Rでゲノム・トランスクリプトーム解析

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ユニット
門田 幸二（かどた　こうじ）
http://www.iu.a.u-tokyo.ac.jp/~kadota/
kadota@iu.a.u-tokyo.ac.jp

自己紹介

- 1995年3月
  - 高知工業高等専門学校・工業化学科　卒業
- 1997年3月
  - 東京農工大学・工学部・物質生物工学科　卒業
- 1999年3月
  - 東京農工大学・大学院工学研究科・物質生物工学専攻　修士課程修了
- 2002年3月
  - 東京大学・大学院農学生命科学研究科・応用生命工学専攻　博士課程修了
  - 学位論文：「cDNAマイクロアレイを用いた遺伝子発現解析手法の開発」
    （指導教官：清水謙多郎教授）
- 2002/4/1~
  - **産総研・生命情報科学研究センター（CBRC）** 産総研特別研究員
- 2003/11/1~
  - 放医研・先端遺伝子発現研究センター　研究員
- 2005/2/16~
  - 東京大学・大学院農学生命科学研究科
    特任助手→…

参考URL

(Rで)塩基配列解析(主にNGSやRNA-seq解析)

(last modified 2014/02/21, since 2010)

What's new?

- 2014年3月17-19日に九州大学にて、ワークショップ(よく分かる次世代シークエンサー解析～最先端トランスクリプトーム解析～)が開催されます。私は3日目(3/19, 13:00-16:30)を担当します。興味ある方はどうぞ。締切は確か2/21です。(2014/02/17) NEW
- 項目名の整理を行っています。3C (Hi-C)やBS-seq周辺についても少し言及してあります。(2014/02/08) NEW
- 一連の解析パイプライン(RNA-seqデータ取得 -> マッピング -> カウントデータやRPKMデータ取得 -> サンプル間クラスタリングや発現変動解析およびM-A plot描画まで)をアップデートしました。項目名の一番下のほうです。(2013/10/19)
- 発現変動解析用Rパッケージ TCC (ver. 1.2.0; Sun et al., BMC Bioinformatics, 2013)がBioconductorよりリリースされました。最新版を利用したい方は、R (ver. 3.0.2)をインストールしたのち、Bioconductor (ver. 2.13)をインストールしてください。(2013/10/17)
- どのブラウザからでもエラーなく見られる(W3C validation)ように((Rで)マイクロアレイデータ解析も含めて)リニューアルしました。(2013/07/30)
- 2013年7月29日まで公開していた以前の「(Rで)塩基配列解析」のウェブページや関連ファイルはRdeennki.zipからダウンロード可能です(110MB程度)。(2013/07/30)

自前PCでやる場合はここを参考にして必要なパッケージを予めインストールしておかねばなりません。（数時間程度かかります）

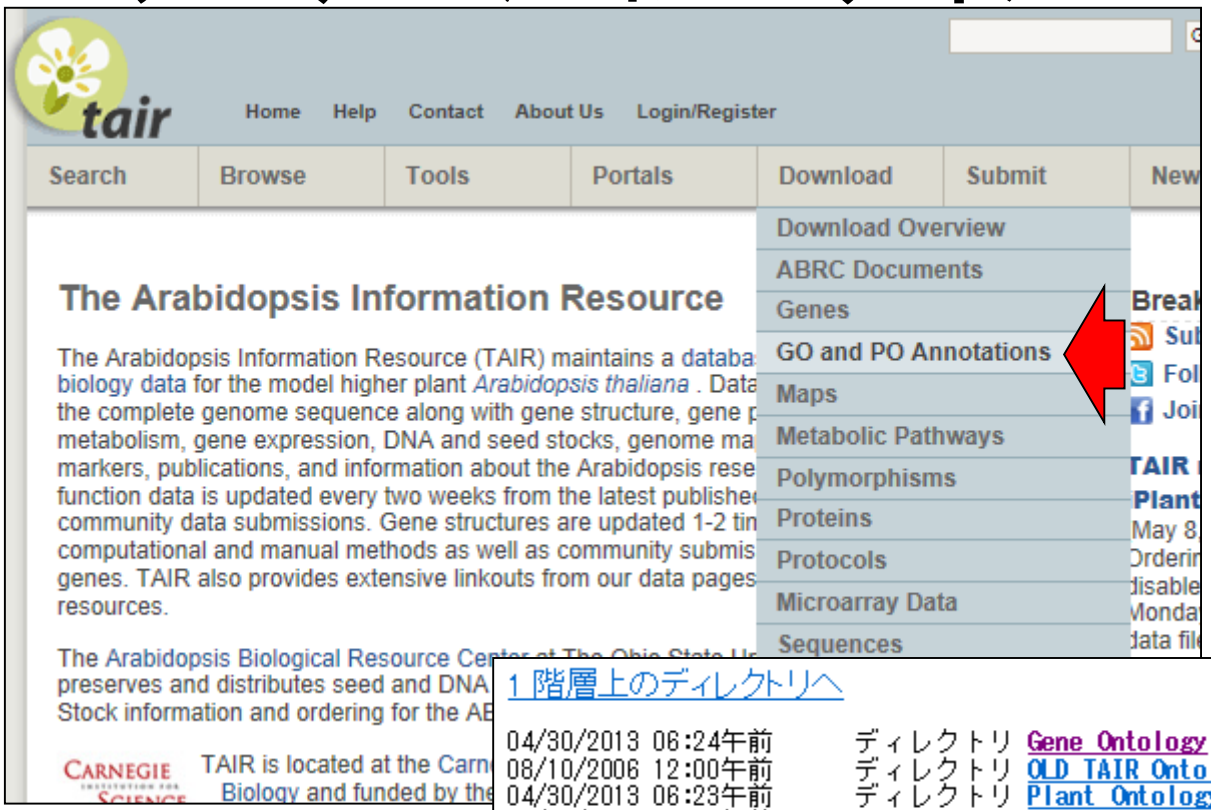
- はじめに(last modified 20
- Rのインストールと起動
- サンプルデータ(last modi
- イントロ | 一般 | ランダムに
- イントロ | 一般 | 任意の文字列を行の最初に挿入(last modified 2013/10/10)
- イントロ | 一般 | 任意のキーワードを含む行を抽出(基礎)(last modified 2014/02/06) NEW
- イントロ | 一般 | ランダムな塩基配列を生成(last modified 2013/09/29)
- イントロ | 一般 | 任意の長さの可能な全ての塩基配列を作成(last modified 2013/06/14)
- イントロ | 一般 | 任意の位置の塩基を置換(last modified 2013/09/12)
- イントロ | 一般 | 指定した範囲の配列を取得(last modified 2014/02/07) NEW
- イントロ | 一般 | 翻訳配列(translate)を取得(last modified 2013/06/14)

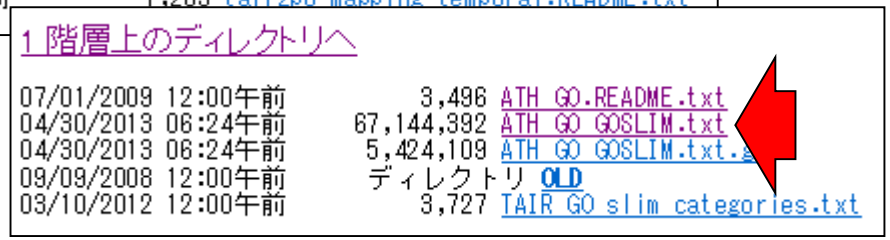
トップページへ

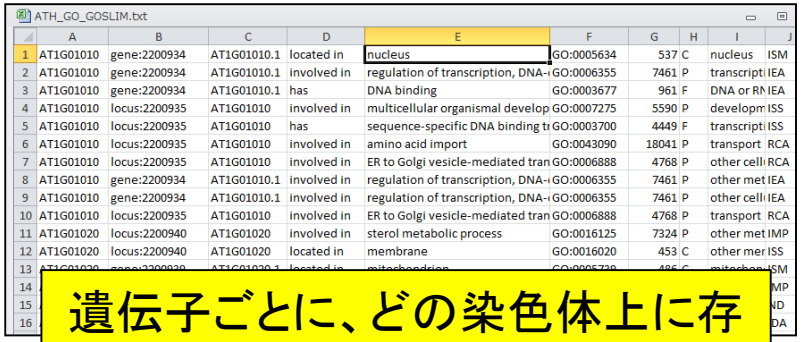
Contents（Rで…）

- ゲノム解析
  - **アノテーションファイルを読み込んで目的のキーワードを含む行のみ抽出**
  - multi-FASTAファイルを自在に解析
    - 配列長分布、GC含量、フィルタリング、部分配列の切り出しなど
    - 連続塩基の出現頻度（CpG）解析、ゲノム配列取得など
- トランスクリプトーム解析
  - 研究目的別留意点：サンプル内とサンプル間の違い
  - マッピング → カウント情報取得
  - データを眺める：クラスタリングやM-A plot
  - 理想的な実験デザイン
  - なぜx倍発現変動という議論がだめなんですか？
  - モデルとか分布って、自分の解析結果にどういう影響を与えているの？
  - 多重比較問題：FDRって何？

アノテーションファイル?!







遺伝子ごとに、どの染色体上に存在するのかやどんなGene Ontology IDが割り当てられているのかなどの情報を含むファイル

アノテーションファイルからの情報抽出

ATH_GO_GOSLIM.txt

	A	B	C	D	E	F	G	H	I	J
1	AT1G01010	gene:2200934	AT1G01010.1	located in	nucleus	GO:0005634	537	C	nucleus	ISM
2	AT1G01010	gene:2200934	AT1G01010.1	involved in	regulation of transcription, DNA-	GO:0006355	7461	P	transcripti	IEA
3	AT1G01010	gene:2200934	AT1G01010.1	has	DNA binding	GO:0003677	961	F	DNA or RN	IEA
4	AT1G01010	locus:2200935	AT1G01010	involved in	multicellular organismal develop	GO:0007275	5590	P	developm	ISS
5	AT1G01010	locus:2200935	AT1G01010	has	sequence-specific DNA binding tr	GO:0003700	4449	F	transcripti	ISS
6	AT1G01010	locus:2200935	AT1G01010	involved in	amino acid import	GO:0043090	18041	P	transport	RCA
7	AT1G01010	locus:2200935	AT1G01010	involved in	ER to Golgi vesicle-mediated tran	GO:0006888	4768	P	other cell	RCA
8	AT1G01010	gene:2200934	AT1G01010.1	involved in	regulation of transcription, DNA-	GO:0006355	7461	P	other met	IEA
9	AT1G01010	gene:2200934	AT1G01010.1	involved in	regulation of transcription, DNA-	GO:0006355	7461	P	other cell	IEA
10	AT1G01010	locus:2200935	AT1G01010	involved in	ER to Golgi vesicle-mediated tran	GO:0006888	4768	P	transport	RCA
11	AT1G01020	locus:2200940	AT1G01020	involved in	sterol metabolic process	GO:0016125	7324	P	other met	IMP
12	AT1G01020	locus:2200940	AT1G01020	located in	membrane	GO:0016020	453	C	other mer	ISS
13	AT1G01020	gene:2200939	AT1G01020.1	located in	mitochondrion	GO:0005739	486	C	mitochon	ISM
14	AT1G01020	locus:2200940	AT1G01020	involved in	sterol metabolic process	GO:0016125	7324	P	other met	IMP
15	AT1G01020	locus:2200940	AT1G01020	has	molecular_function	GO:0003674	3226	F	unknown	ND
16	AT1G01020	locus:2200940	AT1G01020	located in	endoplasmic reticulum	GO:0005783	268	C	ER	IDA

核(nucleus)に存在する遺伝子のみからなるリストを得たいときにもRが利用可能

アノテーションファイルからの情報抽出

入力：アノテーションファイル
（annotation.txt）

	A	B	C	D
1	genename	accession	description	subcellular_location
2	gene1	hoge01	plasma_mem	nuclear
3	gene2	hoge02	hohinu	membrane
4	gene3	hoge03	agribio	endoplasmic
5	gene4	hoge04	genesis	endoplasmic
6	gene5	hoge05	kamo	membrane
7	gene6	hoge06	netteba	humei
8	gene7	hoge07	tebasaki	nuclear
9	gene8	hoge08	biiru	nuclear
10	gene9	hoge09	nihonshu	nuclear
11	gene10	hoge10	agene1	membrane
12	gene11	hoge11	iyaaaa	endoplasmic

入力：リストファイル（genelist1.txt）

	A
1	gene1
2	gene7
3	gene9

出力：hoge1.txt

hoge1.txt

	A	B	C	D
1	gene1	hoge01	plasma_mem	nuclear
2	gene7	hoge07	tebasaki	nuclear
3	gene9	hoge09	nihonshu	nuclear

目的：アノテーションファイル（annotation.txt）中の第1列目に対して、リストファイル（genelist1.txt）中の文字列と一致する行を抜き出して、hoge1.txtというファイル名で出力したい

(Rで)塩基配列解析(主にNGSやRNA-seq解析)

(last modified 2014/02/06, since 2010)

What's new?

- 項目名の整理を行っています。3CやBS-seq周辺についても少し言及してあります。(2014/02/06) NEW
- 一連の解析パイプライン(RNA-seqデータ取得 -> マッピング -> カウントデータやRPKMデータ取得 -> サンプル間クラスタリングや発現変動解析およびM-A plot描画まで)をアップデートしました。項目名の一番下のほうです。(2013/10/19)
- 発現変動解析用Rパッケージ TCC (ver. 1.2.0; Sun et al., BMC Bioinformatics, 2013)がBioconductorよりリリースされました。最新版を利用したい方は、R (ver. 3.0.2)をインストールしたのち、Bioconductor (ver. 2.13)をインストールしてください。(2013/10/17)
- どのブラウザからでもエラーなく見られる(W3C validation)ように((Rで)マイクロアレイデータ解析も含めて)リニューアルしました。(2013/07/30)
- 2013年7月29日まで公開していた以前の「(Rで)塩基配列解析」のウェブページや関連ファイルはRdeennki.zipからダウンロード可能です(110MB程度)。(2013/07/30)

- はじめに(last modified 2014/01/30) NEW
- Rのインストールと起動(last modified 2013/09/27)
- サンプルデータ(last modified 2014/02/06) NEW
- イントロ | 一般 | ランダムに行を抽出(last modified 2013/10/1(
- イントロ | 一般 | 任意の文字列を行の最初に挿入(last modifi
- イントロ | 一般 | 任意のキーワードを含む行を抽出(基礎)(last
- イントロ | 一般 | ランダムな塩基配列を生成(last modified 201
- イントロ | 一般 | 任意の長さの可能な全ての塩基配列を作成(
- イントロ | 一般 | 任意の位置の塩基を置換(last modified 2013
- イントロ | 一般 | 指定した範囲の配列を取得(last modified 20
- イントロ | 一般 | 翻訳配列(translate)を取得(last modified 2013
- イントロ | 一般 | 相補鎖(complement)を取得(last modified 20
- イントロ | 一般 | 逆相補鎖(reverse complement)を取得(last m
- イントロ | 一般 | 逆鎖(reverse)を取得(last modified 2013/06/14
- イントロ | 一般 | 2連続塩基の出現頻度情報を取得(last modi
- イントロ | 一般 | 3連続塩基の出現頻度情報を取得(last modi
- イントロ | 一般 | 任意の長さの連続塩基の出現頻度情報を取
- イントロ | 一般 | Tips | 任意の拡張子でファイルを保存(last m
- イントロ | 一般 | Tips | 拡張子は同じで任意の文字を追加して
- イントロ | 一般 | 配列取得 | ゲノム配列 | 公共DBから(last mo
- イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome(last modi

イントロ | 一般 | 任意のキーワードを含む行を抽出(基礎) NEW

例えばタブ区切りテキストファイルの annotation.txtが手元にあり、この中からgenelist1.txtのようなリストファイル中の文字列を含む行を抽出するやり方を示します。
Linux (UNIX)のgrepコマンドのようなものです。perlのハッシュのようなものです。
「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. 目的のタブ区切りテキストファイル(annotation.txt)中の第1列目をキーとして、リストファイル(genelist1.txt)中のものが含まれる行全体を出力したい場合:

```
in_f1 <- "annotation.txt"                #入力ファイル名を指定してin_f1に格納(アノテーションファイル)
in_f2 <- "genelist1.txt"                 #入力ファイル名を指定してin_f2に格納(リストファイル)
out_f <- "hoge1.txt"                     #出力ファイル名を指定してout_fに格納
param <- 1                               #アノテーションファイル中の検索したい列番号を指定

#入力ファイルの読み込み
data <- read.table(in_f1, header=TRUE, sep="\t", quote="")#in_f1で指定したファイルの読み込み
keywords <- readLines(in_f2)             #in_f2で指定したファイルの読み込み
dim(data)                                #オブジェクトdataの行数と列数を表示

#本番
obj <- is.element(as.character(data[,param]), keywords)#条件を満たすかどうかを判定した結果をobjに格納
out <- data[obj,]                        #objがTRUEとなる行のみ抽出した結果をoutに格納
dim(out)                                 #オブジェクトoutの行数と列数を表示

#ファイルに保存
write.table(out, out_f, sep="\t", append=F, quote=F, row.names=F)#outの中身をout_fで指定したファイル
```

1. 目的のタブ区切りテキストファイル(annotation.txt)中の第1列目をキーとして、リストファイル(genelist1.txt)中のものが含まれる行全体を出力したい場合:

```
in_f1 <- "annotation.txt"                #入力ファイル名を指定してin_f1に格納(アノテーションファイル)
in_f2 <- "genelist1.txt"                 #入力ファイル名を指定してin_f2に格納(リストファイル)
out_f <- "hoge1.txt"                     #出力ファイル名を指定してout_fに格納
param <- 1                               #アノテーションファイル中の検索したい列番号を指定

#入力ファイルの読み込み
data <- read.table(in_f1, header=TRUE, sep="\t", quote="")#in_f1で指定したファイルの読み込み
keywords <- readLines(in_f2)             #in_f2で指定したファイルの読み込み
dim(data)                                #オブジェクトdataの行数と列数を表示

#本番
obj <- is.element(as.character(data[,param]), keywords)#条件を満たすかどうかを判定した結果をobjに格納
out <- data[obj,]                        #objがTRUEとなる行のみ抽出した結果をoutに格納
dim(out)                                 #オブジェクトoutの行数と列数を表示

#ファイルに保存
write.table(out, out_f, sep="\t", append=F, quote=F, row.names=F)#outの中身をout_fで指定したファイル名で保存
```

