか使っていないので、1つのファイルのみ生成されます。出力ファイルの情報に相当するoutオブジェクト中の、自分がほしいprobe ID列とgene symbol列が1列目と11列目に存在することがあらかじめ分かっているという前提です。colnames(out)でわかります。アノテーション情報のバージョンが異なりうるため若干違うかもしれませんがhoge3_GPL1355.txtと酷似したものができていると思います。

```
param1 <- "GSE7623"                          #入手したいGE
param2 <- "hoge3_"                           #出力ファイル
param_posi <- c(1, 11)                       #outオブジェ

#必要なパッケージをロード
library(GEOquery)                            #パッケージの

#前処理
data <- getGEO(param1)                       #指定したGEO
sapply(data, annotation)                     #用いられたア

#本番
hoge <- sapply(data, annotation)             #用いられたア
for(i in 1:length(hoge)){                    #hogeの要素数
    out_f <- paste(param2, hoge[i], ".txt", sep="")
    out <- data[[i]]@featureData@data        #アノテーショ
    colnames(out)                            #確認してるた
```

R Console

```
> for(i in 1:length(hoge)){                          #hogeの要$
+     out_f <- paste(param2, hoge[i], ".txt", sep=$
+     out <- data[[i]]@featureData@data  #アノテ$
+     colnames(out)                                  #確認して$
+     out <- out[,param_posi]                        #サブセッ$
+     write.table(out, out_f, sep="\t", append=F, $
+ }
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files(pattern="hoge3")
[1] "hoge3_GPL1355.txt"
> head(out, n=3)
                ID Gene Symbol
1367452_at 1367452_at       Sumo2
1367453_at 1367453_at       Cdc37
1367454_at 1367454_at       Copb2
> dim(out)
[1] 31099     2
>
```

ID変換

エクセルで開くときには注意が必要！1行1列目のところが”ID”から始まる文字列の場合にこのような現象が起こるようですが、基本無視で構いません。

参考

ID変換

BMC Bioinformatics. 2004 Jun 23;5:80.

Mistaken identifiers: gene name errors can be introduced inadvertently when using Excel in bioinformatics.

Zeeberg BR[1], Riss J, Kane DW, Bussey KJ, Uchio E, Linehan WM, Barrett JC, Weinstein JN.

Author information

Abstract

BACKGROUND: When processing microarray data sets, we recently noticed that some gene names were being changed inadvertently to non-gene names.

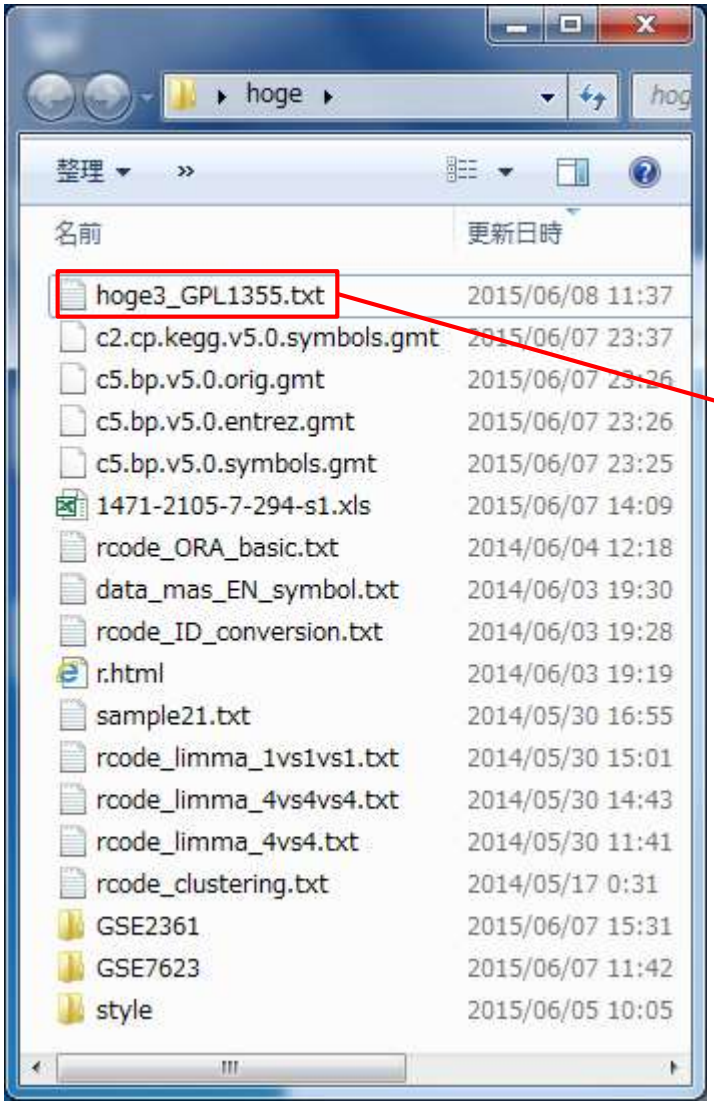
RESULTS: A little detective work traced the problem to default date format conversions and floating-point format conversions in the very useful Excel program package. The date conversions affect at least 30 gene names; the floating-point conversions affect at least 2,000 if Riken identifiers are included. These conversions are irreversible; the original gene names cannot be recovered.