入力1: annotation.txt

	A	B	C	D
1	genename	accession	description	subcellular_location
2	gene1	hoge01	plasma_mem	nuclear
3	gene2	hoge02	hohinu	membrane
4	gene3	hoge03	agribio	endoplasmic
5	gene4	hoge04	genesis	endoplasmic
6	gene5	hoge05	kamo	membrane
7	gene6	hoge06	netteba	humei
8	gene7	hoge07	tebasaki	nuclear
9	gene8	hoge08	biiru	nuclear
10	gene9	hoge09	nihonshu	nuclear
11	gene10	hoge10	agene1	membrane
12	gene11	hoge11	iyaaaa	endoplasmic

入力2: genelist1.txt

	A
1	gene1
2	gene7
3	gene9

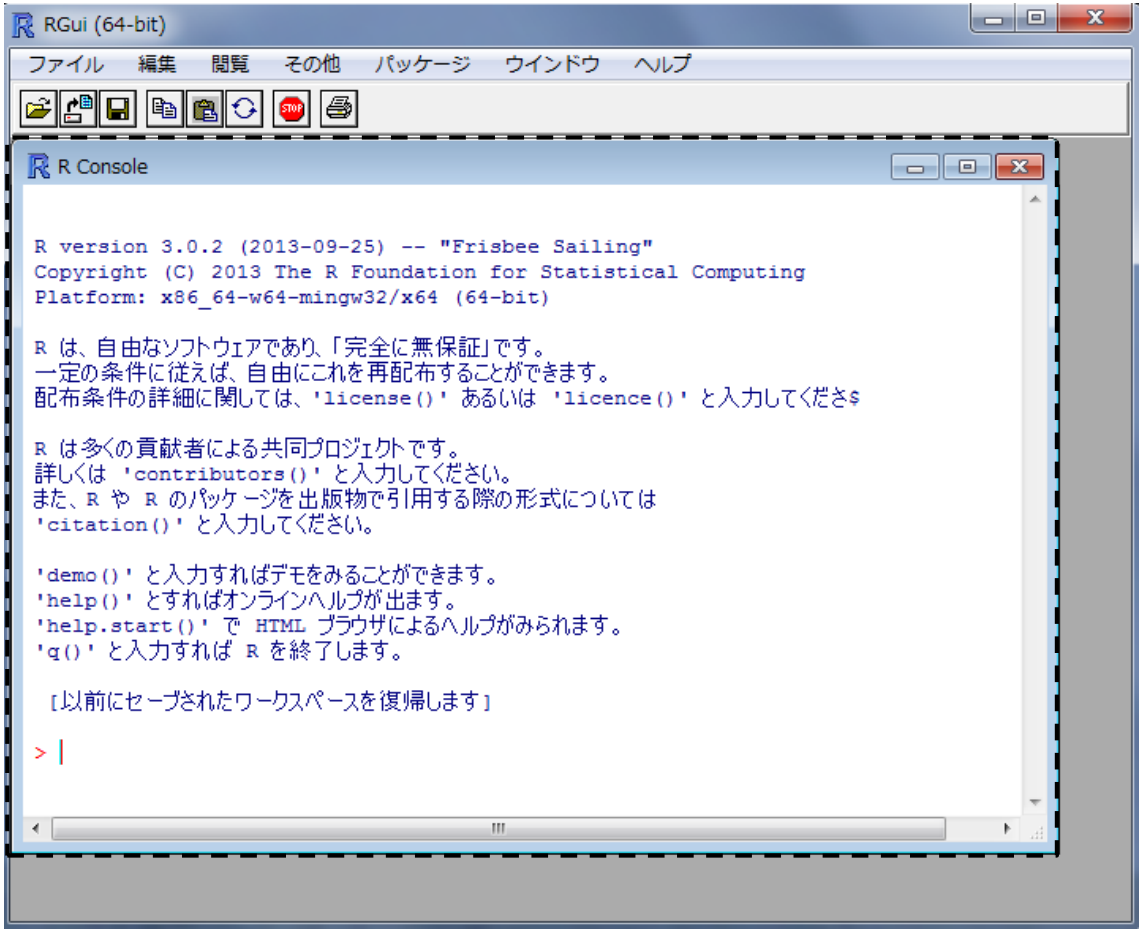
出力: hoge1.txt

hoge1.txt

	A	B	C	D
1	gene1	hoge01	plasma_mem	nuclear
2	gene7	hoge07	tebasaki	nuclear
3	gene9	hoge09	nihonshu	nuclear

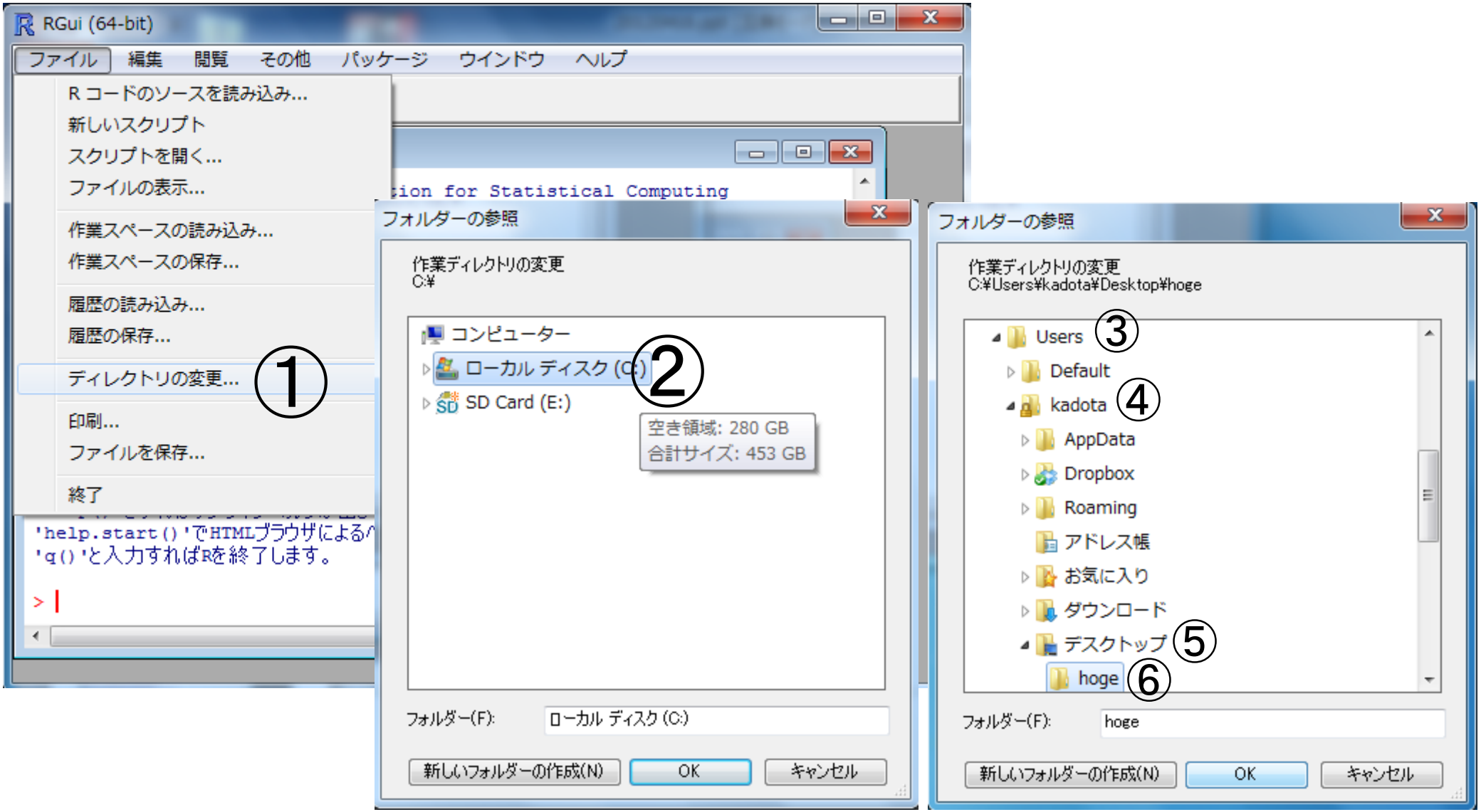
デスクトップ上にhogeという名前のフォルダがあり、フォルダ中にannotation.txtとgenelist1.txtが存在するという前提です。メモ帳で開くと改行コードが崩れている場合は、ワードパッドなどで開くとよい

Rの起動

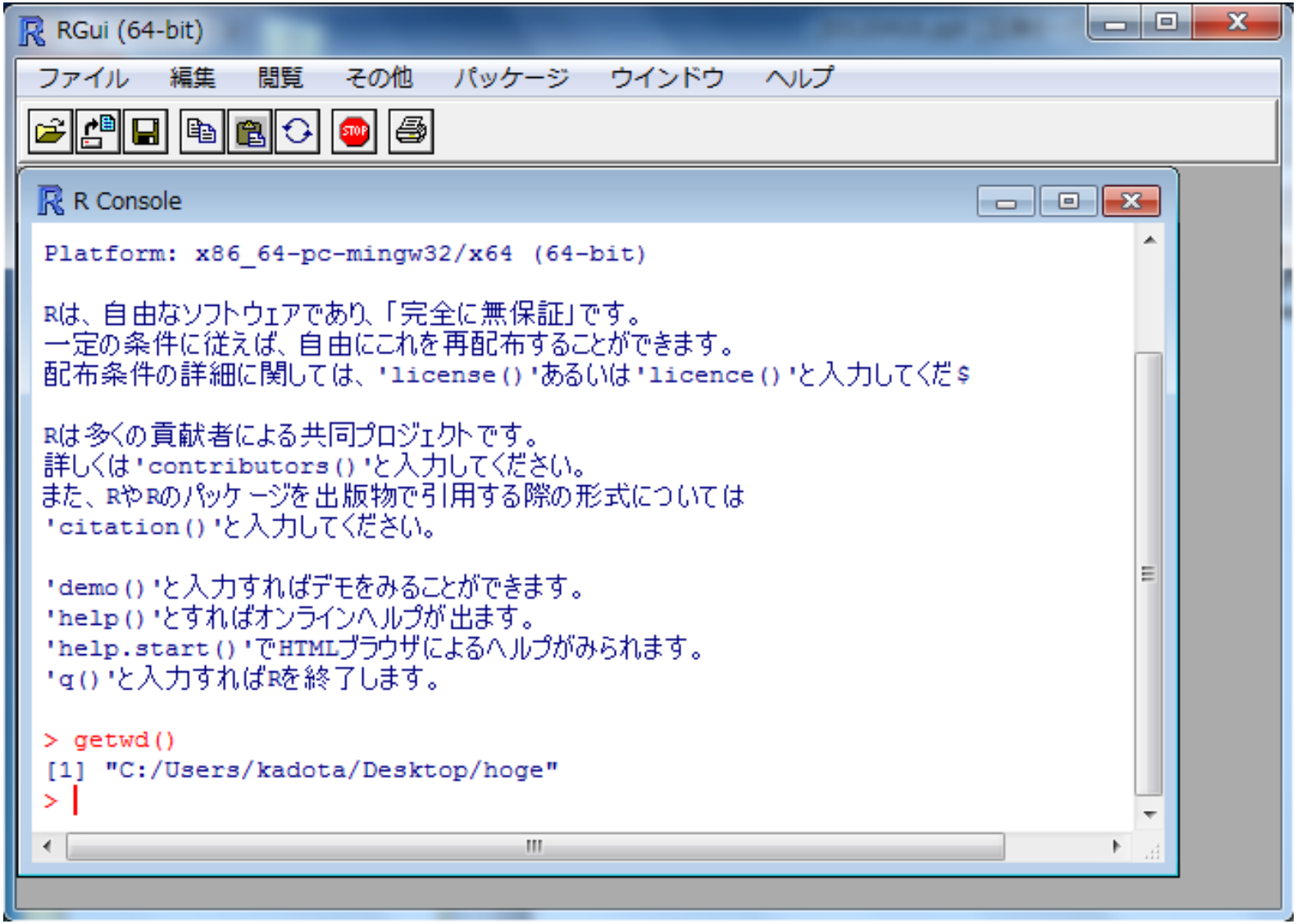


デスクトップにあるhogeフォルダ中のファイルを解析

作業ディレクトリの変更



getwd() と打ち込んで確認



基本はコピペ

2013年7月以降のリニューアルで、コードのコピーがやりずらくなっています。**CTRLとALTキー**を押しながらコードの枠内で**左クリック**すると、全選択できます。

イントロ | 一般 | 任意のキーワードを含む行を抽出(基礎) NEW

例えばタブ区切りテキストファイルが手元にあり、この中からリストファイル中の文字列を含む行を抽出するやり
(UNIX)のgrepコマンドのようなものであり、perlのハッシュのようなものです。
「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. 目的のタブ区切りテキストファイル(annotation.txt)中の行全体を出力したい場合:

```
in_f1 <- "annotation.txt"
in_f2 <- "genelist1.txt"
out_f <- "hoge1.txt"
param <- 1

#入力ファイルの読み込み
data <- read.table(in f1, he
keywords <- readLines(in f2)
dim(data)

#本番
obj <- is.element(as.charact
out <- data[obj,]
dim(out)

#ファイルに保存
write.table(out, out_f, sep=
```

切り取り(T)
コピー(C)
貼り付け
すべて選択(A)
印刷(I)...
印刷プレビュー
Bing でマップ
Bing で翻訳
Google で検索
電子メール (W
すべてのアクセ
Send to OneN

RGui (64-bit)

ファイル 編集 閲覧 その他 パッケージ ウインドウ ヘルプ

R Console

```
Platform: x86_64-pc-mingw32/x64 (64-bit)

Rは、自由なソフトウェアであり、「完全に無保証」です。
一定の条件に従えば、自由にこれを再配布することができます。
配布条件の詳細に関しては、'license()'あるいは'licence()'と入力してくだ

Rは多くの貢献者による共同プロジェクトです。
詳しくは'contributors()'と入力してくだ
また、RやRのパッケージを出版物で引用する
'citation()'と入力してください。

'demo()'と入力すればデモをみることができ
'help()'とすればオンラインヘルプが出ます。
'help.start()'でHTMLブラウザによるヘル
'q()'と入力すればRを終了します。

> getwd()
[1] "C:/Users/kadota/Desktop/hoge
>
```

コピー Ctrl+C
ペースト Ctrl+V
コマンドのみペースト
コピー&ペースト Ctrl+X
ウインドウの消去 Ctrl+L
全て選択
バッファに出力 Ctrl+W

①一連のコマンド群を**コピー**して
②R Console画面上で**ペースト**

実行結果

RGui (64-bit)

ファイル　編集　閲覧　その他　パッケージ　ウインドウ　ヘルプ

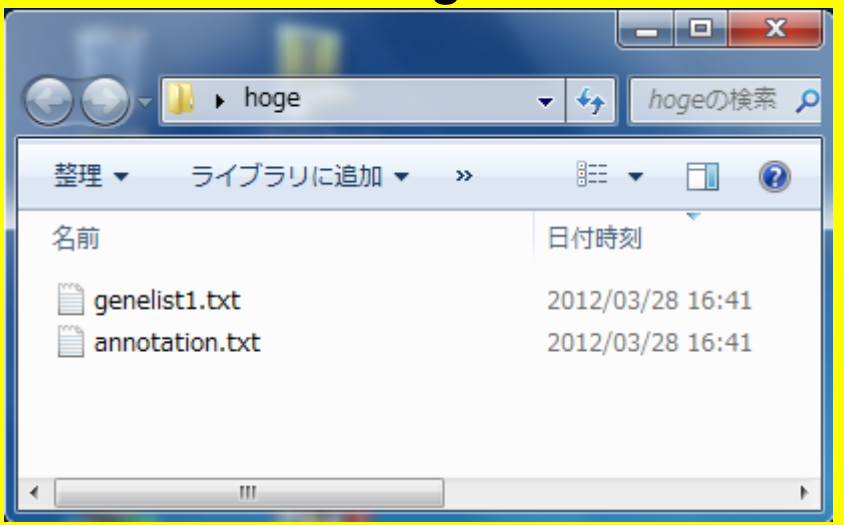
R Console

```
> in_f2 <- "genelist1.txt"                                                #
> out_f <- "hoge1.txt"                                                    #
> param <- 1                                                              #
>
> #ファイルの読み込み
> data <- read.table(in_f1, header=TRUE, sep="\t", quote="")              #
> keywords <- readLines(in_f2)                                            #
> dim(data)                                                               #
[1] 11  4
>
> #本番
> obj <- is.element(as.character(data[,param]), keywords)                 #
> out <- data[obj,]                                                       #
> dim(out)                                                                #
[1] 3 4
> write.table(out, out_f, sep="\t", append=F, quote=F, row.names=F)#
>
>
```

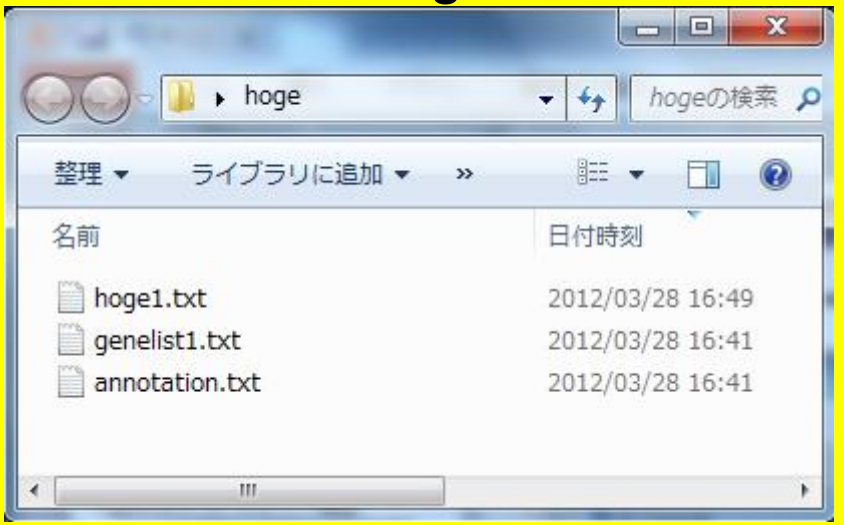
hoge1.txt

	A	B	C	D
1	genename	accession	description	subcellular_location
2	gene1	hoge01	plasma_mem	nuclear
3	gene7	hoge07	tebasaki	nuclear
4	gene9	hoge09	nihonshu	nuclear

実行前のhogeフォルダ



実行後のhogeフォルダ



(Rで)塩基配列解析(主にNGSやRNA-seq解析)

(last modified 2014/02/06, since 2010)

What's new?

- 項目名の整理を行っています。3CやBS-seq周辺についても少し言及してあります。(2014/02/06) NEW
- 一連の解析パイプライン(RNA-seqデータ取得 -> マッピング -> カウントデータやRPKMデータ取得 -> サンプル間クラスタリングや発現変動解析およびM-A plot描画まで)をアップデートしました。項目名の一番下のほうです。(2013/10/19)
- 発現変動解析用Rパッケージ TCC (ver. 1.2.0; Sun et al., BMC Bioinformatics, 2013)がBioconductorよりリリースされました。最新版を利用したい方は、R (ver. 3.0.2)をインストールしたのち、Bioconductor (ver. 2.13)をインストールしてください。(2013/10/17)
- どのブラウザからでもエラーなく見られる(W3C validation)ように((Rで)マイクロアレイデータ解析も含めて)リニューアルしました。(2013/07/30)
- 2013年7月29日まで公開していた以前の「(Rで)塩基配列解析」のウェブページや関連ファイルは Rdeennki.zip からダウンロード可能です(110MB程度)。(2013/07/30)

- はじめに (modified 2014/01/30) NEW
- Rのインストールと起動 (last modified 2013/09/27)
- サンプルデータ (last modified 2014/02/06) NEW
- イントロ | 一般 | ランダムに行を抽出 (last modified 2013/10/10)
- イントロ | 一般 | 任意の文字列を行の最初に挿入 (last modified 2013/10/10)
- イントロ | 一般 | 任意のキーワードを含む行を抽出(基礎) (last modified 2014/02/06) NEW
- イントロ | 一般 | ランダムな塩基配列を生成 (last modified 2013/09/29)
- イントロ | 一般 | 任意の長さの 可能な全ての 塩基配列を作成 (last modified 2013/06/14)
- イントロ | 一般 | 任意の位置の塩基を置換 (last modified 2013/09/12)
- イントロ | 一般 | 指定した範囲の配列を取得 (last modified 2013/09/29)
- イントロ | 一般 | 翻訳配列(translate)を取得 (last modified 2013/06/14)
- イントロ | 一般 | 相補鎖(complement)を取得 (last modified 2013/06/14)
- イントロ | 一般 | 逆相補鎖(reverse complement)を取得 (last modified 2013/06/14)
- イントロ | 一般 | 逆鎖(reverse)を取得 (last modified 2013/06/14)
- イントロ | 一般 | 2連続塩基の 出現頻度情報を取得 (last modified 2014/02/05) NEW
- イントロ | 一般 | 3連続塩基の 出現頻度情報を取得 (last modified 2013/06/14)
- イントロ | 一般 | 任意の長さの 連続塩基の 出現頻度情報を取得 (last modified
- イントロ | 一般 | Tips | 任意の 拡張子でファイルを保存 (last modified 2013/09/2
- イントロ | 一般 | Tips | 拡張子は同じで任意の文字を追加して保存 (last modifie
- イントロ | 一般 | 配列取得 | ゲノム配列 | 公共DBから (last modified 2013/08/15
- イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome (last modified 2012/02/05)

このページ内で用いる色についての説明:

コメント

特にやらなくてもいいコマンド

プログラム実行時に目的に応じて変更すべき箇所

色についての説明

このページ内で用いる色についての説明:
コメント
特にやらなくてもいいコマンド
プログラム実行時に目的に応じて変更すべき箇所

```
in_f1 <- "annotation.txt"                #入力ファイル名を指定してin_f1に格納(アノテーションファイル)
in_f2 <- "genelist1.txt"                 #入力ファイル名を指定してin_f2に格納(リストファイル)
out_f <- "hoge1.txt"                     #出力ファイル名を指定してout_fに格納
param <- 1                               #アノテーションファイル中の検索したい列番号を指定

#入力ファイルの読み込み
data <- read.table(in_f1, header=TRUE, sep="\t", quote="")#in_f1で指定したファイルの読み込み
keywords <- readLines(in_f2)             #in_f2で指定したファイルの読み込み
dim(data)                                #オブジェクトdataの行数と列数を表示

#本番
obj <- is.element(as.character(data[,param]), keywords)#条件を満たすかどうかを判定した結果をobjに格納
out <- data[obj,]                        #objがTRUEとなる行のみ抽出した結果をoutに格納
dim(out)                                 #オブジェクトoutの行数と列数を表示

#ファイルに保存
write.table(out, out_f, sep="\t", append=F, quote=F, row.names=F)#outの中身をout_fで指定したファイル名で保存
```

上記は1列目でキーワード検索する場合

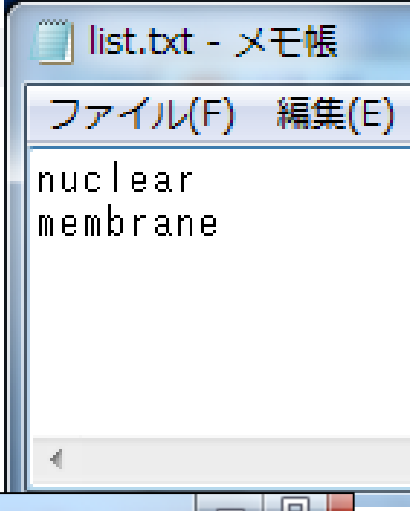
	A	B	C	D
1	genename	accession	description	subcellular_location
2	gene1	hoge01	plasma_mem	nuclear
3	gene2	hoge02	hohinu	membrane
4	gene3	hoge03	agribio	endoplasmic
5	gene4	hoge04	genesis	endoplasmic
6	gene5	hoge05	kamo	membrane
7	gene6	hoge06	netteba	humei
8	gene7	hoge07	tebasaki	nuclear
9	gene8	hoge08	biiru	nuclear
10	gene9	hoge09	nihonshu	nuclear
11	gene10	hoge10	agene1	membrane
12	gene11	hoge11	iyaaaa	endoplasmic

4列目でキーワード検索したいときは?

	A	B	C	D
1	genename	accession	description	subcellular_location
2	gene1	hoge01	plasma_mem	nuclear
3	gene2	hoge02	hohinu	membrane
4	gene3	hoge03	agribio	endoplasmic
5	gene4	hoge04	genesis	endoplasmic
6	gene5	hoge05	kamo	membrane
7	gene6	hoge06	netteba	humei
8	gene7	hoge07	tebasaki	nuclear
9	gene8	hoge08	biiru	nuclear
10	gene9	hoge09	nihonshu	nuclear
11	gene10	hoge10	agene1	membrane
12	gene11	hoge11	iyaaaa	endoplasmic

解答例

1. 目的のキーワードリストを含むファイルを作成し（例：list.txt）
2. 該当箇所を変更し、R Console画面上でコピペ



run1.txt - メモ帳

ファイル(F) 編集(E) 書式(O) 表

```
in_f1 <- "annotation.txt"
in_f2 <- "genelist1.txt"
out_f <- "hoge1.txt"
param <- 1

#ファイルの読み込み
data <- read.table(in_f1,
keywords <- readLines(in_f
dim(data)

#本番
obj <- is.element(as.chara
out <- data[obj,]
dim(out)
write.table(out, out_f, se
```

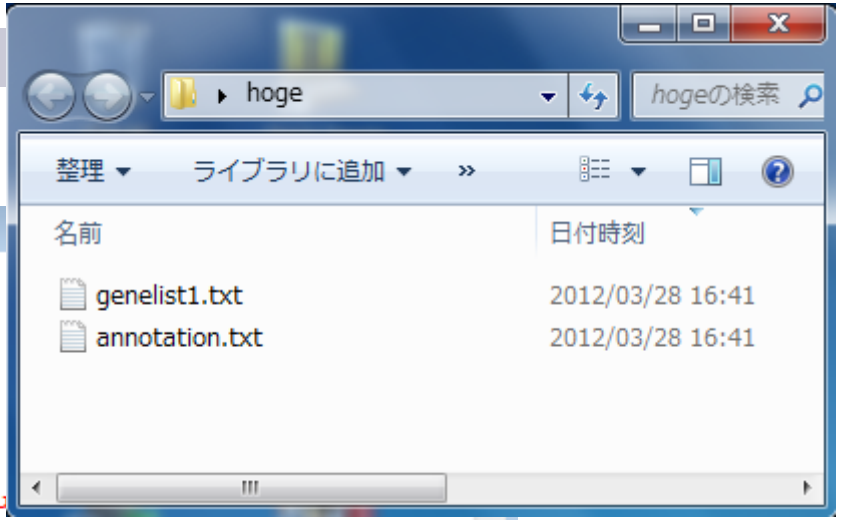
run1.txt - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
in_f1 <- "annotation.txt"
in_f2 <- "list.txt"
out_f <- "hoge1.txt"
param <- 4

#ファイルの読み込み
data <- read.table(in_f1, header=TRUE, sep="¥t", quote="")
keywords <- readLines(in_f2)
dim(data)

#本番
obj <- is.element(as.character(data[,param]), keywords)
out <- data[obj,]
dim(out)
write.table(out, out_f, sep="¥t", append=F, quote=F, row.names=
```

一連の作業手順を記述した**スクリプト**を1つのファイルとして保存することをお勧め。list.txtファイル作成時に、**membrane**と打った後に改行を入れた場合と入れない場合の挙動の違いも見ておくとよい

ありがちなミス1

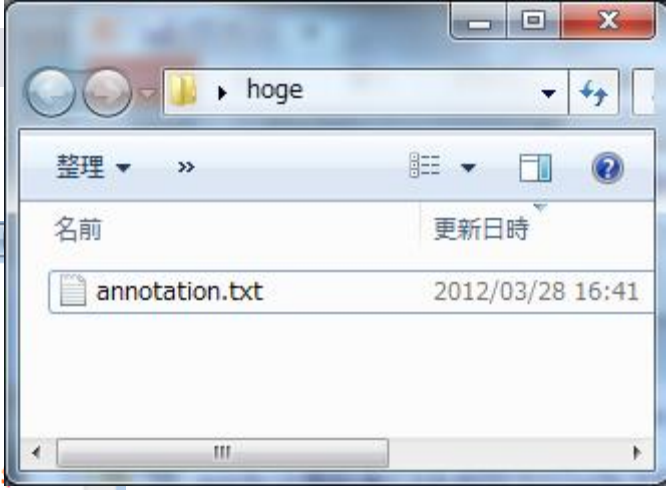


R Console

```
> in_f1 <- "annotation.txt"
> in_f2 <- "genelist1.txt"
> out_f <- "hoge1.txt"
> param <- 1
>
> #ファイルの読み込み
> data <- read.table(in_f1, header=TRUE, sep="\t", qu
 以下にエラー file(file, "rt") :  コネクションを開くことができません
 追加情報:    警告メッセージ:
In file(file, "rt") :
   ファイル 'annotation.txt' を開くことができません: No such file or director$
> keywords <- readLines(in_f2)                                    #入力ファ$
 以下にエラー file(con, "r") :  コネクションを開くことができません
 追加情報:    警告メッセージ:
In file(con, "r") :
   ファイル 'genelist1.txt' を開くことができません: No such file or directory
> dim(data)                                                       #オブジェ$
NULL
>
> #本番
> obj <- is.element(as.character(data[,param]), keywords)         #in_f1で読$
 以下にエラー data[, param] :
   'closure' 型のオブジェクトは部分代入可能ではありません
> out <- data[obj,]                                               #行列data$
 エラー:  オブジェクト 'obj' がありません
> dim(out)                                                        #オブジェ$
 エラー:  オブジェクト 'out' がありません
> write.table(out, out_f, sep="\t", append=F, quote=F, row.names=F)#outの中身$
 以下にエラー is.data.frame(x) :  オブジェクト 'out' がありません
>
> getwd()
[1] "C:/Users/kadota/Documents"
```

作業ディレクトリの変更を忘れている…

ありがちなミス2

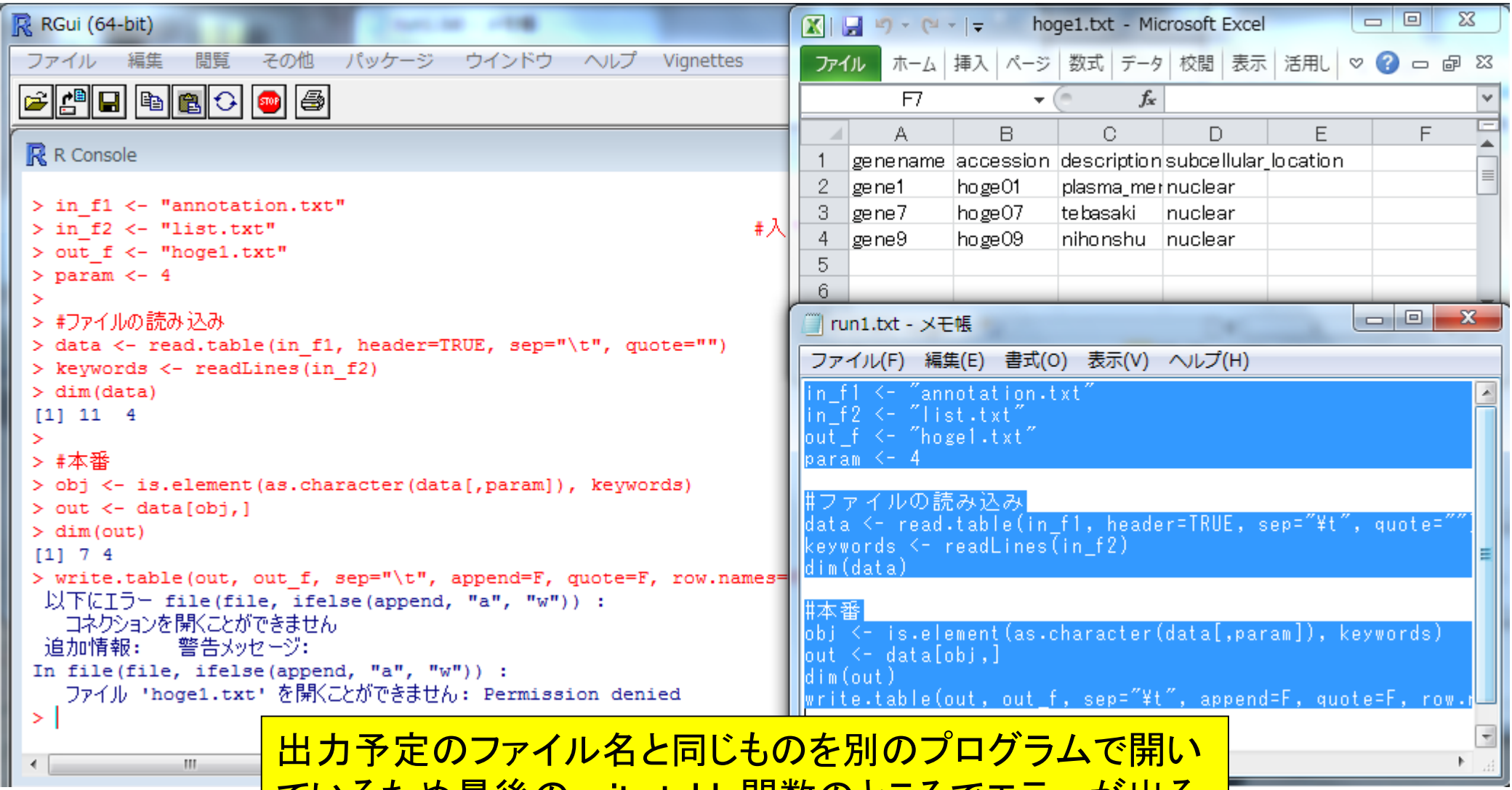


R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> in_f1 <- "annotation.txt"                                    #入力ファイル$
> in_f2 <- "genelist1.txt"                                     #入力ファイル$
> out_f <- "hoge1.txt"                                         #出力ファイル$
> param <- 1                                                   #in_f1で読み
>
> #ファイルの読み込み
> data <- read.table(in_f1, header=TRUE, sep="\t", quote="")   #入力ファイル$
> keywords <- readLines(in_f2)                                 #入力ファイル$
 以下にエラー file(con, "r") :  コネクションを開くことができません
 追加情報:    警告メッセージ:
In file(con, "r") :
   ファイル 'genelist1.txt' を開くことができません: No such file or directory
> dim(data)                                                    #オブジェクト$
[1] 11  4
>
> #本番
> obj <- is.element(as.character(data[,param]), keywords)      #in_f1で読み$
 以下にエラー match(el, set, 0L) :  オブジェクト 'keywords' がありません
> out <- data[obj,]                                            #行列dataから$
 以下にエラー `[.data.frame`(data, obj, ) :  オブジェクト 'obj' がありません
> dim(out)                                                     #オブジェクト$
 エラー:  オブジェクト 'out' がありません
> write.table(out, out_f, sep="\t", append=F, quote=F, row.names=F)#outの中身をo$
 以下にエラー is.data.frame(x) :  オブジェクト 'out' がありません
>
```