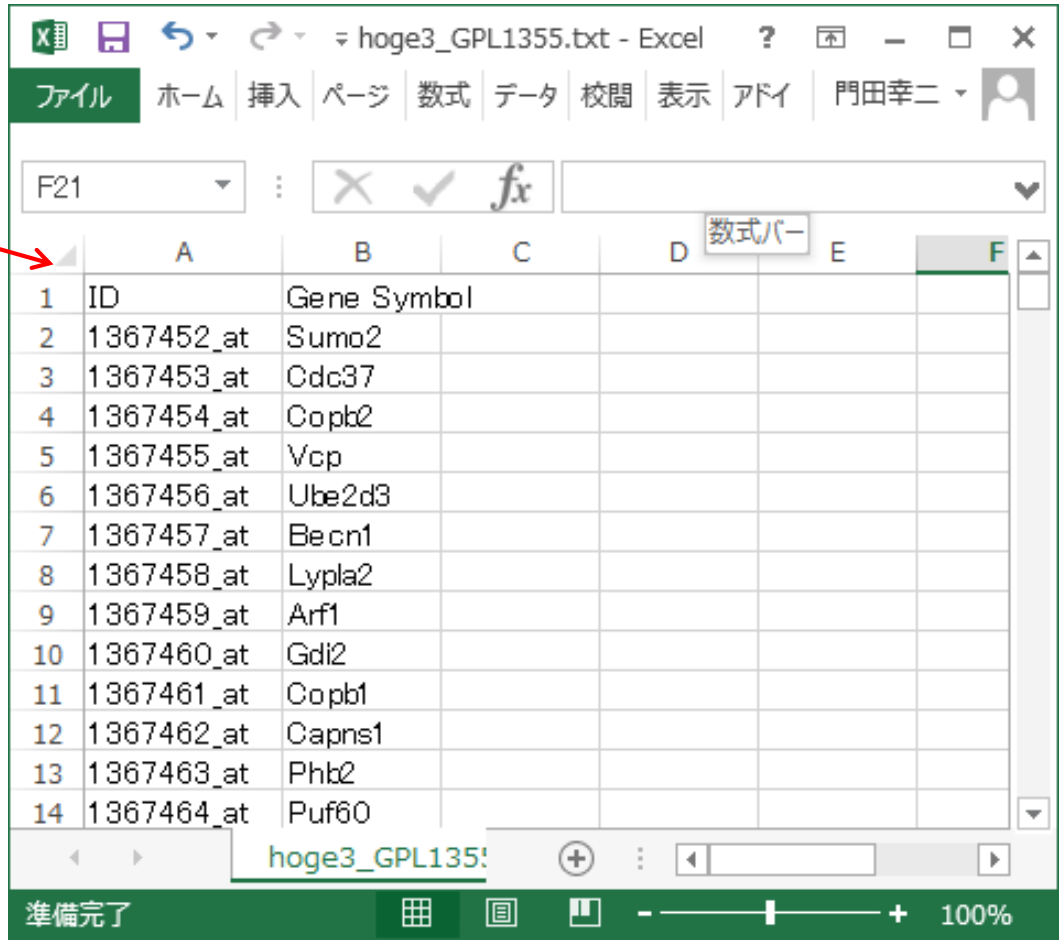
CONCLUSIONS: Users of Excel for analyses involving gene names should be aware of this problem, which can cause genes, including medically important ones, to be lost from view and which has contaminated even carefully curated public databases. We provide work-arounds and scripts for circumventing the problem.

編集して保存したい場合には、ドラッグ&ドロップで開いてはだめです。「ファイル」–「開く」でファイルを指定して開くべし!そのまま開くと例えば**March2**というgene symbolが日付と認識されてしまうため、これを防ぐ必要があります！

ID変換

ここでは、ファイルの中身を眺めるだけなので、再度ドラッグ&ドロップ。1回目は失敗しても2回目は普通に開けます。





	A	B
1	ID	Gene Symbol
2	1367452_at	Sumo2
3	1367453_at	Cdc37
4	1367454_at	Copb2
5	1367455_at	Vcp
6	1367456_at	Ube2d3
7	1367457_at	Becn1
8	1367458_at	Lypla2
9	1367459_at	Arf1
10	1367460_at	Gdi2
11	1367461_at	Copb1
12	1367462_at	Capns1
13	1367463_at	Phb2
14	1367464_at	Puf60

ID変換

Gene Symbol列で
ソートしてみると…

	A	B	C	D	E	F	G	H	I
1	ID	Gene Symbol		BAT_fed1	BAT_fed2	BAT_fed3	BAT_fed4	BAT_fas1	BAT_fas2
2	1367452_at	Sumo2	1367452_at	12.78446	12.44708	12.80591	12.30472	12.58943	12.6075
3	1367453_at	Cdc37	1367453_at	11.80125	12.15293	11.94223	11.96848	11.84538	11.6817
4	1367454_at	Copb2	1367454_at	11.3899	11.16076	11.14599	11.21209	11.54065	11.3088
5	1367455_at	Vcp	1367455_at	12.36435	12.52974	12.43257	12.60401	12.44199	12.2499
6	1367456_at	Ube2d3	1367456_at	13.44849	13.54305	13.55279	13.6298	13.36913	13.2442
7	1367457_at	Becn1	1367457_at	10.40403	10.69632	10.47508	10.45579	10.14192	10.2906
8	1367458_at	Lypla2	1367458_at	9.925339	10.24454	9.972001	9.957607	8.702884	9.35787
9	1367459_at	Arf1	1367459_at	13.83374	13.71342	13.95477	13.70214	13.76626	13.6696
10	1367460_at	Gdi2	1367460_at	13.36349	13.55406	13.48064	13.43393	13.5366	13.5084

hoge3_GPL1355.txt

data_mas_EN.txt

ID変換

同じgene symbolを持つプローブIDが複数存在することがわかる

	A	B	C	D	E	F	G	H	I
1	ID	Gene Symbol		BAT_fed1	BAT_fed2	BAT_fed3	BAT_fed4	BAT_fas1	BAT_fas2
2	1367452_at	Sumo2	1367452_at	12.78446	12.44708	12.80591	12.30472	12.58943	12.6075
3	1367453_at	Cdc37	1367453_at	11.80125	12.15293	11.94223	11.96848	11.84538	11.6817
4	1367454_at	Copb2	1367454_at	11.3899	11.16076	11.14599	11.21209	11.54065	11.3088
5	1367455_at	Vcp	1367455_at	12.36435	12.52974	12.43257	12.60401	12.44199	12.2499
6	1367456_at	Ube2d3	1367456_at	13.44849	13.54305	13.55279	13.6298	13.36913	13.2442
7	1367457_at	Becn1	1367457_at	10.40403	10.69632	10.47508	10.45579	10.14192	10.2906
8	1367458_at	Lypla2	1367458_at	9.925339	10.24454	9.972001	9.957607	8.702884	9.35787
9	1367459_at	Arf1	1367459_at	13.83374	13.71342	13.95477	13.70214	13.76626	13.6696
10	1367460_at	Gdi2	1367460_at	13.36349	13.55406	13.48064	13.43393	13.5366	13.5084