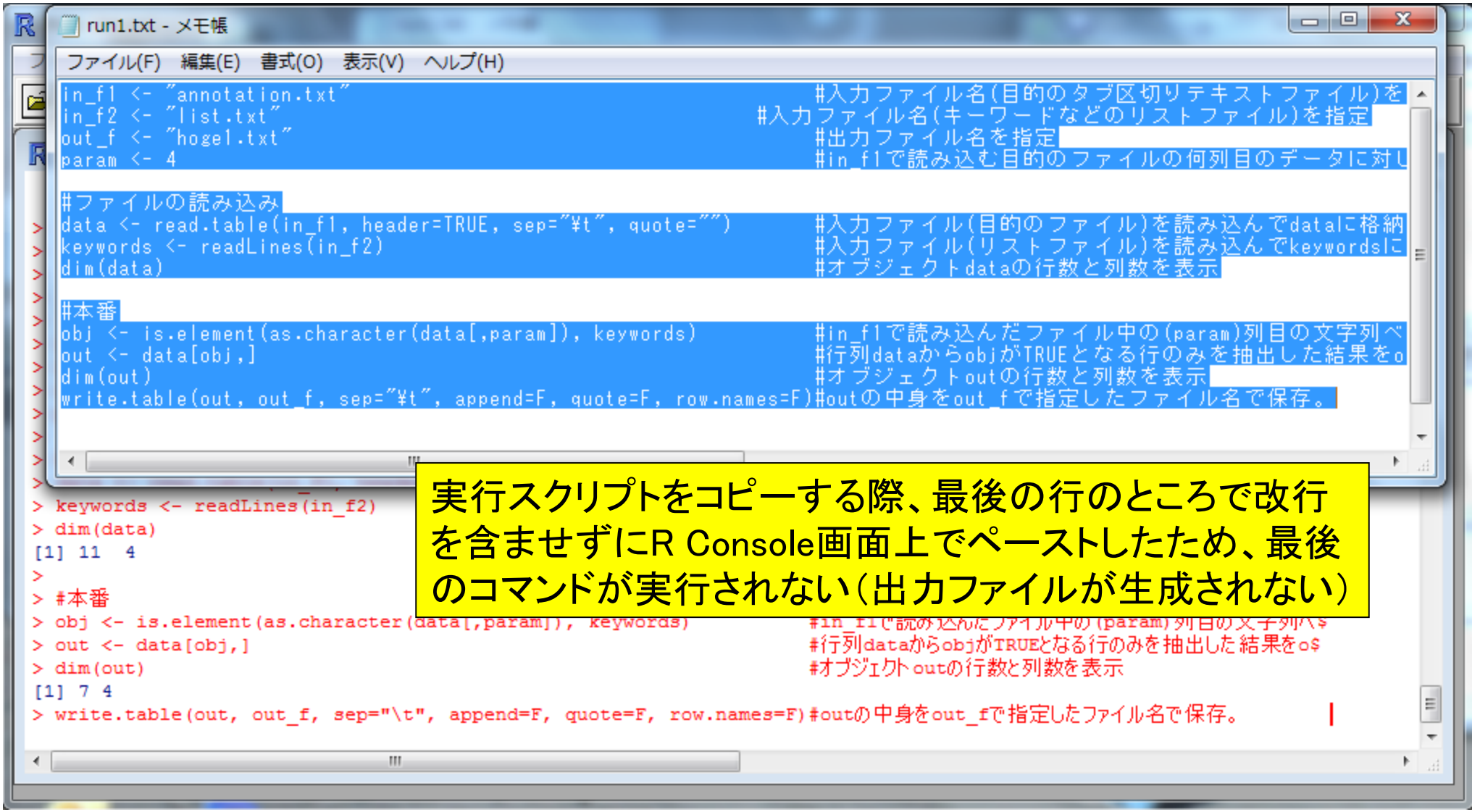
必要な入力ファイルが作業ディレクトリ中に存在しない…

ありがちなミス3



出力予定のファイル名と同じものを別のプログラムで開いているため最後のwrite.table関数のところでエラーが出る

ありがちなミス4



実行スクリプトをコピーする際、最後の行のところで改行を含ませずにR Console画面上でペーストしたため、最後のコマンドが実行されない（出力ファイルが生成されない）

読み込み

annotation.txt 参考

	A	B	C	D
1	genename	accession	description	subcellular_location
2	gene1	hoge01	plasma_mem	nuclear
3	gene2	hoge02	hohinu	membrane
4	gene3	hoge03	agribio	endoplasmic
5	gene4	hoge04	genesis	endoplasmic
6	gene5	hoge05	kamo	membrane
7	gene6	hoge06	netteba	humei
8	gene7	hoge07	tebasaki	nuclear
9	gene8	hoge08	biiru	nuclear
10	gene9	hoge09	nihonshu	nuclear
11	gene10	hoge10	agene1	membrane
12	gene11	hoge11	iyaaaa	endoplasmic

```
------    ここから    ------
in_f1 <- "annotation.txt"
in_f2 <- "genelist1.txt"
out_f <- "hoge1.txt"
param <- 1

#ファイルの読み込み
data <- read.table(in_f1, header=TRUE, sep="¥t", quote="")
keywords <- readLines(in_f2)
dim(data)

#本番
obj <- is.element(as.character
out <- data[obj,]
dim(out)
write.table(out, out_f, sep="¥t", append=F, quote=F, row.names=F)

------    ここまで    ------
```

①in_f1で指定したファイルを読み込め
②読み込むファイルの最初の行はヘッダー部分です
③ファイルの区切り文字は**タブ**です

行列data

annotation.txt 参考

	A	B	C	D
1	genename	accession	description	subcellular_location
2	gene1	hoge01	plasma_mem	nuclear
3	gene2	hoge02	hohinu	membrane
4	gene3	hoge03	agribio	endoplasmic
5	gene4	hoge04	genesis	endoplasmic
6	gene5	hoge05	kamo	membrane
7	gene6	hoge06	netteba	humei
8	gene7	hoge07	tebasaki	nuclear
9	gene8	hoge08	biiru	nuclear
10	gene9	hoge09	nihonshu	nuclear
11	gene10	hoge10	agene1	membrane
12	gene11	hoge11	iyaaaa	endoplasmic

RGui (64-bit)

ファイル 編集 閲覧 その他 パッケージ ウインドウ ヘルプ

R Console

```
>
> #ファイルの読み込み
> data <- read.table(in_f1, header=TRUE, sep="\t", qu
>
> data
   genename accession description subcellular_location
1     gene1     hoge01  plasma_mem              nuclear
2     gene2     hoge02      hohinu             membrane
3     gene3     hoge03     agribio           endoplasmic
4     gene4     hoge04     genesis           endoplasmic
5     gene5     hoge05        kamo             membrane
6     gene6     hoge06     netteba                humei
7     gene7     hoge07    tebasaki              nuclear
8     gene8     hoge08       biiru              nuclear
9     gene9     hoge09    nihonshu              nuclear
10   gene10     hoge10      agene1             membrane
11   gene11     hoge11      iyaaaa           endoplasmic
>
```

入力ファイルの中身を正しく
読み込めていることがわかる

```
------    ここから    ------
in_f1 <- "annotation.txt"
in_f2 <- "genelist1.txt"
out_f <- "hoge1.txt"
param <- 1

#ファイルの読み込み
data <- read.table(i
keywords <- readLine
dim(data)

#本番
obj <- is.element(as
out <- data[obj,]
dim(out)
write.table(out, out

------    ここまで
```

annotation.txt　参考

	A	B	C	D
1	genename	accession	description	subcellular_location
2	gene1	hoge01	plasma_mem	nuclear
3	gene2	hoge02	hohinu	membrane
4	gene3	hoge03	agribio	endoplasmic
5	gene4	hoge04	genesis	endoplasmic
6	gene5	hoge05	kamo	membrane
7	gene6	hoge06	netteba	humei
8	gene7	hoge07	tebasaki	nuclear
9	gene8	hoge08	biiru	nuclear
10	gene9	hoge09	nihonshu	nuclear
11	gene10	hoge10	agene1	membrane
12	gene11	hoge11	iyaaaa	endoplasmic

RGui (64-bit)

ファイル　編集　閲覧

R Console

```
> data
   genename accessio
1     gene1    hoge01  plasma_mem          nuclear
2     gene2    hoge02      hohinu         membrane
3     gene3    hoge03     agribio       endoplasmic
4     gene4    hoge04
5     gene5    hoge05
6     gene6    hoge06
7     gene7    hoge07
8     gene8    hoge08
9     gene9    hoge09    nihonshu          nuclear
10   gene10    hoge10      agene1         membrane
11   gene11    hoge11      iyaaaa       endoplasmic
> dim(data)
[1] 11  4
>
```

オブジェクトdataの行数と列数は11と4。
webpage中の表記が灰色なのは、特にやらなくてもいいコマンドだから。

行列の要素へのアクセス

RGui (64-bit)

ファイル　編集　閲覧　その他　パッケージ　ウインドウ　ヘルプ

R Console

```
10    gene10     hoge10       agene1              memb
11    gene11     hoge11       iyaaaa           endopla
> dim(data)
[1] 11  4
> data[6,4]
[1] humei
Levels: endoplasmic humei membrane nuclear
> data[2,]
  genename accession description subcellular_location
2    gene2    hoge02      hohinu              membrane
> data[,2]
 [1] hoge01 hoge02 hoge03 hoge04 hoge05 hoge06 hoge07
 [8] hoge08 hoge09 hoge10 hoge11
11 Levels: hoge01 hoge02 hoge03 hoge04 hoge05 ... hoge11
> data[,param]
 [1] gene1  gene2  gene3  gene4  gene5  gene6  gene7
 [8] gene8  gene9  gene10 gene11
11 Levels: gene1 gene10 gene11 gene2 gene3 ... gene9
>
```

data[行，列]

annotation.txt

	A	B	C	D
1	genename	accession	description	subcellular_location
2	gene1	hoge01	plasma_mem	nuclear
3	gene2	hoge02	hohinu	membrane
4	gene3	hoge03	agribio	endoplasmic
5	gene4	hoge04	genesis	endoplasmic
6	gene5	hoge05	kamo	membrane
7	gene6	hoge06	netteba	humei
8	gene7	hoge07	tebasaki	nuclear
9	gene8	hoge08	biiru	nuclear
10	gene9	hoge09	nihonshu	nuclear
11	gene10	hoge10	agene1	membrane
12	gene11	hoge11	iyaaaa	endoplasmic

```
------    ここから    ---
in_f1 <- "annotation.txt"
in_f2 <- "genelist1.txt"
out_f <- "hoge1.txt"
param <- 1
```

paramには1という数値が代入されていたから

やりたかったことをおさらい

参考
genelist1.txt

	A
1	gene1
2	gene7
3	gene9

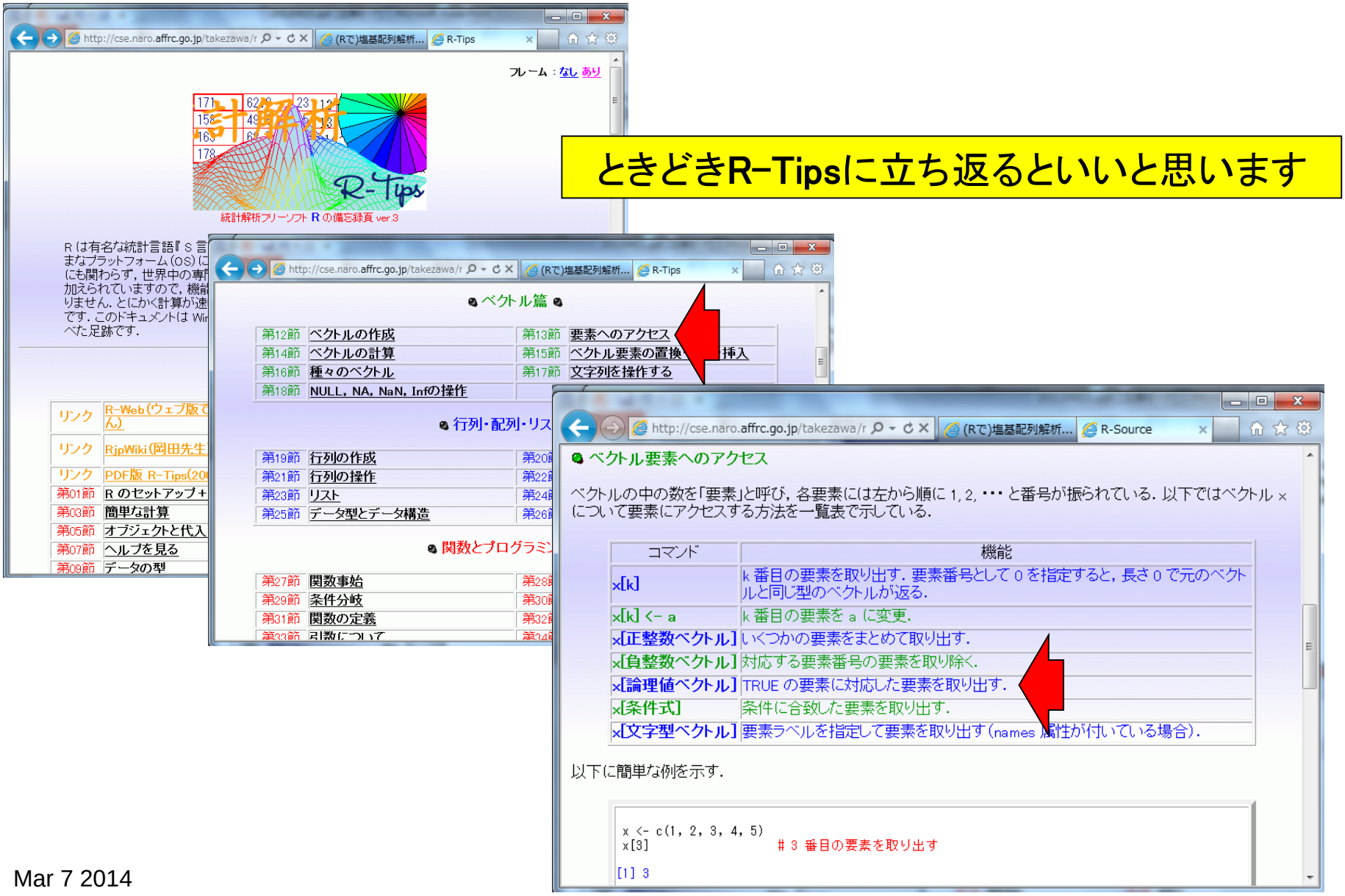
```
------    ここから    ------
in_f1 <- "annotation.txt"
in_f2 <- "genelist1
out_f <- "hoge1.txt
param <- 1

#ファイルの読み込み
data <- read.table(
keywords <- readLin
dim(data)

#本番
obj <- is.element(a
out <- data[obj,]
dim(out)
write.table(out, ou

------    ここまで
```

R Console

```
> obj
 [1]  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE
> data
   genename accession description subcellular_location
1     gene1    hoge01  plasma_mem               nuclear
2     gene2    hoge02      hohinu              membrane
3     gene3    hoge03     agribio           endoplasmic
4     gene4    hoge04     genesis           endoplasmic
5     gene5    hoge05        kamo              membrane
6     gene6    hoge06     netteba                 humei
7     gene7    hoge07    tebasaki               nuclear
8     gene8    hoge08      hijiru               nuclear
9     gene9    hoge09    nihonshu               nuclear
10   gene10    hoge10
11   gene11    hoge11
> obj
 [1]  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE
> data[obj,]
  genename accession description subcellular_location
1    gene1    hoge01  plasma_mem               nuclear
7    gene7    hoge07    tebasaki               nuclear
9    gene9    hoge09    nihonshu               nuclear
>
```

論理値ベクトルobjを用いてTRUEの要素に対応する行を抽出している

R-Tipsでお勉強



ときどきR-Tipsに立ち返るといいと思います

論理値ベクトルを理解

genelist1.txt

	A
1	gene1
2	gene7
3	gene9

```
------    ここから    ------
in_f1 <- "annotation.txt"
in_f2 <- "genelist1.txt"
out_f <- "hoge1.txt"
param <- 1

#ファイルの読み込み
data <- read.table(in_f1, hea
keywords <- readLines(in_f2)
dim(data)

#本番
obj <- is.element(as.character(data[,param]), keywords)
out <- data[obj,]
dim(out)
write.table(out, out_f, sep="¥t", append=F, quote=F, row.names=F)

------    ここまで    ------
```

R Console

```
>
> keywords                          #keywordsの中身を表示
[1] "gene1" "gene7" "gene9"
> length(keywords)                  #keywordsベクトルの要素数を表示
[1] 3
> keywords[3]                       #3番目の要素を表示
[1] "gene9"
> keywords[4]                       #4番目の要素は...ない
[1] NA
> keywords == "gene7"               #"gene7"という文字の位置情報を表示
[1] FALSE  TRUE FALSE
> obj <- keywords == "gene7"        #上記結果をobjに格納
> keywords[obj]                     #objがTRUEとなる要素のみを表示
[1] "gene7"
>
```

論理値ベクトルを理解

genelist1.txt

	A
1	gene1
2	gene7
3	gene9

R Console

```
> hoge <- c("gene7", "gene9")#二つの要素からなるベクトルhogeを作成
> hoge                       #hogeの中身を表示させてるだけ
[1] "gene7" "gene9"
> keywords == hoge           #FALSE TRUE TRUEになることを期待したが...
[1] FALSE FALSE FALSE
 警告メッセージ:
In keywords == hoge :
   長いオブジェクトの長さが短いオブジェクトの長さの倍数になっていません
> is.element(keywords, hoge) #keywords中の各要素は集合hogeに含まれるか否か
[1] FALSE  TRUE  TRUE
>
```

疑問に思ったら、自分の理解できるところから試す

```
#本番
obj <- is.element(as.character(data[,param]), keywords)
out <- data[obj,]
dim(out)
write.table(out, out_f, sep="¥t", append=F, quote=F, row.names=F)

------    ここまで    ------
```

Contents（Rで…）

- ゲノム解析
  - アノテーションファイルを読み込んで目的のキーワードを含む行のみ抽出
  - **multi-FASTAファイルを自在に解析**
    - **配列長分布、GC含量、フィルタリング、部分配列の切り出しなど**
    - 連続塩基の出現頻度（CpG）解析、ゲノム配列取得など
- トランスクリプトーム解析
  - 研究目的別留意点：サンプル内とサンプル間の違い
  - マッピング → カウント情報取得
  - データを眺める：クラスタリングやM-A plot
  - 理想的な実験デザイン
  - なぜx倍発現変動という議論がだめなんですか？
  - モデルとか分布って、自分の解析結果にどういう影響を与えているの？
  - 多重比較問題：FDRって何？

multi-FASTAファイルからの各種情報抽出

(Rで)塩基配列解析(主にNGSやRNA-seq解析)

(last modified 2014/02/06, since 2010)

- イントロ | 一般 | 相補鎖(complement)を取得(last modified 2013/06/14)
- イントロ | 一般 | 逆相補鎖(reverse complement)を取得(last modified 2013/06/14)
- イントロ | 一般 | 逆鎖(reverse)を取得(last modified 2013/06/14)
- イントロ | 一般 | 2連続塩基の 出現頻度情報を取得(last modified 2014/02/05) NEW
- イントロ | 一般 | 3連続塩基の 出現頻度情報を取得(last modified 2013/06/14)
- イントロ | 一般 | 任意の長さの 連続塩基の出現頻度情報を取得(last modified 2013/06/14)
- イントロ | 一般 | Tips | 任意の拡張子でファイルを保存(last modified 2013/09/26)
- イントロ | 一般 | Tips | 拡張子は同じで任意の文字を追加して保存(last modified 2013/09/26)
- イントロ | 一般 | 配列取得 | ゲノム配列 | 公共DBから(last modified 2013/08/15)
- イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome(last modified 2012/02/05)
- イントロ | 一般 | 配列取得 | プロモーター配列 | BSgenome(last modified 2013/10/10)
- イントロ | 一般 | 配列取得 | プロモーター配列 | GenomicFeatures(Lawrence 2013)(last modified 2013/10/10)
- イントロ | 一般 | 配列取得 | トランスクリプトーム配列 | biomaRt(Durinck
- イントロ | NGS | 様々なプラットフォーム(last modified 2013/06/12)
- イントロ | NGS | qPCRやmicroarrayなどとの比較(last modified 2010/12
- イントロ | NGS | Viewer(last modified 2014/01/29) NEW
- イントロ | NGS | 配列取得 | FASTQ or SRALite | 公共DBから(last mod
- イントロ | NGS | 配列取得 | FASTQ or SRALite | SRAdb(Zhu 2013)(las
- イントロ | NGS | アノテーション情報取得 | GFF/GTF形式ファイル(last m
- イントロ | NGS | アノテーション情報取得 | refFlat形式ファイル(last mod
- イントロ | NGS | アノテーション情報取得 | biomaRt(Durinck 2009)(last
- イントロ | NGS | アノテーション情報取得 | TranscriptDb | TxDb.*から(la
- イントロ | NGS | アノテーション情報取得 | TranscriptDb | GenomicFeatu
- イントロ | NGS | アノテーション情報取得 | TranscriptDb | GFF/GTF形式
- イントロ | NGS | 読み込み | FASTA形 | 基本情報を取得(last modifie
- イントロ | NGS | 読み込み | FASTA形 | description行の 記述を整形(l
- イントロ | NGS | 読み込み | FASTQ形式(last modified 2013/06/17)
- イントロ | NGS | 読み込み | FASTQ形式 | description行の 記述を整形(l
- イントロ | NGS | 読み込み | Illuminaの*_seq.txt(last modified 2013/06/1
- イントロ | NGS | 読み込み | Illuminaの*_qseq.txt(last modified 2013/06/
- イントロ | ファイル形式の変換 | について(last modified 2013/09/30)
- イントロ | ファイル形式の変換 | BAM --> BED(last modified 2013/10/2

multi-FASTAって何?

イントロ | NGS | 読み込み | FASTA形式 | 基本情報を取得 NEW

multi-fastaファイルを読み込んで、Total lengthやaverage lengthなどの各種情報取得を行うためのやり方を示します。
「ファイル」-「ディレクトリの変更」でファイルを保存したいディレクトリに移動し以下をコピペ。

1. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたmulti-fastaファイル(hoge4.fa)の場合:

```
in_f <- "hoge4.fa"                    #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(Biostrings)                   #パッケージの読み込み

#入力ファイルの読み込み
fasta <- readDNAStringSet(in_f, format="fasta")#in_fで指定したファイルの読み込み

#本番(基本情報取得)
Total_len <- sum(width(fasta))        #コンティグの「トータルの長さ」を取得
Number_of_contigs <- length(fasta)    #「コンティグ数」を取得
Average_len <- mean(width(fasta))     #コンティグの「平均長」を取得
Median_len <- median(width(fasta))    #コンティグの「中央値」を取得
Max_len <- max(width(fasta))          #コンティグの長さの「最大値」を取得
Min_len <- min(width(fasta))          #コンティグの長さの「最小値」を取得

#本番(N50情報取得)
sorted <- rev(sort(width(fasta)))     #長さ情報を降順にソートした結果をsortedに格納
N50 <- sorted[cumsum(sorted) >= Total_len/2][1]#「N50」(Total_lenの50%に達したときのコンティグ長)を取

#本番(GC含量情報取得)
```

multi-FASTAファイルからの各種情報抽出

FASTAフォーマット [編集]

FASTA では、シーケンスデータの記述形式として FASTAフォーマットという形式を使う。FASTAフォーマットはプレーンテキストである。1つのシーケンスのデータは、">" で始まる1行のヘッダ行と、2行目以降の実際のシーケンス文字列で構成される。ヘッダ行では、">" の次にシーケンスデータを識別するための文字列を記述し、続けてそのシーケンスデータを説明する文字列を記述する(両方とも省略してよい)。ヘッダ行の ">" と識別文字列の間にスペースを入れてはいけない。FASTAフォーマットの全ての行は、80文字未満とすることが推奨される。">" で始まる別の行が出現すると、そこでシーケンスデータが区切られ、別のシーケンスデータが始まる。

FASTA ファイルフォーマットの例を示す。

```
>gi|5524211|gb|AAD44166.1| cytochrom
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATA
EWIWGGFSVDKATLNRFFAFHFILPFTMVALAGVHL
LLILILLLLLLALLSPDMLGDPDNHMPADPLNTPLH
GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTW
IENY
```

hoge4.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
>contig_1
CGGACAGCTCCTCGGCATCCGGAT
>contig_2
GTCTGCCTCAAGCGCCCCAAGTGGGTTTGGAGGCCTAACATCGCAAGTCG
ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCT
GTC
>contig_3
TGTAGGAGAAGGGCGGTATCAGCGTCCACTTACACGATCCGTTACTAATT
GTATGAGGTCGGGCA
>contig_4
CGTGCTGATTCCACACAGCAGTAAACGCGGACCTCTACCTATGAACATG
```

Rでmulti-FASTAファイルを読み込んで自在に解析できます

コピー（CTRL+ALT+左クリック）&ペースト

イントロ | NGS | 読み込み | FASTA形式 | 基本情報を取得 NEW

multi-fastaファイルを読み込んで、Total lengthやaverage lengthなどの各種情報取得を行うためのやり方を示します。
「ファイル」-「ディレクトリの変更」でファイルを保存したいディレクトリに移動し以下をコピペ。

1. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたmulti-fastaファイル(hoge4.fa)の場合:

```
in_f <- "hoge4.fa"                    #入力ファイル
out_f <- "hoge1.txt"

#必要なパッケージをロー
library(Biostrings)

#入力ファイルの読み込み
fasta <- readDNAStrin

#本番(基本情報取得)
Total_len <- sum(widt
Number_of_contigs <-
Average_len <- mean(w
Median_len <- median(
Max_len <- max(width(
Min_len <- min(width(

#本番(N50情報取得)
sorted <- rev(sort(wi
N50 <- sorted[cumsum(sorted) >= Total_len/2][1]#「N

#本番(GC含量情報取得)
```

①

切り取り(T)
コピー(C)
貼り付け
すべて選択(A)
印刷(I)...
印刷プレビュー(N)...
Bing でマップ
Bing で翻訳
Google で検索
電子メール (Windows Live Hotmail)
すべてのアクセラレータ
Send to OneNote

hoge4.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
>contig_1
CGGACAGCTCCTCGGCATCCGGAT
>contig_2
GTCTGCCTCAAGCGCCCCAAGTGGGTTTGGAGGCCTAACATCGCAAGTCG
ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCT
GTC
>contig_3
TGTAGGAGAAGGGCGGTATCAGCGTCCACTTACACGATCCGTTACTAATT
GTATGAGGTCGGGCA
>contig_4
CGTGCTGATTCCACACAGCAGTAAACGCGGACCTCTACCTATGAACATG
```

RGui (64-bit)

ファイル 編集 閲覧 その他

R Console

```
> #GC含量(GC content)計算のと
> count <- alphabetFrequency
> CG <- rowSums(count[,2:3])
> ACGT <- rowSums(count[,1:4]
> GC_content <- sum(CG)/sum(A
>
> #出力用に結果をまとめて(w
> tmp <- NULL
> tmp <- rbind(tmp, c("Total length (bp)", Total_length))
> tmp <- rbind(tmp, c("Number of contigs", Number_of_contigs))
> tmp <- rbind(tmp, c("Average length", Average_length))
> tmp <- rbind(tmp, c("Median length", Median_length))
> tmp <- rbind(tmp, c("Max length", Max_length))
> tmp <- rbind(tmp, c("Min length", Min_length))
> tmp <- rbind(tmp, c("N50", N50))
> tmp <- rbind(tmp, c("GC content", GC_content))
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)#tmpの$
>
>
```

②

①一連のコマンド群をコピーして
②R Console画面上でペースト

hogeフォルダにhoge1.txtが作成されているはず

結果ファイルを眺めて動作確認

ID	Length
contig_1	24
contig_2	103
contig_3	65
contig_4	49

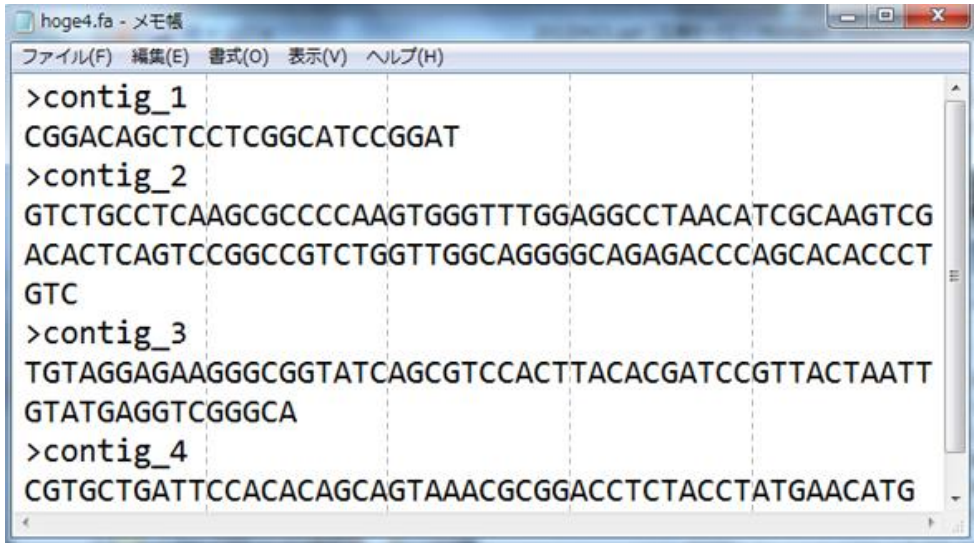
イントロ | NGS | 読み込み | FASTA形式 | 基本情報を取得 NEW

multi-fastaファイルを読み込んで、Total lengthやaverage lengthなどの各種情報取得を行うためのやり方を示します。
「ファイル」−「ディレクトリの変更」でファイルを保存したいディレクトリに移動し以下をコピペ。

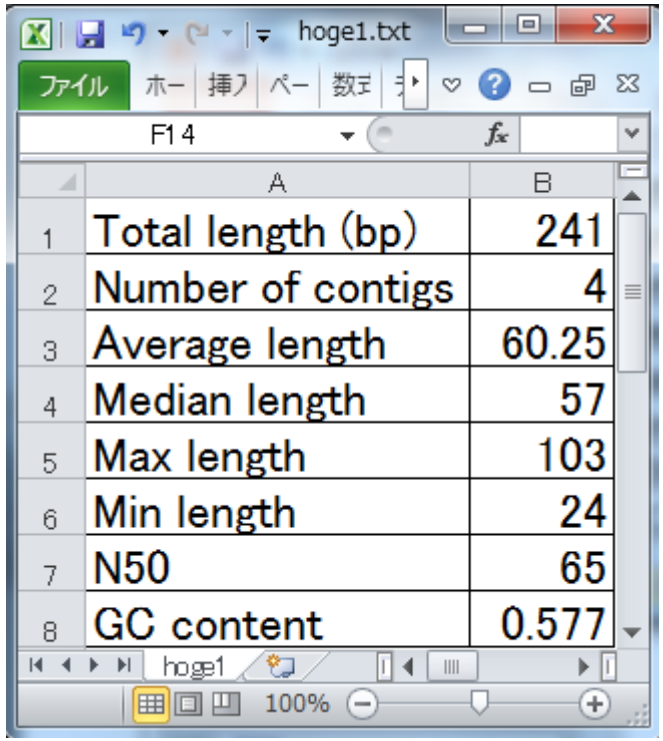
1. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたmulti-fastaファイル(hoge4.fa)の場合:

```
in_f <- "hoge4.fa"                    #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
```

入力: hoge4.fa

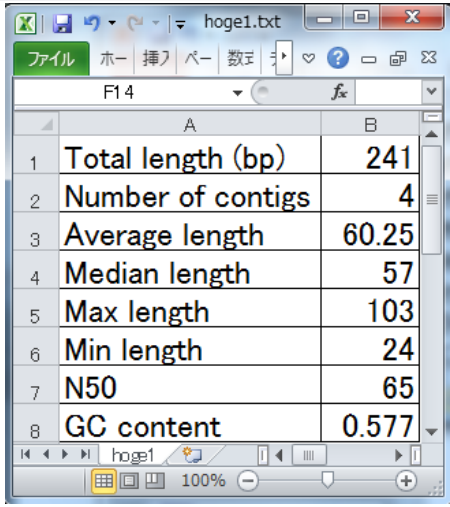


出力: hoge1.txt



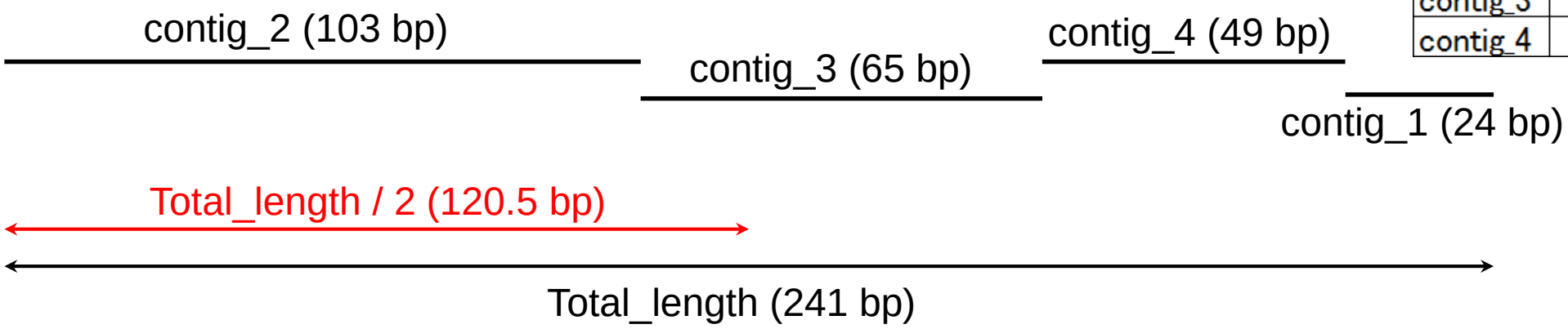
N50

- アセンブルがどれだけうまくいっているかを表す指標の一つ
  - 長いコンティグから足していってTotal_lengthの50%に達したときのコンティグの長さ



	A	B
1	Total length (bp)	241
2	Number of contigs	4
3	Average length	60.25
4	Median length	57
5	Max length	103
6	Min length	24
7	N50	65
8	GC content	0.577

ID	Length
contig_1	24
contig_2	103
contig_3	65
contig_4	49



averageだと外れ値の影響を受けやすく、medianだと短いコンティグが多くを占める場合に不都合らしい…。

情報抽出手順の一部

イントロ | NGS | 読み込み | FASTA形式 | 基本情報を取得 NEW

multi-fastaファイルを読み込んで、Total lengthやaverage lengthなどの各種情報取得を行うためのやり方を示します。
「ファイル」−「ディレクトリの変更」でファイルを保存したいディレクトリに移動し以下をコピペ。

1. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたmulti-fastaファイル(hoge4.fa)の場合:

```
in_f <- "hoge4.fa"                    #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(Biostrings)                   #パッケージの読み込み

#入力ファイルの読み込み
fasta <- readDNAStringSet(in_f, format="fasta")#in_fで指定したファイルの読み込み

#本番(基本情報取得)
Total_len <- sum(width(fasta))        #コンティグの「トータルの長さ」を取得
Number_of_contigs <- length(fasta)    #「コンティグ数」を
Average_len <- mean(width(fasta))     #コンティグの「平均
Median_len <- median(width(fasta))    #コンティグの「中央
Max_len <- max(width(fasta))          #コンティグの長さの
Min_len <- min(width(fasta))          #コンティグの長さの
```

R Console

```
> fasta
  A DNAStringSet instance of length 4
    width seq                                          names
[1]    24 CGGACAGCTCCTCGGCATCCGGAT                     contig_1
[2]   103 GTCTGCCTCAAGCGCC...CCCAGCACACCCTGTC          contig_2
[3]    65 TGTAGGAGAAGGGCGG...TGTATGAGGTCGGGCA          contig_3
[4]    49 CGTGCTGATTCCACAC...TCTACCTATGAACATG          contig_4
> width(fasta)
[1]  24 103  65  49
> sum(width(fasta))
[1] 241
>
```

width関数を使えば配列長情報を取り出せるようだ

参考

ID	Length
contig_1	24
contig_2	103
contig_3	65
contig_4	49

情報抽出手順の一部

R Console

```
> fasta[1]
  A DNAStringSet instance of length 1
    width seq                                         names
[1]    24 CGGACAGCTCCTCGGCATCCGGAT                    contig_1
> fasta[2]
  A DNAStringSet instance of length 1
    width seq                                         names
[1]   103 GTCTGCCTCAAGCGCC...CCCAGCACACCCTGTC        contig_2
> width(fasta) >= 50
[1] FALSE  TRUE  TRUE FALSE
> fasta[width(fasta) >= 50]
  A DNAStringSet instance of length 2
    width seq                                         names
[1]   103 GTCTGCCTCAAGCGCC...CCCAGCACACCCTGTC        contig_2
[2]    65 TGTAGGAGAAGGGCGG...TGTATGAGGTCGGGCA        contig_3
>
```

50 bp以上のコンティグからなる
サブセットの抽出ができそうだ!