ID	Gene Symbol		BAT_fed1	BAT_fed2	BAT_fed3	BAT_fed4	BAT_fas1	BAT_fas2	BAT_fas3	BAT_fas4	WAT_fe
1369509_a_at	A1bg	1369509_a_at	2.283274	2.663464	3.393108	5.581337	4.444774	2.070498	2.366629	1.242299	3.202
1368784_at	A1cf	1368784_at	5.844694	5.913832	5.755455	5.509977	4.184778	6.572447	5.657811	5.381227	5.246
1370027_a_at	A1i3 /// Mug1	1370027_a_at	7.186455	6.369903	3.668318	3.498324	7.211794	6.427345	2.480638	3.215938	5.959
1378371_at	A2bp1	1378371_at	3.395028	5.712191	5.063096	6.231098	6.612678	6.064106	6.833413	6.143807	4.328
1380516_at	A2bp1	1380516_at	4.359652	4.69428	3.010153	4.315149	4.516653	2.632701	3.892409	5.18021	1.917
1383130_at	A2bp1	1383130_at	6.081817	4.593572	4.535909	8.811696	9.702333	9.75088	9.105133	8.825036	5.409
1383257_at	A2bp1	1383257_at	1.501018	2.966239	2.232853	7.283267	8.426518	8.651317	7.625359	7.31545	1.329
1385235_at	A2bp1	1385235_at	3.847942	4.026388	3.911792	8.330072	9.3733	9.137933	8.885068	8.734736	4.634
1390594_at	A2bp1	1390594_at	4.641244	6.627334	5.234572	5.89079	2.245885	1.417343	6.138234	6.252581	5.294
1382945_at	A2ld1	1382945_at	9.974381	10.2494	9.811327	9.831842	9.093722	8.947993	8.82087	9.13077	9.386
1392725_at	A2ld1	1392725_at	6.696741	5.425293	6.610669	6.104502	6.084417	6.684742	5.49265	2.381073	6.770

ID変換

マイクロアレイごとに搭載されている遺伝子の種類や重複度が異なるため、この作業は重要。

入力1：hoge3_GPL1355.txt

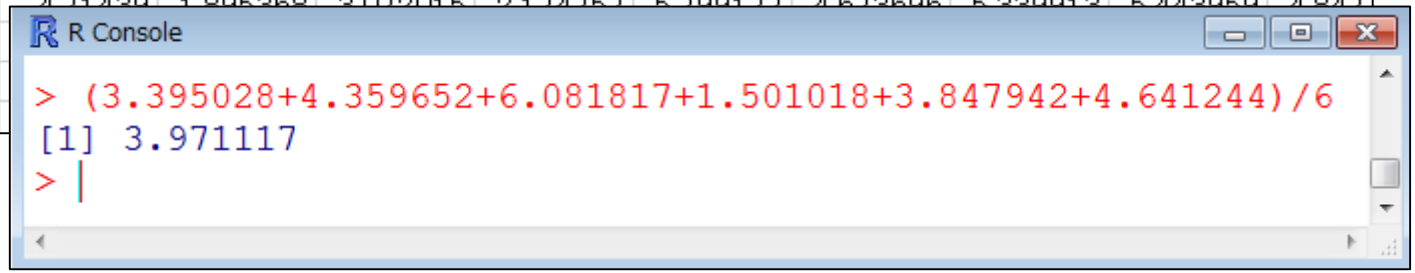
ID	Gene Symbol
1369509_a_at	A1 bg
1368784_at	A1 cf
1370027_a_at	A1 i3 /// Mug
1378371_at	A2 bp1
1380516_at	A2 bp1
1383130_at	A2 bp1
1383257_at	A2 bp1
1385235_at	A2 bp1
1390594_at	A2 bp1
1382945_at	A2 ld1
1392725_at	A2 ld1

入力2：data_mas_EN.txt

	BAT_fed1	BAT_fed2	BAT_fed3	BAT_fed4	BAT_fas1	BAT_fas2	BAT_fas3	BAT_fas4	WAT_fe
1369509_a_at	2.283274	2.663464	3.393108	5.581337	4.444774	2.070498	2.366629	1.242299	3.202
1368784_at	5.844694	5.913832	5.755455	5.509977	4.184778	6.572447	5.657811	5.381227	5.246
1370027_a_at	7.186455	6.369903	3.668318	3.498324	7.211794	6.427345	2.480638	3.215938	5.959
1378371_at	3.395028	5.712191	5.063096	6.231098	6.612678	6.064106	6.833413	6.143807	4.328
1380516_at	4.359652	4.69428	3.010153	4.315149	4.516653	2.632701	3.892409	5.18021	1.917
1383130_at	6.081817	4.593572	4.535909	8.811696	9.702333	9.75088	9.105133	8.825036	5.409
1383257_at	1.501018	2.966239	2.232853	7.283267	8.426518	8.651317	7.625359	7.31545	1.329
1385235_at	3.847942	4.026388	3.911792	8.330072	9.3733	9.137933	8.885068	8.734736	4.634
1390594_at	4.641244	6.627334	5.234572	5.89079	2.245885	1.417343	6.138234	6.252581	5.294
1382945_at	9.974381	10.2494	9.811327	9.831842	9.093722	8.947993	8.82087	9.13077	9.386
1392725_at	6.696741	5.425293	6.610669	6.104502	6.084417	6.684742	5.49265	2.381073	6.770