コードの中身が分かると応用範囲が拡大

- 前処理 | クオリティチェック | 配列長分布を調べる (last modified 2013/06/18)
- 前処理 | フィルタリング | PHREDスコアが低い塩基をNに置換 (last modified 2013/06/15)
- 前処理 | フィルタリング | PHREDスコアが低い配列(リード)を除去 (last modified 2013/06/15)
- 前処理 | フィルタリング | ACGTのみからなる配列を抽出 (last modified 2013/06/18)
- 前処理 | フィルタリング | ACGT以外のcharacter"-"をNに変換 (last modified 2013/06/18)
- 前処理 | フィルタリング | ACGT以外の文字数が閾値以下の配列を抽出 (last modified 2013/09/27)
- 前処理 | フィルタリング | 重複のない配列セットを作成 (last modified 2013/06/18)
- 前処理 | フィルタリング | 指定した長さ以上の配列を抽出 (last modified 2013/06/18)
- 前処理 | フィルタリング | 指定した長さの範囲の配列を抽出 (last modified 2013/06/18)
- 前処理 | フィルタリング | 任意のIDを含む配列
- 前処理 | フィルタリング | Illuminaのpass filterin
- 前処理 | フィルタリング | GFF/GTF形式ファイル
- 前処理 | フィルタリング | 組合せ | ACGTのみ
- 前処理 | トリミング | ポリA配列除去 (last modi
- 前処理 | トリミング | アダプター配列除去(基礎
- 前処理 | トリミング | アダプター配列除去(組合
- 前処理 | トリミング | アダプター配列除去(giraf
- 前処理 | トリミング | 指定した末端塩基数だけ
- アセンブル | について (last modified 2011/07/2
- アセンブル | ゲノム用 (last modified 2013/06/1
- アセンブル | トランスクリプトーム(転写物)用 (l
- アセンブル | ゲノム既知で転写物構造推定用

指定した配列長以下のものを抽出したいときは「<=」とすればよい

前処理 | フィルタリング | 指定した長さ以上の配列を抽出 NEW

FASTA形式やFASTQ形式ファイルを入力として、指定した配列長以上の配列を抽出するやり方を示します。
「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. FASTA形式ファイル(hoge4.fa)の場合:

```
in_f <- "hoge4.fa"                                  #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
param <- 50                                         #配列長の閾値を指定

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

#入力ファイルの読み込み
fasta <- readDNAStringSet(in_f, format="fasta")#in_fで指定したファイルの読み込み
fasta                                               #確認してるだけです

#本番
obj <- as.logical(width(fasta) >= param)#条件を満たすかどうかを判定した結果をobjに格納
fasta <- fasta[obj]                                 #objがTRUEとなる要素のみ抽出した結果をfastaに格納
fasta                                               #確認してるだけです

#ファイルに保存
writeXStringSet(fasta, file=out_f, format="fasta", width=50)#fastaの中身を指定したファ
```

参考

イントロ | 一般 | 指定した範囲の配列を取得 NEW

single-FASTA形式やmulti-FASTA形式ファイルから様々な部分配列を取得するやり方を示します。
「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. (single-)FASTA形式ファイル(sample1.fasta)の場合:

任意の範囲 (始点が3, 終点が9)の配列を抽出し、得られた部分配列をFASTA形式ファイル(hoge1.txt)に出力するやり方です。

```
in_f <- "sample1.fasta"                    #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
param <- c(3, 9)                           #抽出したい範囲の始点と終点を指定

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

#入力ファイルの読み込み
fasta <- readDNAStringSet(in_f, format="fasta")#in_fで指定したファイルの読み込み
fasta                                      #確認してるだけです

#本番
fasta <- subseq(fasta, param[1], param[2])#paramで指定した始点と終点の範囲の配列を抽出し
fasta                                      #確認してるだけです

#ファイルに保存
writeXStringSet(fasta, file=out_f, format="fasta", width=50)#fastaの中身を指定したファイル名で保存
```

入力:sample1.fasta

```
>kadota
AGTGACGGTCTT
```

出力:hoge1.txt

```
>kadota
TGACGGT
```

1. (single-)FASTA形式ファイル(sample1.fasta)の場合:

任意の範囲(始点が3, 終点が9)の配列を抽出し、得られた部分配列をFASTA形式ファイル(hoge1.txt)に

```
in_f <- "sample1.fasta"                    #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
param <- c(3, 9)                           #抽出したい範囲の始点と終点を指定

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

#入力ファイルの読み込み
fasta <- readDNAStringSet(in_f, format="fasta")#in_fで指定したファイルの読み込み
fasta                                      #確認してるだけです

#本番
fasta <- subseq(fasta, param[1], param[2])#paramで指定した始点と終点の範囲の配列
fasta

#ファイルに保存
writeXStringSet(fasta, file
```

• イントロ | 一般 | 指定した範囲の配列を取得

参考

入力:sample1.fasta

```
>kadota
AGTGACGGTCTT
```

出力:hoge1.txt

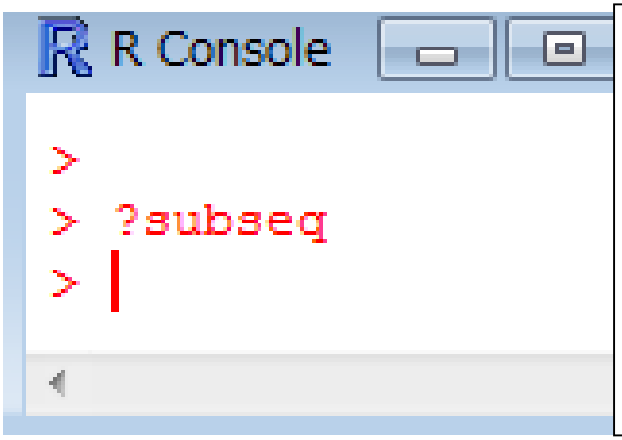
```
>kadota
TGACGGT
```

R Console

```
> fasta
  A DNAStringSet instance of length 1
    width seq                 names
[1]    12 AGTGACGGTCTT        kadota
> subseq(fasta, param[1], param[2])
  A DNAStringSet instance of length 1
    width seq                 names
[1]     7 TGACGGT             kadota
> param
[1] 3 9
> subseq(fasta, 3, 9)
  A DNAStringSet instance of length 1
    width seq                 names
[1]     7 TGACGGT             kadota
>
```

subseq関数は「塩基配列, start, end」という形式がデフォルトのようだ

関数の使用法について



XVector-class {IRanges} R Documentation

XVector objects

Description

The XVector virtual class is a general container for storing an "external vector". It inherits from the Vector, which has a very rich interface.

The following classes derive directly from the XVector class:

...

```
subseq(x4, start=10)
subseq(x4, start=-10)
subseq(x4, start=-20, end=-10)
subseq(x4, start=10, width=5)
subseq(x4, end=10, width=5)
subseq(x4, end=10, width=0)

x3[length(x3):1]
x3[length(x3):1, drop=FALSE]
```

[Package *IRanges* version 1.12.6 Index]

- **?関数名**で使用法を記したウェブページが開く
- ページの下のほうに、大抵の場合使用例が掲載されている
- 使用法既知の関数のマニュアルをいくつか読んで慣れておく

参考

原因既知状態で意図的にエラーを出す

R Console

```
> fasta
  A DNAStringSet instance of length 1
    width seq                                                names
[1]    12 AGTGACGGTCTT                                       kadota
> subseq(fasta, start=3, end=9)
  A DNAStringSet instance of length 1
    width seq                                                names
[1]     7 TGACGGT                                            kadota
> subseq(fasta, start=3, width=11)
 以下にエラー .Call2("solve_user_SEW", refwidths, start, end, width, transla$
  solving row 1: 'allow.nonnarrowing' is FALSE and the solved end (13) is > $
> subseq(fasta, start=3, width=10)
  A DNAStringSet instance of length 1
    width seq                                                names
[1]    10 TGACGGTCTT                                         kadota
>
```

入力：sample1.fasta

```
>kadota
AGTGACGGTCTT
```

出力：hoge1.txt

```
>kadota
TGACGGT
```

マニュアルの使用例をいくつか試して、ステップアップ

4. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたmulti-fastaファイル(hoge4.fa)の場合:

目的のaccession番号が複数ある場合に対応したものです。予め用意しておいた「1列目:accession, 2列目:start, 3列目:end位置」からなるリストファイル (list_sub2.txt) を読み込ませて、目的の配列のmulti-fastaファイルをゲットするやり方です。


```
in_f1 <- "hoge4.fa"                          #入力ファイル名を指定してin_f1に格納(multi-fastaファイル)
in_f2 <- "list_sub2.txt"                     #入力ファイル名を指定してin_f2に格納(リストファイル)
out_f <- "hoge4.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(Biostrings)                          #パッケージの読み込み

#入力ファイルの読み込み
fasta <- readDNAStringSet(in_f1, format="fasta")#in_f1で指定したファイルの読み込み
posi <- read.table(in_f2)                    #in_f2で指定したファイルの読み込み
fasta                                        #確認してるだけです

#本番
hoge <- NULL                                 #最終的に得る結果を格納するためのプレースホルダhogeを作成して
for(i in 1:nrow(posi)){                      #length(posi)回だけループを回す
    obj <- names(fasta) == posi[i,1]         #条件を満たすかどうかを判定した結果をobjに格納
    hoge <- append(hoge, subseq(fasta[obj], start=posi[i,2], end=posi[i,3]))#subseq関数を用いて
}
```

入力1: hoge4.fa

hoge4.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
>contig_1
CGGACAGCTCCTCGGCATCCGGAT
>contig_2
GTCTGCCTCAAGCGCCCCAAGTGGGTTTGGAGGCCTAACATCGCAAGTCG
ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCT
GTC
>contig_3
TGTAGGAGAAGGGCGGTATCAGCGTCCACTTACACGATCCGTTACTAATT
GTATGAGGTCGGGCA
>contig_4
CGTGCTGATTCCACACAGCAGTAAACGCGGACCTCTACCTATGAACATG
```

入力2: list_sub2.txt

contig_4	1	10
contig_2	3	8

出力:hog4.txt

```
>contig_4
CGTGCTGATT
>contig_2
CTGCCT
```

- 正規化 | サンプル間 | 3群間 | 複製あり | iDEGES/edgeR(Sun 2013)(last modified 2013/09/15)推奨
- 正規化 | サンプル間 | 3群間 | 複製あり | TMM(Robinson 2010)(last modified 2013/09/16)
- 解析 | 一般 | アラインメント(ペアワイズ;基本編1) (last modified 2010/6/8)
- 解析 | 一般 | アラインメント(ペアワイズ;基本編2) (last modified 2010/6/8)
- 解析 | 一般 | アラインメント(ペアワイズ;応用編) (last modified 2010/6/8)
- 解析 | 一般 | パターンマッチング (last modified 2013/06/19)
- 解析 | 一般 | GC含量 (GC contents) (last modified 2013/06/24)
- 解析 | 一般 | Sequence logos(Schneider 1990)(last modified 2012/06/27)
- 解析 | 一般 | 上流配列解析 | LDSS(Yamamoto 2007)(last modified 2012/07/17)
- 解析 | 一般 | 上流配列解析 | Relative Appearance Ratio(Yamamoto 2011)(last modified 2012/07/17)
- 解析 | 基礎 | 平均-分散プロット(Technical replicates)(last modified 2013/12/27)
- 解析 | 基礎 | 平均-分散プロット(Biological replicates)(last
- 解析 | クラスタリング | について(last modified 2014/02/0
- 解析 | クラスタリング | サンプル間 | hclust(last modified
- 解析 | クラスタリング | 遺伝子間 | MBCluster.Seq (Si 20
- 解析 | 発現変動 | ポアソン分布 | シミュレーションデータ(
- 解析 | 発現変動 | 負の二項分布 | シミュレーションデータ
- 解析 | 発現変動 | について(last modified 2013/08/29)
- 解析 | 発現変動 | 2群間 | 対応なし | について(last modif
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC (Su
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | edgeR (R

参考

- 解析 | 一般 | GC含量 (GC contents)

解析 | 一般 | GC含量 (GC contents)

multi-fasta形式ファイルを読み込んで配列ごとのGC含量 (GC contents)を出力するやり方を示します。出力ファイルは、「description」「CGの総数」「ACGTの総数」「配列長」「%GC含量」としています。尚、%GC含量は「CGの総数/ACGTの総数」で計算しています。
「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. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたmulti-fastaファイル(hoge4.fa)の場合:

```
in_f <- "hoge4.fa"                          #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(Biostrings)                         #パッケージの読み込み

#入力ファイルの読み込み
fasta <- readDNAStringSet(in_f, format="fasta")#in_fで指定したファイルの読み込み

#本番
hoge <- alphabetFrequency(fasta)            #A,C,G,T,..の数を各配列ごとにカウントした結果を
CG <- rowSums(hoge[,2:3])                   #C,Gの総数を計算してCGに格納
ACGT <- rowSums(hoge[,1:4])                 #A,C,G,Tの総数を計算してACGTに格納
GC_content <- CG/ACGT*100                   #%GC含量を計算してGC_contentに格納

#ファイルに保存
tmp <- cbind(names(fasta), CG, ACGT, width(fasta), GC_content)#ファイルに保存したい
colnames(tmp) <- c("description", "CG", "ACGT", "Length", "%GC_contents")#列名情報
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.names=T)#tmp
```

配列ごとのGC含量を計算したいとき

Contents（Rで…）

- ゲノム解析
  - アノテーションファイルを読み込んで目的のキーワードを含む行のみ抽出
  - multi-FASTAファイルを自在に解析
    - 配列長分布、GC含量、フィルタリング、部分配列の切り出しなど
    - **連続塩基の出現頻度（CpG）解析、ゲノム配列取得など**
- トランスクリプトーム解析
  - 研究目的別留意点：サンプル内とサンプル間の違い
  - マッピング → カウント情報取得
  - データを眺める：クラスタリングやM-A plot
  - 理想的な実験デザイン
  - なぜx倍発現変動という議論がだめなんですか？
  - モデルとか分布って、自分の解析結果にどういう影響を与えているの？
  - 多重比較問題：FDRって何？

ヒトゲノム中のCpG出現確率は低い

- 全部で16通りの2連続塩基の出現頻度分布を調べると、CGとなる確率の実測値(0.986%)は期待値(4.2%)よりもかなり低い
- 期待値
  - ゲノム中のGC含量を考慮した場合：約41%(A:0.295, C:0.205, G: 0.205, T:0.295)なので、0.205×0.205= 4.2%
  - ゲノム中のGC含量を考慮しない場合： 50%(A:0.25, C:0.25, G: 0.25, T:0.25)なので、0.25×0.25= 6.25%

2連続塩基の出現頻度：基本形

イントロ | 一般 | 2連続塩基の出現頻度情報を取得 NEW

multi-fasta形式ファイルを読み込んで、"AA", "AC", "AG", "AT", "CA", "CC", "CG", "CT", "GA", "GC", "GG", "GT", "TA", "TC", "TG", "TT"の計$4^2 = 16$通りの2連続塩基の出現頻度を調べるやり方を示します。
ヒトゲノムで"CG"の割合が期待値よりも低い(Lander et al., 2001; Saxonov et al., 2006)ですが、それを簡単に検証できます。
「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. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたmulti-fastaファイル(hoge4.fa)の場合:

タイトル通りの出現頻度です。

```
in_f <- "hoge4.fa"                          #入力ファイル名を指定してin_fに格
out_f <- "hoge1.txt"                        #出力ファイル名を指定してout_fに

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

#入力ファイルの読み込み
fasta <- readDNAStringSet(in_f, format="fasta")#in_fで指定したファイルの

#本番
out <- dinucleotideFrequency(fasta)         #2連続塩基の出現頻度情報をoutに格

#ファイルに保存
tmp <- cbind(names(fasta), out)             #最初の列にID情報、そのあとに出現
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)#tmpの中身を指定したファイル名
```

hoge4.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
>contig_1
CGGACAGCTCCTCGGCATCCGGAT
>contig_2
GTCTGCCTCAAGCGCCCCAAGTGGGTTTGGAGGCCTAACATCGCAAGTCG
ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCT
GTC
>contig_3
TGTAGGAGAAGGGCGGTATCAGCGTCCACTTACACGATCCGTTACTAATT
GTATGAGGTCGGGCA
>contig_4
CGTGCTGATTCCACACAGCAGTAAACGCGGACCTCTACCTATGAACATG
```

出力：hoge1.txt

	AA	AC	AG	AT	CA	CC	CG	CT	GA	GC	GG	GT	TA	TC	TG	TT
contig_1	0	1	1	2	2	2	3	2	2	2	3	0	0	3	0	0
contig_2	4	6	9	1	11	11	5	6	4	9	10	8	1	8	6	3
contig_3	2	4	5	4	4	2	5	2	4	3	7	6	6	4	3	3
contig_4	3	6	2	3	5	3	3	4	3	3	1	2	3	2	4	1

2連続塩基の出現確率：基本形

2. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたmulti-fastaファイル(hoge4.fa)の場合:

出現頻度ではなく、出現確率を得るやり方です。

```
in_f <- "hoge4.fa"                    #入力ファイル名を指定してin_fに格納
out_f <- "hoge2.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(Biostrings)                   #パッケージの読み込み

#入力ファイルの読み込み
fasta <- readDNAStringSet(in_f, format="fasta")#in_fで指定したファイル(

#本番
out <- dinucleotideFrequency(fasta, as.prob=T)#2連続塩基の出現確率情報を

#ファイルに保存
tmp <- cbind(names(fasta), out)       #最初の列にID情報、そのあとに出現
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)#tmp
```

hoge4.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
>contig_1
CGGACAGCTCCTCGGCATCCGGAT
>contig_2
GTCTGCCTCAAGCGCCCCAAGTGGGTTTGGAGGCCTAACATCGCAAGTCG
ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCT
GTC
>contig_3
TGTAGGAGAAGGGCGGTATCAGCGTCCACTTACACGATCCGTTACTAATT
GTATGAGGTCGGGCA
>contig_4
CGTGCTGATTCCACACAGCAGTAAACGCGGACCTCTACCTATGAACATG
```

出力：hoge2.txt

	AA	AC	AG	AT	CA	CC	CG	CT	GA	GC	GG	GT	TA	TC	TG	TT
contig_1	0.0%	4.3%	4.3%	8.7%	8.7%	8.7%	13.0%	8.7%	8.7%	8.7%	13.0%	0.0%	0.0%	13.0%	0.0%	0.0%
contig_2	3.9%	5.9%	8.8%	1.0%	10.8%	10.8%	4.9%	5.9%	3.9%	8.8%	9.8%	7.8%	1.0%	7.8%	5.9%	2.9%
contig_3	3.1%	6.3%	7.8%	6.3%	6.3%	3.1%	7.8%	3.1%	6.3%	4.7%	10.9%	9.4%	9.4%	6.3%	4.7%	4.7%
contig_4	6.3%	12.5%	4.2%	6.3%	10.4%	6.3%	6.3%	8.3%	6.3%	6.3%	2.1%	4.2%	6.3%	4.2%	8.3%	2.1%

2連続塩基の出現確率:ヒトゲノム

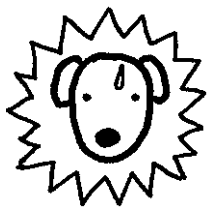
5. BSgenomeパッケージ中のヒトゲノム配列("BSgenome.Hsapiens.UCSC.hg19")の場合:

出現頻度ではなく、出現確率