出力：data_mas_EN_symbol.txt

	BAT_fed1	BAT_fed2	BAT_fed3	BAT_fed4	BAT_fas1	BAT_fas2	BAT_fas3	BAT_fas4	WAT_fe
A1 bg	2.283274	2.663464	3.393108	5.581337	4.444774	2.070498	2.366629	1.242299	3.202
A1 cf	5.844694	5.913832	5.755455	5.509977	4.184778	6.572447	5.657811	5.381227	5.246
A1 i3 /// Mug1	7.186455	6.369903	3.668318	3.498324	7.211794	6.427345	2.480638	3.215938	5.959
A2 bp1	3.971117	4.770001	3.998062	6.810345	6.812894	6.275713	7.079936	7.075304	3.81
A2 ld1	8.335561	7.837348	8.210998	7.968172	7.58907	7.816367	7.15676	5.755922	8.07
A2 m	4.201252	5.612429	4.637289	3.967974	3.373227	3.301394	4.6048	3.135392	8.089
A3 galt2	5.848709	7.467876	6.437417	6.415588	5.585956	6.751774	7.105526	7.278448	7.548
A4 galt	4.71439	1.895368	3.072015	2.174757	5.799177	4.673696	5.339913	5.443959	4.847
AA926063 ///									
Aaas									
Aacs									



ID変換

2つの入力ファイル（発現データと変換表）から1つの出力ファイルが得られます。

(Rで)マイクロアレイデータ解析

(last modified 2015/05/25, since 2005)

What's ne

- 門田幸
  する最
  んでい
  トーム解
- お知ら
  や講演

- 前処理 | フィルタリング | 分散が小さいものを除去 (last modified 2013/11/15)
- 前処理 | ID変換 | について(last modified 2014/06/03)
- 前処理 | ID変換 | probe ID --> gene symbol(last modified 2014/06/03)
- 前処理 | ID変換 | probe ID --> Entrez ID(last modified 2014/06/03)
- 前処理 | ID変換 | probe ID --> その
- 前処理 | ID変換 | 同じ遺伝子名を
- 解析 | 基礎 | 共通遺伝子の抽出(l
- 解析 | 基礎 | ベクトル間の距離(las
- 解析 | 基礎 | 遺伝子ごとの各種要
- 解析 | 基礎 | 最大発現量を示す組
- 解析 | 基礎 | 似た発現パターンを
- 解析 | 基礎 | 平均-分散プロット (la
- 解析 | クラスタリング | 階層的 | に
- 解析 | クラスタリング | 階層的 | pvc
- 解析 | クラスタリング | 階層的 | hcl

前処理 | ID変換 | probe ID --> gene symbol NEW

probe IDの遺伝子発現行列を入力として、gene symbolの遺伝子発現行列を出力するやり方を示します。probe IDとgene symbolの対応関係情報が必要ですので、様々なやり方を示しています。同じgene symbolをもつ複数のprobe IDsが存在する場合にはparamで指定した方法で要約統計量を計算します。代表値(要約統計量)は、平均値(mean)、中央値(median)、最大値(max)など好きなものを指定できます。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し、以下をコピペ

1. サンプルデータ20の31,099 probesets×24 samplesのRMA-preprocessedデータ(data_rma_2.txt)の場合:

Affymetrix Rat Genome 230 2.0 Arrayを用いて得られたNakai et al., BBB, 2008のデータです。イントロ | アノテーション情報取得 | GEOquery(Davis_2007)の3を行って得られたhoge3_GPL1355.txtの情報も利用しています。
data_rma_2.txtの1列目のID情報とhoge3_GPL1355.txtの1列目のID情報の対応がとれる(同じ行の位置でなくてもよい)ことが前提です。1分程度で終わります。

```
in_f1 <- "data_rma_2.txt"                #入力ファイル名を指定してin_f1に格納(発現データ)
in_f2 <- "hoge3_GPL1355.txt"             #入力ファイル名を指定してin_f2に格納(Gene symbolと
out_f <- "hoge1.txt"                     #出力ファイル名を指定してout_fに格納
param <- mean                            #代表値を指定

#前処理(IDとGene symbolとの対応関係を含む情報を入手)
sym <- read.table(in_f2, header=TRUE, row.names=1, sep="\t", quote="")#in_f2で指定したフ
IDs <- as.vector(sym[,1])                #Gene symbol情報をベクトルに変換し、IDsに格納
names(IDs) <- rownames(sym)              #IDsを行名で対応づけられるようにしている
uniqID <- unique(IDs)                    #non-redundant ID情報を抽出し、uniqIDに格納
uniqID <- uniqID[uniqID != ""]           #uniqIDの中から指定したIDがないものを除く
uniqID <- uniqID[!is.na(uniqID)]         #uniqIDの中から指定したIDが"NA"のものを除く
uniqID <- uniqID[!is.nan(uniqID)]        #uniqIDの中から指定したIDが"NaN"のものを除く

#本番
```

ID変換

hogeフォルダ中のdata_mas_EN_symbol.txtは、このコードのコピペで作成しています。作業ディレクトリに入力ファイルがあることを確認してから実行しましょう。

前処理 | ID変換 | probe ID --> gene symbol NEW

probe IDの遺伝子発現行列を入力として、gene symbolの遺伝子発現行列を出力するやり方を示します。probe IDとgene symbolの対応関係情報が必要ですので、様々なやり方を示しています。同じgene symbolをもつ複数のprobe IDsが存在する場合にはparamで指定した方法で要約統計量を計算します。代表値(要約統計量)は、平均値(mean)、中央値(median)、最大値(max)など好きなものを指定できます。

「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し、以下をコピペ

1. サンプルデータ20の31,099 probesets×24 samplesの

Affymetrix Rat Genome 230 2.0 Arrayを用いて得られ
ン情報取得 | GEOquery(Davis_2007)の3を行って得
data_rma_2.txtの1列目のID情報とhoge3_GPL1355.t
い)ことが前提です。1分程度で終わります。

```
in_f1 <- "data_rma_2.txt"
in_f2 <- "hoge3_GPL1355.txt"
out_f <- "hoge1.txt"
param <- mean

#前処理(IDとGene symbolとの対応関係を含む
sym <- read.table(in_f2, header=TRUE, r
IDs <- as.vector(sym[,1])
names(IDs) <- rownames(sym)
uniqID <- unique(IDs)
uniqID <- uniqID[uniqID != ""]
uniqID <- uniqID[!is.na(uniqID)]
uniqID <- uniqID[!is.nan(uniqID)]

#本番
```

rcode_ID_conversion.txt

```
in_f1 <- "data_mas_EN.txt"          #入力ファイル名を指定してin_f1に格納(発現
in_f2 <- "hoge3_GPL1355.txt"        #入力ファイル名を指定してin_f2に格納(Gene
out_f <- "data_mas_EN_symbol.txt"   #出力ファイル名を指定してout_fに格納
param <- mean                       #代表値を指定

#前処理(IDとGene symbolとの対応関係を含む情報を入手)
sym <- read.table(in_f2, header=TRUE, row.names=1, sep="¥t", quote="")#in_f2で指
IDs <- as.vector(sym[,1])           #Gene symbol情報をベクトルに変換し、IDsに
names(IDs) <- rownames(sym)         #IDsを行名で対応づけられるようにしている
uniqID <- unique(IDs)               #non-redundant ID情報を抽出し、uniqIDに格
uniqID <- uniqID[uniqID != ""]      #uniqIDの中から指定したIDがないものを除く
uniqID <- uniqID[!is.na(uniqID)]    #uniqIDの中から指定したIDが"NA"のものを除
uniqID <- uniqID[

#本番
data <- read.tabl
hoge <- t(apply(a
                a
        }, data,
rownames(hoge) <-

#ファイルに保存
tmp <- cbind(rown
write.table(tmp, out_f, sep="¥t", append=F, quote=F, row.names=F)#tmpの中身を指
```

R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/GSE7623"
> list.files(pattern="hoge")
[1] "hoge1.txt"          "hoge3_GPL1355.txt"
> list.files(pattern="mas_EN")
[1] "data_mas_EN.txt"
>
```

ID変換

hogeフォルダ中のdata_mas_EN_symbol.txtは、14,132個のユニークなgene symbolsからなることがわかります。

rcode_ID_conversion.txt

```
in_f1 <- "data_mas_EN.txt"              #入力ファイル名を指定してin_f1に格納(発現
in_f2 <- "hoge3_GPL1355.txt"            #入力ファイル名を指定してin_f2に格納(Gene
out_f <- "data_mas_EN_symbol.txt"       #出力ファイル名を指定してout_fに格納
param <- mean                           #代表値を指定

#前処理(IDとGene symbolとの対応関係を含む情報を入手)
sym <- read.table(in_f2, header=TRUE, row.names=1, sep=
IDs <- as.vector(sym[,1])               #Gene symbol情報
names(IDs) <- rownames(sym)             #IDsを行名で対応
uniqID <- unique(IDs)                   #non-redundant ID
uniqID <- uniqID[uniqID != ""]          #uniqIDの中から指
uniqID <- uniqID[!is.na(uniqID)]        #uniqIDの中から指
uniqID <- uniqID[!is.nan(uniqID)]       #uniqIDの中から指

#本番
data <- read.table(in_f1, header=TRUE, row.names=1, sep=
hoge <- t(apply(as.matrix(uniqID), 1, function(i, d = da
                apply(d[which(s == i), ], 2, p, na.rm =
        }, data, IDs, param))           #apply関数でdata
rownames(hoge) <- uniqID                #non-redundant ID

#ファイルに保存
tmp <- cbind(rownames(hoge), hoge)      #指定したIDの列を
write.table(tmp, out_f, sep="¥t", append=F, quote=F, row
```

R Console

```
> #ファイルに保存
> tmp <- cbind(rownames(hoge), hoge)      #指定$
> write.table(tmp, out_f, sep="\t", append=F, q$
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/GSE7623"
> list.files(pattern="mas_EN")
[1] "data_mas_EN.txt"
[2] "data_mas_EN_symbol.txt"
> dim(tmp)
[1] 14132    25
> tmp[1:3, 1:4]
                BAT_fed1       BAT_fed2
Sumo2 "Sumo2" "12.7844634"  "12.44708219"
Cdc37 "Cdc37" "11.80124704" "12.15293493"
Copb2 "Copb2" "11.38990178" "11.16075717"
      BAT_fed3
Sumo2 "12.80590758"
Cdc37 "11.94222741"
Copb2 "11.14598707"
>
```

Tips：as.matrix

プログラムの組み方で速度が結構違います（データフレーム形式より行列形式のほうが早いらしい）。孫建強氏作は1分、門田作は2分（爆）

1. サンプルデータ20の31,099 probesets×24 samplesのRMA-preprocessedデータ(data_rma_2.txt)の場合

Affymetrix Rat Genome 230 2.0 Arrayを用いて得られたNakai et al., BBB, 2008のデータです。イントロ | アノテーション情報取得 | GEOquery(Davis_2007)の3を行って得られたhoge3_GPL1355.txtの情報も利用しています。data_rma_2.txtの1列目のID情報とhoge3_GPL1355.txtの1列目のID情報の対応がとれる(同じ行の位置でなくてもよい)ことが前提です。1分程度で終わります。

```
in_f1 <- "data_rma_2.txt"                    #入力ファイル名を指定してin_f1に格納(発現データ)
in_f2 <- "hoge3_GPL1355.txt"
out_f <- "hoge1.txt"
param <- mean

#前処理(IDとGene symbolとの対応関係を含
sym <- read.table(in_f2, header=TRUE,
IDs <- as.vector(sym[,1])
names(IDs) <- rownames(sym)
uniqID <- unique(IDs)
uniqID <- uniqID[uniqID != ""]
uniqID <- uniqID[!is.na(uniqID)]
uniqID <- uniqID[!is.nan(uniqID)]

#本番
data <- read.table(in_f1, header=TRUE
hoge <- t(apply(as.matrix(uniqID), 1,
                apply(d[which(s == i)
        }, data, IDs, param))
rownames(hoge) <- uniqID

#ファイルに保存
```

2. サンプルデータ20の31,099 probesets×24 samplesのRMA-preprocessedデータ(data_rma_2.txt)の場合:

Affymetrix Rat Genome 230 2.0 Arrayを用いて得られたNakai et al., BBB, 2008のデータです。イントロ | アノテーション情報取得 | GEOquery(Davis_2007)の3を行って得られたhoge3_GPL1355.txtの情報も利用しています。data_rma_2.txtの1列目のID情報とhoge3_GPL1355.txtの1列目のID情報の対応がとれる(同じ行の位置でなくてもよい)ことが前提です。2分程度で終わります。

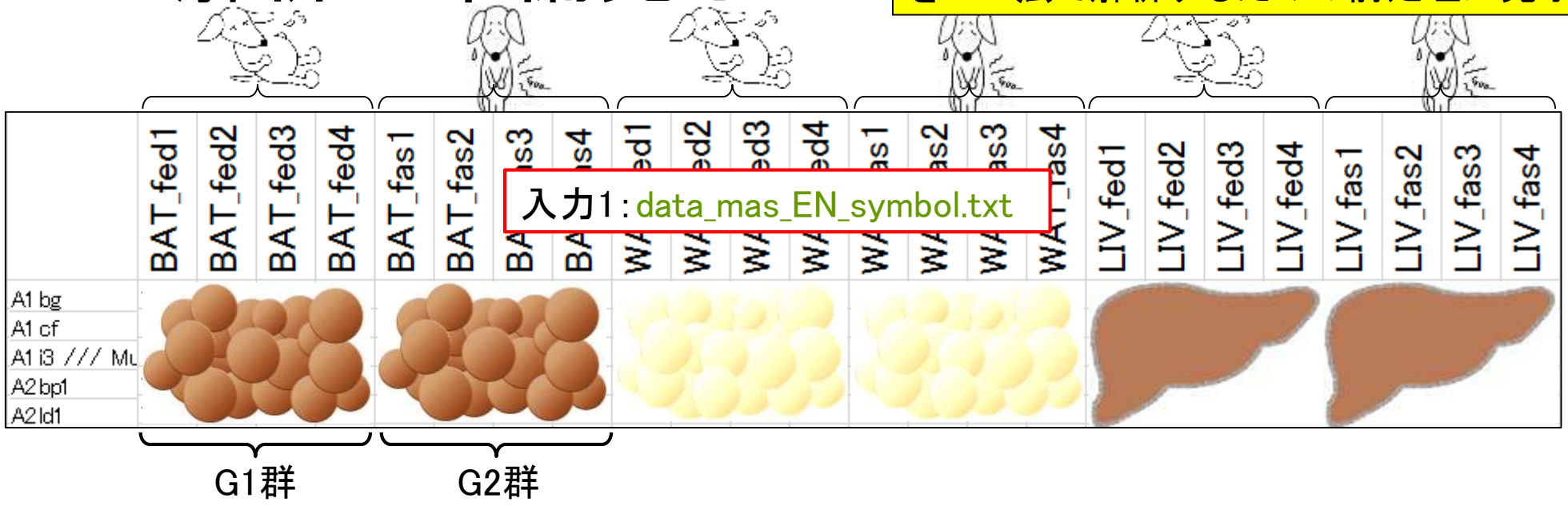
```
in_f1 <- "data_rma_2.txt"                    #入力ファイル名を指定してin_f1に格納(発現データ)
in_f2 <- "hoge3_GPL1355.txt"                 #入力ファイル名を指定してin_f2に格納(Gene symbol
out_f <- "hoge2.txt"                         #出力ファイル名を指定してout_fに格納
param <- mean                                #代表値を指定

#前処理(IDとGene symbolとの対応関係を含む情報を入手)
sym <- read.table(in_f2, header=TRUE, row.names=1, sep="\t", quote="")#in_f2で指定した
IDs <- as.vector(sym[,1])                    #Gene symbol情報をベクトルに変換し、IDsに格納
names(IDs) <- rownames(sym)                  #IDsを行名で対応づけられるようにしている
uniqID <- unique(IDs)                        #non-redundant ID情報を抽出し、uniqIDに格納
uniqID <- uniqID[uniqID != ""]               #uniqIDの中から指定したIDがないものを除く
uniqID <- uniqID[!is.na(uniqID)]             #uniqIDの中から指定したIDが"NA"のものを除く
uniqID <- uniqID[!is.nan(uniqID)]            #uniqIDの中から指定したIDが"NaN"のものを除く

#本番
data <- read.table(in_f1, header=TRUE, row.names=1, sep="\t", quote="")#in_f1で指定し
hoge <- NULL                                 #最終的に欲しい情報を格納するためのプレースホルダ
for(i in 1:length(uniqID)){                  #non-redundant ID数分だけループを回す
    hoge <- rbind(hoge, apply(data[which(IDs == uniqID[i]),], 2, param, na.rm=TRUE))#
}                                            #non-redundant ID数分だけループを回す
rownames(hoge) <- uniqID                     #non-redundant IDをhogeの行の名前として利用
```

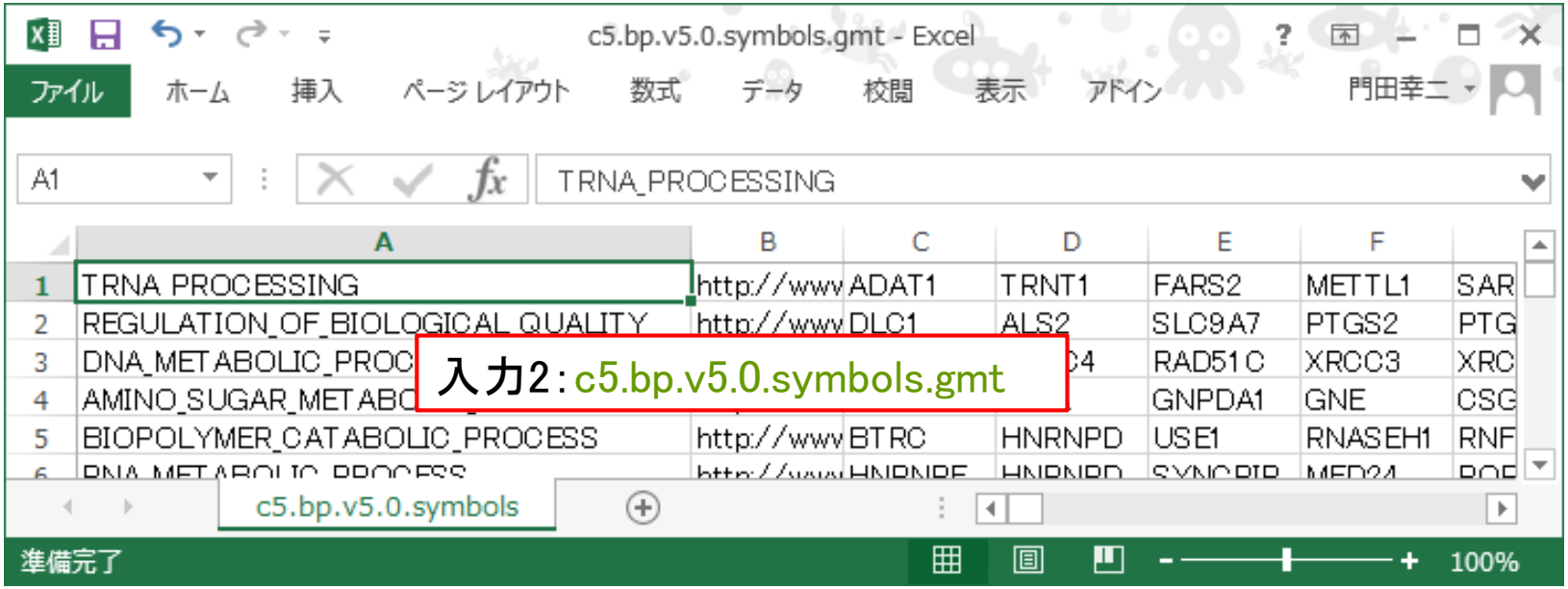
Contents

- デザイン行列の意味を理解（教科書p173–182）
  - limmaパッケージを用いた2群間比較のおさらい
  - limmaパッケージを用いた3群間比較（反復あり）
- 反復なし多群間比較（教科書p182–188）
  - limmaパッケージを用いた3群間比較（反復なし）
  - TCCパッケージ中のROKU法を用いた特異的発現遺伝子検出
- 機能解析（遺伝子セット解析）
  - 基本的な考え方
  - 前処理
    - MSigDBからの遺伝子セット情報（gmt形式ファイル）取得
    - ID変換（probe ID → gene symbol）
  - GSAパッケージを用いた遺伝子セット解析

GO解析の準備完了

褐色脂肪「満腹 vs. 空腹」の発現変動に関連したGO Biological Process遺伝子セットをGSA法で解析するための前処理が完了





GSAでGO解析

data_mas_EN_symbol.txtを入力としてBAT_fed vs. BAT_fasのGO解析をやってみよう。

(Rで)マイクロアレイデータ解析

(last modified

What's new?

- 門田幸二
- お知らせは

- 解析 | 発現変動 | 時系列 | non-periodic genes | SAM (Tusher_2001) (last modified 2009/8/3)
- 解析 | 機能解析 | 遺伝子オントロジー(GO)解析 |について(last modified 2015/06/07) NEW
- 解析 | 機能解析 | 遺伝子オントロジー(GO)解析 | GAGE (Luo_2009)(last modified 2014/06/01)
- 解析 | 機能解析 | 遺伝子オントロジー(GO)解析 | GSA (Efron_2007)(last modified 2014/06/03)推奨
- 解析 | 機能解析 | 遺伝子オントロジー(GO)解析 | Category (Jiang_2007)(last modified 2014/06/01)

①

解析 | 機能解析 | 遺伝子オントロジー(GO)解析 | GSA (Efron_2007) NEW

GSAパッケー

「ファイル」-「

②

1.サンプルデータ

肝臓のみからな

ルのprobeset ID

```
in_f1 <- "da
in_f2 <- "c5
out_f1 <- "h
out_f2 <- "h
param_G1 <-
param_G2 <-
param_FDR <-

#必要なパッケ
library(GSA)

#入力ファイル
gmt <- GSA.r
```

3.サンプルデータ20の data_rma_2_nr.txtの場合:

14,132 genes×24 samplesのデータです。オリジナルのprobeset IDからgene symbolにID変換がなされています。
BAT_fed vs. BAT_fastedの2群間比較を行うべく、それらの 位置情報を指定しています。

```
in_f1 <- "data_rma_2_nr.txt"          #入力ファイル名を指定してin_f1に格納(発現データ
in_f2 <- "c5.bp.v5.0.symbols.gmt"     #入力ファイル名を指定してin_f2に格納(gmtファイ
out_f1 <- "hoge3_G1.txt"              #出力ファイル名を指定してout_f1に格納(G1群で高
out_f2 <- "hoge3_G2.txt"              #出力ファイル名を指定してout_f2に格納(G2群で高
param_G1 <- 4                         #G1群のサンプル数を指定
param_G2 <- 4                         #G2群のサンプル数を指定
param_FDR <- 0.1                      #DEG検出時のfalse discovery rate (FDR)閾値を
param_posi <- c(1:4, 5:8)             #G1群およびG2群の位置情報を指定

#必要なパッケージをロード
library(GSA)

#入力ファイルの読み込みとラベル情報の
gmt <- GSA.read.gmt(in_f2)
data <- read.table(in_f1, header=
rownames(data) <- toupper(rownames
data.cl <- c(rep(1, param_G1), re
data <- data[,param_posi]
colnames(data)

#GSA本番
out <- GSA(data, data.cl,             #GSAを実行し、結果をoutに格納
```

R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files(pattern="symbol")
[1] "c2.cp.kegg.v5.0.symbols.gmt"
[2] "c5.bp.v5.0.symbols.gmt"
[3] "data_mas_EN_symbol.txt"
>
```

GSAでGO解析

FDR 10%の閾値を満たす有意な遺伝子セット数はG1群で高発現のものが24個、G2群で高発現のものが4個だったことがわかる。ヒトによって若干結果が異なります。

3.サンプルデータ20のdata_rma_2_nr.txtの場合:

14,132 genes×24 samplesのデータです。オリジナルのprobeset IDからgene symbolにID変換がなされていま
BAT_fed vs. BAT_fastedの2群間比較を行うべく、それらの位置情報を指定しています。

```
in_f1 <- "data_rma_2_nr.txt"            #入力ファイル名を指定してin_f1に格納(発現データ
in_f2 <- "c5.bp.v5.0.symbols.gmt"       #入力ファイル名を指定してin_f2に格納(gmtファイ
out_f1 <- "hoge3_G1.txt"                #出力ファイル名を指定してout_f1に格納(G1群で高
out_f2 <- "hoge3_G2.txt"                #出力ファイル名を指定して
param_G1 <- 4                           #G1群のサンプル数を指定
param_G2 <- 4                           #G2群のサンプル数を指定
param_FDR <- 0.1                        #DEG検出時のfalse disco
param_posi <- c(1:4, 5:8)               #G1群およびG2群の位置情

#必要なパッケージをロード
library(GSA)                            #パッケージの読み込み

#入力ファイルの読み込みとラベル情報の作成、そしてサブセットの作成
gmt <- GSA.read.gmt(in_f2)              #in_f2で指定したファイル
data <- read.table(in_f1, header=TRUE, row.names=1, sep="\t",
rownames(data) <- toupper(rownames(data))#IDを大文字に変換して
data.cl <- c(rep(1, param_G1), rep(2, param_G2))#G1群を1、G2群
data <- data[,param_posi]               #サブセットを抽出
colnames(data)                          #確認してるだけです

#GSA本番
out <- GSA(data, data.cl,               #GSAを実行し、結果をout
```

R Console

```
+          FDRcut=param_FDR)                    #$
>
> #ファイルに保存
> write.table(tmp$negative, out_f1, sep="\$
> write.table(tmp$positive, out_f2, sep="\$
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files(pattern="symbol")
[1] "c2.cp.kegg.v5.0.symbols.gmt"
[2] "c5.bp.v5.0.symbols.gmt"
[3] "data_mas_EN_symbol.txt"
> list.files(pattern="hoge3")
[1] "hoge3_G1.txt"       "hoge3_G2.txt"
[3] "hoge3_GPL1355.txt"
> nrow(tmp$negative)
[1] 24
> nrow(tmp$positive)
[1] 4
>
```

私は、結果の評価はできません

GSAでGO解析

Gene_set	Gene_set_name	Score	p-value	FDR
22	PHOSPHOLIPID_BIOSYNTHETIC_PROCESS	-0.5692	0	0
56	REGULATION_OF_DNA_BINDING	-0.3102	0	0
92	COFACTOR_METABOLIC_PROCESS	-0.4346	0	0
134	BIOSYNTHETIC_PROCESS	-0.2019	0	0
147	INTRACELLULAR_PROTEIN_TRANSPORT	-0.1555	0	0
205	PROTEIN_IMPORT	-0.2265	0	0
264	MEMBRANE_LIPID_BIOSYNTHETIC_PROCESS	-0.4554	0	0
287	LIPID_METABOLIC_PROCESS	-0.2399	0	0
317	LIPID_BIOSYNTHETIC_PROCESS	-0.7518	0	0
331	REGULATION_OF_TRANSCRIPTION_FACTOR_ACTIVIT	-0.295	0	0
370	MITOCHONDRION_ORGANIZATION_AND_BIOGENESIS	-0.3251	0	0
383	NEGATIVE_REGULATION_OF_DNA_BINDING	-0.6935	0	0
415	NEGATIVE_REGULATION_OF_CELLULAR_COMPONENT	-0.2769	0	0
466	CELLULAR_RESPONSE_TO_STIMULUS	-0.4359	0	0
471	INTERACTION_WITH_HOST	-0.389	0	0
490	RIBONUCLEOTIDE_METABOLIC_PROCESS	-0.6475	0	0
541	CELLULAR_LIPID_METABOLIC_PROCESS	-0.2414	0	0
607	NEGATIVE_REGULATION_OF_BINDING	-0.6935	0	0
609	PHOSPHOLIPID_METABOLIC_PROCESS	-0.2709	0	0
619	REGULATION_OF_CYTOSKELETON_ORGANIZATION_A	-0.2051	0	0
667	NEGATIVE_REGULATION_OF_TRANSCRIPTION_FACTC	-0.804	0	0
669	RIBOSOME_BIOGENESIS_AND_ASSEMBLY	-0.3904	0	0
721	PROTEIN_TRANSPORT	-0.1425	0	0
762	STEROID_BIOSYNTHETIC_PROCESS	-1.2465	0	0

Gene_set	Gene_set_name	Score	p-value	FDR
39	POSITIVE_REGULATION_OF_SIGNAL_TRANSDUCTION	0.1897	0	0
121	PATTERN_SPECIFICATION_PROCESS	0.4294	0	0
461	PROTEIN_KINASE_CASCADE	0.1781	0	0
783	ACTIVATION_OF_PROTEIN_KINASE_ACTIVITY	0.6607	0	0