```
out_f <- "hoge5.txt"
param <- "BSgenome.H

#必要なパッケージをロー
library(Biostrings)
library(param, chara

#前処理(paramで指定し
tmp <- ls(paste("pac
genome <- eval(parse
fasta <- getSeq(geno
names(fasta) <- seqn
fasta

#本番
out <- dinucleotideF

#ファイルに保存
tmp <- cbind(names(f
write.table(tmp, out
```

出力:hoge5.txt

	AA	AC	AG	AT	CA	CC	CG	CT	GA	GC	GG	GT	TA	TC	TG	TT
chr1	9.5%	5.0%	7.1%	7.5%	7.3%	5.4%	1.0%	7.1%	6.0%	4.4%	5.4%	5.0%	6.4%	6.0%	7.3%	9.5%
chr2	10.0%	5.0%	7.0%	7.9%	7.2%	5.0%	0.9%	7.0%	5.9%	4.1%	5.0%	5.0%	6.7%	5.9%	7.2%	10.0%
chr3	10.1%	5.0%	6.9%	8.0%	7.2%	4.9%	0.8%	6.9%	5.9%	4.0%	4.9%	5.0%	6.9%	5.9%	7.2%	10.2%
chr4	10.6%	5.0%	6.7%	8.5%	7.1%	4.5%	0.8%	6.7%	5.8%	3.8%	4.5%	5.0%	7.4%	5.8%	7.1%	10.6%
chr5	10.2%	5.0%	6.9%	8.1%	7.2%	4.8%	0.8%	6.9%	5.9%	4.0%	4.8%	5.0%	7.0%	5.9%	7.2%	10.2%
chr6	10.2%	5.0%	6.9%	8.1%	7.2%	4.9%	0.9%	6.9%	5.9%	4.0%	4.9%	5.0%	6.9%	5.9%	7.2%	10.2%
chr7	9.8%	5.0%	7.0%	7.8%	7.2%	5.2%	1.0%	7.0%	5.9%	4.3%	5.2%	5.0%	6.6%	5.9%	7.2%	9.9%
chr8	10.0%	5.1%	6.9%	7.9%	7.2%	5.0%	0.9%	6.9%	5.9%	4.1%	5.0%	5.0%	6.7%	5.9%	7.2%	10.0%
chr9	9.7%	5.0%	7.0%	7.6%	7.3%	5.3%	1.0%	7.0%	5.9%	4.3%	5.3%	5.0%	6.4%	5.9%	7.3%	9.7%
chr10	9.6%	5.0%	7.0%	7.5%	7.3%	5.4%	1.0%	7.1%	6.0%	4.4%	5.4%	5.1%	6.3%	6.0%	7.3%	9.6%
chr11	9.5%	5.0%	7.1%	7.6%	7.3%	5.4%	1.0%	7.1%	6.0%	4.4%	5.4%	5.0%	6.4%	6.0%	7.3%	9.5%
chr12	9.8%	5.0%	7.0%	7.7%	7.2%	5.2%	1.0%	7.0%	5.9%	4.2%	5.2%	5.0%	6.6%	5.9%	7.2%	9.8%
chr13	10.5%	5.0%	6.7%	8.5%	7.1%	4.6%	0.8%	6.7%	5.8%	3.8%	4.6%	5.0%	7.3%	5.8%	7.1%	10.5%
chr14	9.7%	5.0%	7.0%	7.7%	7.2%	5.2%	1.0%	7.0%	5.9%	4.3%	5.2%	5.1%	6.6%	5.9%	7.3%	9.9%
chr15	9.4%	5.1%	7.1%	7.3%	7.3%	5.5%	1.1%	7.2%	6.0%	4.5%	5.5%	5.1%	6.2%	6.0%	7.3%	9.4%
chr16	8.6%	5.1%	7.3%	6.7%	7.5%	6.2%	1.4%	7.3%	6.1%	5.0%	6.2%	5.1%	5.4%	6.1%	7.6%	8.6%
chr17	8.4%	5.0%	7.4%	6.4%	7.4%	6.5%	1.5%	7.4%	6.0%	5.2%	6.5%	5.0%	5.3%	6.1%	7.4%	8.5%
chr18	10.1%	5.0%	6.9%	8.1%	7.2%	4.9%	0.9%	6.9%	5.9%	4.1%	4.9%	5.1%	6.9%	5.9%	7.2%	10.1%
chr19	7.6%	5.1%	7.4%	5.7%	7.6%	7.2%	1.9%	7.4%	6.1%	5.7%	7.3%	5.1%	4.5%	6.1%	7.6%	7.6%
chr20	8.7%	5.0%	7.3%	6.8%	7.5%	6.0%	1.2%	7.3%	6.0%	4.9%	6.1%	5.1%	5.6%	6.1%	7.6%	8.9%
chr21	9.9%	5.1%	6.9%	7.8%	7.3%	5.2%	1.1%	6.8%	5.9%	4.3%	5.2%	5.1%	6.6%	5.9%	7.3%	9.8%
chr22	7.6%	5.1%	7.5%	5.8%	7.7%	7.1%	1.7%	7.5%	6.1%	5.7%	7.1%	5.1%	4.6%	6.1%	7.7%	7.6%
chrX	10.1%	5.0%	6.8%	8.2%	7.2%	4.9%	0.8%	6.8%	5.9%	3.9%	4.9%	5.1%	7.0%	5.9%	7.2%	10.2%
chrY	9.9%	5.1%	6.8%	8.1%	7.3%	4.9%	0.8%	6.8%	6.0%	3.9%	4.9%	5.2%	6.7%	6.0%	7.5%	10.0%



イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome

BSgenomeパッケージを用いて様々な生物種のゲノム配列を取得するやり方を示します。ミヤマハタザオ (A. lyrata)、セイヨウミツバチ (A. mellifera)、シロイヌナズナ(A.thaliana)、ウシ(B.taurus)、線虫(C.elegans)、犬(C.familiaris)、キイロショウジョウバエ(D.melanogaster)、ゼブラフィッシュ(D.rerio)、大腸菌(E.coli)、イトヨ(G.aculeatus)、セキショクヤケイ(G.gallus)、ヒト(H.sapiens)、アカゲザル(M.mulatta)、マウス(M.musculus)、チンパンジー(P.troglodytes)、ラット(R.norvegicus)、出芽酵母(S.cerevisiae)、トキソプラズマ(T.gondii)と実に様々な生物種が利用可能であることがわかります。

getSeq関数はBSgenomeオブジェクト中の「single sequences」というあたりにリストアップされているchr...というものを全て抽出しています。したがって、例えばマウスゲノムは「chr1」以外に「chr1_random」や「chrUn_random」なども等価に取扱っている点に注意してください。

「ファイル」-「ディレクトリの変更」でファイルを保存したいディレクトリに移動し以下をコピペ。

1. 利用可能な生物種とRにインストール済みの生物種をリストアップしたい場合:

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

#本番（利用可能なリストアップ；インストール済みとは限らない）
available.genomes()                        #このパッケージ中で利用可能なゲノムをリストアップ

#本番（インストール済みの生物種をリストアップ）
installed.genomes()                        #インストール済みの生物種をリストアップ

#後処理（パッケージ名でだいたいわかるがproviderやversionを分割して表示したい場合）
installed.genomes(splitNameParts=TRUE) #インストール済みの生物種をリストアップ
```

様々な生物種のゲノム配列がRのパッケージとして提供されています

イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome

BSgenomeパッケージを用いて様々な生物種のゲノム配列を取得するやり方を示します。ミヤマハタザオ (A. lyrata)、セイヨウミツバチ (A. mellifera)、シロイヌナズナ(A.thaliana)、ウシ(B.taurus)、線虫(C.elegans)、犬(C.familiaris)、キイロショウジョウバエ(D.melanogaster)、ゼブラフィッシュ(D.rerio)、大腸菌(E.coli)、イトヨ(G.aculeatus)、セキショクヤケイ(G.gallus)、ヒト(H.sapiens)、アカゲザル(M.mulatta)、マウス(M.musculus)、チンパンジー(P.troglodytes)、ラット(R.norvegicus)、出芽酵母(S.cerevisiae)、トキソプラズマ(T.gondii)と実に様々な生物種が利用可能であることがわかります。

getSeq関数はBSgenomeオブジェクト中の「single sequen
ています。したがって、例えばマウスゲノムは「chr1」以
注意してください。

「ファイル」−「ディレクトリの変更」でファイルを保存した

1. 利用可能な生物種とRにインストール済みの生物種

```
#必要なパッケージをロード
library(BSgenome)

#本番（利用可能なリストアップ；インストール
available.genomes()

#本番（インストール済みの生物種をリストアッ
installed.genomes()

#後処理（パッケージ名でだいたいわかるがprov
installed.genomes(splitNameParts=TRUE)
```

R Console

```
[24] "BSgenome.Hsapiens.UCSC.hg18"
[25] "BSgenome.Hsapiens.UCSC.hg19"
[26] "BSgenome.Mmulatta.UCSC.rheMac2"
[27] "BSgenome.Mmulatta.UCSC.rheMac3"
[28] "BSgenome.Mmusculus.UCSC.mm10"
[29] "BSgenome.Mmusculus.UCSC.mm8"
[30] "BSgenome.Mmusculus.UCSC.mm9"
[31] "BSgenome.Osativa.MSU.MSU7"
[32] "BSgenome.Ptroglodytes.UCSC.panTro2"
[33] "BSgenome.Ptroglodytes.UCSC.panTro3"
[34] "BSgenome.Rnorvegicus.UCSC.rn4"
[35] "BSgeno
[36] "BSgeno
[37] "BSgeno
[38] "BSgeno
[39] "BSgeno
>
> #本番(インストール済みの生物種をリストアッ$
> installed.genomes()                    #イ$
[1] "BSgenome.Celegans.UCSC.ce2"
[2] "BSgenome.Drerio.UCSC.danRer7"
[3] "BSgenome.Ecoli.NCBI.20080805"
[4] "BSgenome.Hsapiens.UCSC.hg19"
[5] "BSgenome.Hsapiens.UCSC.hg19.Rb wtie"
[6] "BSgenome.Mmusculus.UCSC.mm9"
[7] "BSgenome.Scerevisiae.UCSC.sacCer2"
>
```

ヒトゲノム(BSgenome.Hsapiens.UCSC.hg19)の2連続塩基出現頻度計算ができたのは、このパッケージをインストール済みだからです

イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome

• イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome 参考

BSgenomeパッケージを用いて様々な生物種のゲノム配列を取得するやり方を示します。ミヤマハタザオ (A. lyrata)、セイヨウミツバチ (A. mellifera)、シロイヌナズナ(A.thaliana)、ウシ(B.taurus)、線虫(C.elegans)、犬(C.familiaris)、キイロショウジョウバエ(D.melanogaster)、ゼブラフィッシュ(D.rerio)、大腸菌(E.coli)、イトヨ(G.aculeatus)、セキショクヤケイ(G.gallus)、ヒト(H.sapiens)、アカゲザル(M.mulatta)、マウス(M.musculus)、チンパンジー(P.troglodytes)、ラット(R.norvegicus)、出芽酵母(S.cerevisiae)、トキソプラズマ(T.gondii)と実に様々な生物種が利用可能であることがわかります。

getSeq関数はBSgenomeオブジェクト中の「single sequen
ています。したがって、例えばマウスゲノムは「chr1」以
注意してください。
「ファイル」-「ディレクトリの変更」でファイルを保存した

1. 利用可能な生物種とRにインストール済みの生物種

```
#必要なパッケージをロード
library(BSgenome)

#本番（利用可能なリストアップ；インストール
available.genomes()

#本番（インストール済みの生物種をリストアッ
installed.genomes()

#後処理（パッケージ名でだいたいわかるがprov
installed.genomes(splitNameParts=TRUE)
```

R Console

```
[24] "BSgenome.Hsapiens.UCSC.hg18"
[25] "BSgenome.Hsapiens.UCSC.hg19"
[26] "BSgenome.Mmulatta.UCSC.rheMac2"
[27] "BSgenome.Mmulatta.UCSC.rheMac3"
[28] "BSgenome.Mmusculus.UCSC.mm10"
[29] "BSgenome.Mmusculus.UCSC.mm8"
[30] "BSgenome.Mmusculus.UCSC.mm9"
[31] "BSgenome.Osativa.MSU.MSU7"
[32] "BSgenome.Ptroglodytes.UCSC.panTro2"
[33] "BSgenome.Ptroglodytes.UCSC.panTro3"
[38] "BSgenome.Scerevisiae.UCSC.sacCer3"
[39] "BSgenome.Tgondii.ToxoDB.7.0"
>
> #本番（インストール済みの生物種をリストアッ$
> installed.genomes()                    #イ$
[1] "BSgenome.Celegans.UCSC.ce2"
[2] "BSgenome.Drerio.UCSC.danRer7"
[3] "BSgenome.Ecoli.NCBI.20080805"
[4] "BSgenome.Hsapiens.UCSC.hg19"
[5] "BSgenome.Hsapiens.UCSC.hg19.Rbowtie"
[6] "BSgenome.Mmusculus.UCSC.mm9"
[7] "BSgenome.Scerevisiae.UCSC.sacCer2"
>
```

もしゼブラフィッシュ(BSgenome.Drerio.UCSC.danRer7)
ゲノムがインストールされていなければ...

2. ゼブラフィッシュ("BSgenome.Drerio.UCSC.danRer7")のゲノム情報をRにインストールしたい場合:

400MB程度あります...

```
param <- "BSgenome.Drerio.UCSC.danRer7"#パッケージ名を指定

#本番
source("http://bioconductor.org/biocLite.R")#おまじない
biocLite(param)                          #おまじない

#後処理（インストール済みの生物種をリストアップ）
installed.genomes()                      #インストール済みの生物種をリストアップ
```

3. インストール済みのゼブラフィッシュのゲノム配列をmulti-fastaファイルで保存したい場合:

1.4GB程度のファイルが生成されます...

```
out_f <- "hoge3.txt"                     #出力ファイル名を指定してout_fに格納
param <- "BSgenome.Drerio.UCSC.danRer7"#パッケージ名を指定

#必要なパッケージをロード
library(param, character.only=T)         #paramで指定したパッケージの読み込み

#前処理(paramで指定したパッケージ中のオブジェクト名をgenomeに統一)
#tmp <- unlist(strsplit(param, ".", fixed=TRUE))[2]#paramで指定した文字列からオブジェクト名を取得
tmp <- ls(paste("package", param, sep=":"))#paramで指定したパッケージで利用可能なオブジェクト名を
genome <- eval(parse(text=tmp))          #文字列tmpをRオブジェクトとしてgenomeに格納(パッケージ中に
genome                                   #確認してるだけです

#本番
fasta <- getSeq(genome)                  #ゲノム塩基配列情報を抽出した結果をfastaに格納
names(fasta) <- seqnames(genome)         #description情報を追加している

#ファイルに保存
writeXStringSet(fasta, file=out_f, format="fasta", width=50)#fastaの中身を指定したファイル名で保
```

• イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome 参考

available.genomes() でリストアップされているパッケージ名を指定可能です

multi-FASTAファイルとして保存したい場合

5. インストール済みのヒト("BSgenome.Hsapiens.UCSC.hg19")のゲノム配列をmulti-fastaファイルで保存したい場合:

3.0GB程度のファイルが生成されます…。ヒトゲノムは、まだ完全に22本の常染色体とX, Y染色体の計24本になっているわけではないことがわかります。

```
out_f <- "hoge5.txt"                          #出力ファイル名を指定してout_fに格納
param <- "BSgenome.Hsapiens.UCSC.hg19" #パッケージ名を指定

#必要なパッケージをロード
library(param, character.only=T)        #paramで指定したパッケージの

#前処理(paramで指定したパッケージ中のオブジェクト名をgenomeに統一)
#tmp <- unlist(strsplit(par
tmp <- ls(paste("package",
genome <- eval(parse(text=t
genome

#本番
fasta <- getSeq(genome)
names(fasta) <- seqnames(ge

#ファイルに保存
writeXStringSet(fasta, file
```

フリーズするので、得られたhoge5.txtをテキストエディタで開くことはやめましょう

R Console

```
|   chrUn_gl000246
|   chrUn_gl000247
|   chrUn_gl000248
|   chrUn_gl000249
|
| multiple sequences (see '?mseqnam
|   upstream1000  upstream2000  ups
|
| (use the '$' or '[[' operator to
| given sequence)
>
> #本番
> fasta <- getSeq(genome)                    #ゲノム塩基配列情報を抽出し$
> names(fasta) <- seqnames(genome)           #description情報を追加して$
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=50)#fasta$
>
```

① ② ③ ④ ⑤

内部的には、
①writeXStringSet関数を用いて、
②**fasta**オブジェクトを、
③out_fで指定したファイル名で、
④FASTA形式で、
⑤1行あたりの文字数を50文字にして保存しています。

fastaオブジェクトの中身

5. インストール済みのヒト("BSgenome.Hsapiens.UCSC.hg19")のゲノム配列をmulti-fastaファイルで保存したい場合:

3.0GB程度のファイルが生成されます…。ヒトゲノムは、まだ完全に22本の常染色体とX, Y染色体の計24本になっているわけではないことがわかります。

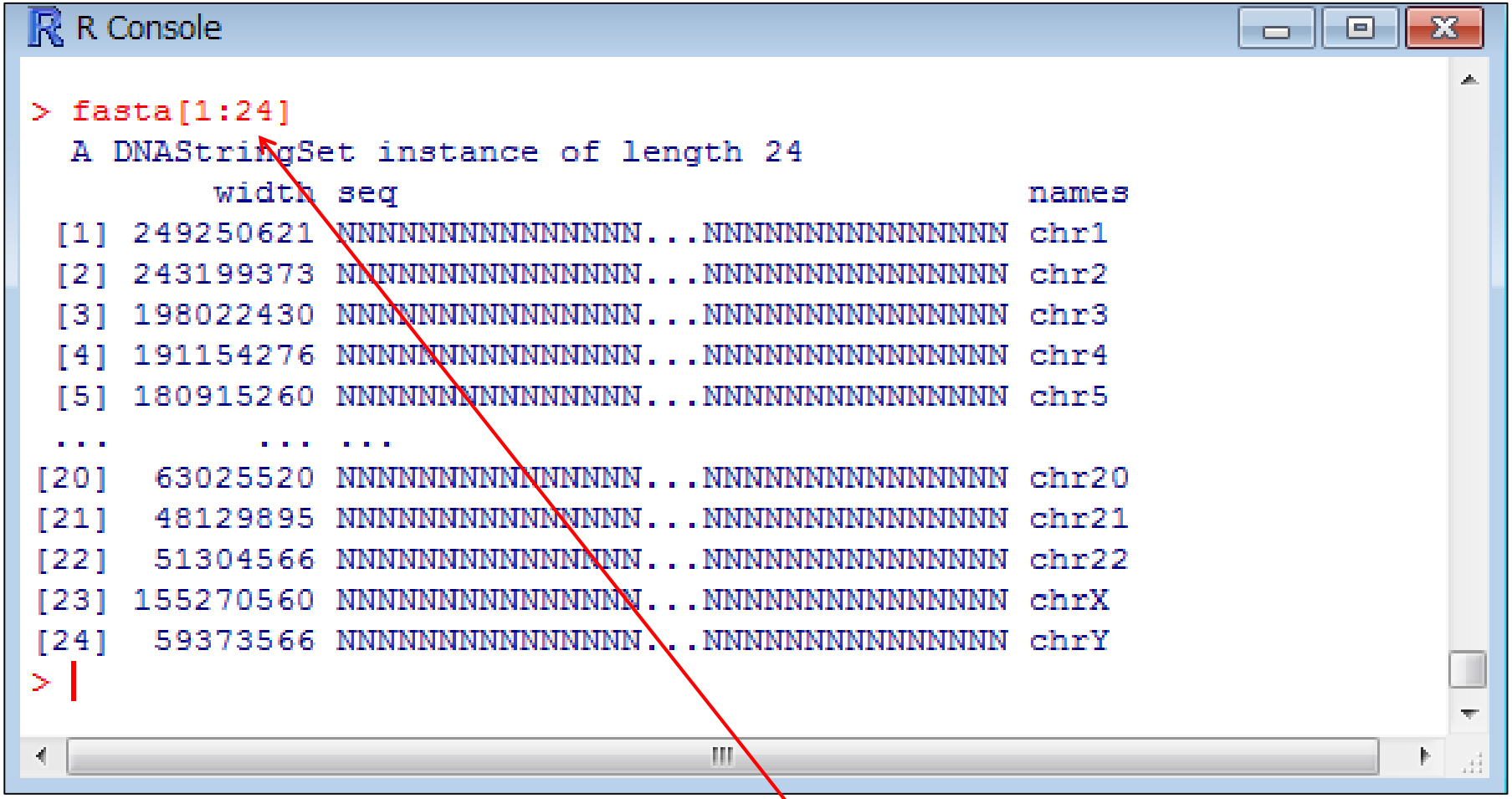
```
out_f <- "hoge5.txt"                    #出力ファイル名を指定してout_fに格納
param <- "BSgenome.Hsapiens.UCSC.hg19" #パッケージ名を指定

#必要なパッケージをロード
library(param, character.only=T)        #paramで指定したパッケージの読み込み

#前処理(paramで指定したパッケージ中のオブジェクト名をgenomeに統一)
#tmp <- unlist(strsplit(param, "
tmp <- ls(paste("package", param,
genome <- eval(parse(text=tmp))
genome

#本番
fasta <- getSeq(genome)
names(fasta) <- seqnames(genome)

#ファイルに保存
writeXStringSet(fasta, file=out_f
```

Rのほうが全体像の俯瞰が容易ですよね

R Console

```
> fasta
  A DNAStringSet instance of length 93
         width seq                                          names
  [1] 249250621 NNNNNNNNNNNNNNN...NNNNNNNNNNNNNNN chr1
  [2] 243199373 NNNNNNNNNNNNNNN...NNNNNNNNNNNNNNN chr2
  [3] 198022430 NNNNNNNNNNNNNNN...NNNNNNNNNNNNNNN chr3
  [4] 191154276 NNNNNNNNNNNNNNN...NNNNNNNNNNNNNNN chr4
  [5] 180915260 NNNNNNNNNNNNNNN...NNNNNNNNNNNNNNN chr5
  ...       ... ...
 [89]     36651 GATCAGATAGGCTTT...TTTCAGAATATGATC chrUn_gl000245
 [90]     38154 GATCTTAAGCCTTTG...CGAGGCGAGTGGATC chrUn_gl000246
 [91]     36422 GATCTAAGTTTGATT...GCTTTTCCCAAGATC chrUn_gl000247
 [92]     39786 GATCTGTCATTGTCT...TTGATACAGTTGATC chrUn_gl000248
 [93]     38502 GATCACCAAGGCTGG...AGTAGAATCTGGATC chrUn_gl000249
>
```

fastaオブジェクトの中身



最初の24個分を表示させたい場合

2連続塩基の出現確率:ヒトゲノムファイル

5. インストール済みのヒト("BSgenome.Hsapiens.UCSC.hg19")のゲノム配列をmulti-fastaファイルで保存したい場合:

3.0GB程度のファイルが生成されます...。ヒトゲノムは、まだ完全に22本の常染色体とX, Y染色体の計24本になっているわけではないことがわかります。

```
out_f <- "hoge5.txt"                    #出力ファイル名を指定してout_fに格納
param <- "BSgenome.Hsapiens.UCSC.hg19" #パッケージ名を指定

#必要なパッケージをロード
library(param, character.only=T)        #paramで指定したパッケージの読み込み

#前処理(paramで指定したパッケージ中のオブジェクト名をgenomeに統一)
#tmp <- unlist(strsplit(param, ".", fixed=TRUE))[2]#paramで指定した文字列
tmp <- ls(paste("package", param, sep=":"))#paramで指定したパッケージで利用可能なオブジェクト名を
genome <- eval(parse(text=tmp))         #文字列tmpをRオブジェクトとしてgenomeに格納(パッケージ中に
genome                                  #確認してるだけです

#本番
fasta <- getSeq(genome)
names(fasta) <- seqnames(

#ファイルに保存
writeXStringSet(fasta, fi
```

ヒトゲノムファイルhoge5.txtを入力ファイルとして与えるやり方でもよい

2. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたmulti-fastaファイル(hoge4.fa)の場合:

出現頻度ではなく、出現確率を得るやり方です。

```
in_f <- "hoge4.fa"                      #入力ファイル名を指定してin_fに格納
out_f <- "hoge2.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(Biostrings)                     #パッケージの読み込み

#入力ファイルの読み込み
fasta <- readDNAStringSet(in_f, format="fasta")#in_fで指定したファイルの読み込み

#本番
out <- dinucleotideFrequency(fasta, as.prob=T)#2連続塩基の出現確率情報をoutに格納

#ファイルに保存
tmp <- cbind(names(fasta), out)         #最初の列にID情報、そのあとに出現頻度情報のoutを結合したtr
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)#tmpの中身を指定したファイルそ
```

パッケージって何？

R Console

```
R version 3.0.2 (2013-09-25) -- "Frisbee Sailing"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R は、自由なソフトウェアであり、「完全に無保証」です。
一定の条件に従えば、自由にこれを再配布することができます。
配布条件の詳細に関しては、'license()' あるいは 'licence()' と入力してくださ$

R は多くの貢献者による共同プロジェクトです。
詳しくは 'contributors()' と入力してください。
また、R や R のパッケージを出版物で引用する際の形式については
'citation()' と入力してください。

'demo()' と入力すればデモをみることができます。
'help()' とすればオンラインヘルプが出ます。
'help.start()' で HTML ブラウザによるヘルプがみられます。
'q()' と入力すれば R を終了します。

> ?subseq
No documentation for 'subseq' in specified packages and libraries:
you could try '??subseq'
> ?dinucleotideFrequency
No documentation for 'dinucleotideFrequency' in specified packages and lib$
you could try '??dinucleotideFrequency'
>
```

Rを再起動した状態で**?関数名**と打ち込んでも、使用法を記したウェブページが開かずにエラーが出ることがあります

パッケージって何？

2. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたmulti-fastaファイル(hoge4.fa)の場合:

出現頻度ではなく、出現確率を得るやり方です。

```
in_f <- "hoge4.fa"
out_f <- "hoge2.txt"

#必要なパッケージをロード
library(Biostrings)

#入力ファイルの読み込み
fasta <- readDNAStringSet(in_f, for

#本番
out <- dinucleotideFrequency(fasta,

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

R Console

```
> library(Biostrings)
要求されたパッケージ BiocGenerics をロード中です
要求されたパッケージ parallel をロード中です

次のパッケージを付け加えます: 'BiocGenerics'

以下のオブジェクトはマスクされています (from 'package:parallel') $

    clusterApply, cluster
   clusterExport, cluster
   parLapplyLB, parRapply

以下のオブジェクトはマスクされてい

    xtabs

以下のオブジェクトはマスクされています (from 'package:base') :

    anyDuplicated, append, as.data.frame, as.vector, cbind,
   colnames, duplicated, eval, evalq, Filter, Find, get, intersect,
   is.unsorted, lapply, Map, mapply, match, mget, order, paste,
   pmax, pmax.int, pmin, pmin.int, Position, rank, rbind, Reduce,
   rep.int, rownames, sapply, setdiff, sort, table, tapply, union,
   unique, unlist

要求されたパッケージ IRanges をロード中です
要求されたパッケージ XVector をロード中です
> ?dinucleotideFrequency
starting httpd help server ... done
>
```

*Biostrings*というパッケージをlibrary関数を用いて読み込むことによって、dinucleotideFrequencyのような*Biostrings*が提供する関数群を利用できるんです

参考

5. BSgenomeパッケージ中のヒトゲノム配列("BSgenome.Hsapiens.UCSC.hg19")の場合:

出現頻度ではなく、出現確率を得るやり方です。

```
out_f <- "hoge5.txt"                          #出力ファ
param <- "BSgenome.Hsapiens.UCSC.hg19" #パッケー

#必要なパッケージをロード
library(Biostrings)                           #パッケー
library(param, character.only=T)              #paramで指

#前処理(paramで指定したパッケージ中のオブジェクト名を
tmp <- ls(paste("package", param, sep=":"))#para
genome <- eval(parse(text=tmp))               #文字列tmp
fasta <- getSeq(genome)                       #ゲノム塩基
names(fasta) <- seqnames(genome)              #descript
fasta                                         #確認して

#本番
out <- dinucleotideFrequency(fasta, as.prob=T)#2

#ファイルに保存
tmp <- cbind(names(fasta), out)               #最初の列
write.table(tmp, out_f, sep="\t", append=F, quot
```

- BioconductorのBiostringsのwebページ
- BioconductorのBSgenomeのwebページ
- Bird AP., Nucleic Acids Res., 1980
- Lander et al., Nature, 2001
- Saxonov et al., Proc Natl Acad Sci U S A., 2006

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Home | Install | Help | Develo

Search:

Home » Bioconductor 2.12 » Software Packages » Biostrings

Biostrings

String objects representing biological sequences, and matching algorithms

Bioconductor version: Release (2.12)

Memory efficient string containers, string matching algorithms, and other utilities, for fast manipulation of large biological sequences or sets of sequences.

Author: H. Pages, P. Aboyoun, R. Gentleman, and S. DebRoy

Maintainer: H. Pages <hpages at fhcrc.org>

To install this package, start R and enter:

```
source("http://bioconductor.org/biocLite.R")
biocLite("Biostrings")
```

パッケージを個別にインストールする場合

To cite this package in a publication, start R and enter:

```
citation("Biostrings")
```

Documentation

PDF	R Script	A short presentation of the basic classes defined in Bi
PDF	R Script	Biostrings Quick Overview
PDF	R Script	Handling probe sequence information
PDF	R Script	Multiple Alignments
PDF	R Script	Pairwise Sequence Alignments
PDF		Reference Manual
Text		NEWS

使い方の解説記事はPDFのところをクリック

Details

biocViews	DataImport, DataRepresentation, Genetics, Infrastructure, SequenceMatching, Sequencing, Software

Biostrings Quick Overview

Hervé Pagès
Fred Hutchinson Cancer Research Center
Seattle, WA

April 3, 2013

参考

Please note that *most* but *not all* the functionalities provided by the Biostrings package are listed in this document.

Function	Description
length	Return the number of sequences in an object.
names	Return the names of the sequences in an object.
[	Extract sequences from an object.
head, tail	Extract the first or last sequences from an object.
rev	Reverse the order of the sequences in an object.
c	Put in a single object the sequences from 2 or more objects.
width, nchar	Return the sizes (i.e. number of letters) of all the sequences in an object.
==, !=	Element-wise comparison of the sequences in 2 objects.
match, %in%	Analog to match and %in% on character vectors.
duplicated, unique	Analog to duplicated and unique on character vectors.
sort, order	Analog to sort and order on character vectors, except that the ordering of DNA or Amino Acid sequences doesn't depend on the locale.
split, relist	Analog to split and relist on character vectors, except that the result is a *DNAStringSetList* or *AAStringSetList* object.

Table 1: Low-level manipulation of *DNAStringSet* or *AAStringSet* objects.

Function	Description
subseq, subseq<-	Extract or replace subsequences in a set of sequences.
reverse complement reverseComplement	Compute the reverse, complement, or reverse-complement, of a set of DNA sequences.
translate	Translate a set of DNA sequences into a set of Amino Acid sequences.
chartr	T
replaceLetterAt	R

Table 2: Basic transformations of sequences.

Function	Description
alphabetFrequency letterFrequency	Tabulate the letters (all the letters in the alphabet for alphabetFrequency, only the specified letters for letterFrequency) of a sequence or set of sequences.
letterFrequencyInSlidingView	Specialized version of letterFrequency that tallies the requested letter frequencies for a fixed-width view that is conceptually slid along the input sequence.
consensusMatrix	Computes the consensus matrix of a set of sequences.
dinucleotideFrequency trinucleotideFrequency oligonucleotideFrequency	Fast 2-mer, 3-mer, and k-mer counting for DNA or RNA.
nucleotideFrequencyAt	Tallies the short sequences formed by extracting the nucleotides found at a set of fixed positions from each sequence of a set of DNA or RNA sequences.

Table 3: Counting / tabulating.

Function	Description
matchPattern countPattern	Find/count all the occurrences of a given pattern (typically short) in a reference sequence (typically long). Support mismatches and indels.
vmatchPattern vcountPattern	Find/count all the occurrences of a given pattern (typically short) in a set of reference sequences. Support mismatches and indels.
matchPDict countPDict whichPDict	Find/count all the occurrences of a set of patterns in a reference sequence. (whichPDict only identifies which patterns in the set have at least one match.) Support a small number of mismatches.
vmatchPDict vcountPDict vwhichPDict	[Note: vmatchPDict not implemented yet.] Find/count all the occurrences of a set of patterns in a set of reference sequences. (whichPDict only identifies for each reference sequence which patterns in the set have at least one match.) Support a small number of mismatches.
pairwiseAlignment	Solve (Needleman-Wunsch) global alignment, (Smith-Waterman) ls-free) overlap alignment problems.
	rences of a Position Weight Matrix in a
	nking patterns from sequences.
	n a reference sequence i.e. matches speci- attern, and a maximum distance between
matchProbePair	Find all the amplicons that match a pair of probes in a reference sequence.

*Biostrings*中の関数を使いこなせると、他の自然言語処理系プログラミング言語（perlやruby）を改めて勉強しなくても必要な解析の大部分が可能です

Mar 7 2014

Contents（Rで…）

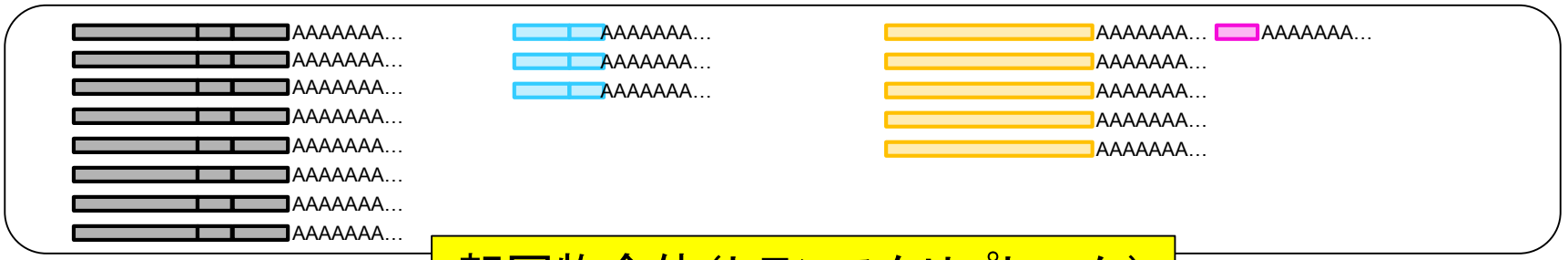
- ゲノム解析
  - アノテーションファイルを読み込んで目的のキーワードを含む行のみ抽出
  - multi-FASTAファイルを自在に解析
    - 配列長分布、GC含量、フィルタリング、部分配列の切り出しなど
    - 連続塩基の出現頻度（CpG）解析、ゲノム配列取得など
- トランスクリプトーム解析
  - **研究目的別留意点：サンプル内とサンプル間の違い**
  - マッピング → カウント情報取得
  - データを眺める：クラスタリングやM-A plot
  - 理想的な実験デザイン
  - なぜx倍発現変動という議論がだめなんですか？
  - モデルとか分布って、自分の解析結果にどういう影響を与えているの？
  - 多重比較問題：FDRって何？

トランスクリプトームとは

- ある状態のあるサンプルのあるゲノムの領域

遺伝子全体（ゲノム）

・どの染色体上のどの領域にどの遺伝子があるかは調べる個体が同じなら、目だろうが心臓だろうが不変



転写物全体（トランスクリプトーム）

・遺伝子1は沢山転写されている（発現している）
・遺伝子4はごくわずかしか転写されてない
・…

トランスクリプトームとは

- ある状態のあるサンプルのあるゲノムの領域

遺伝子全体（ゲノム）

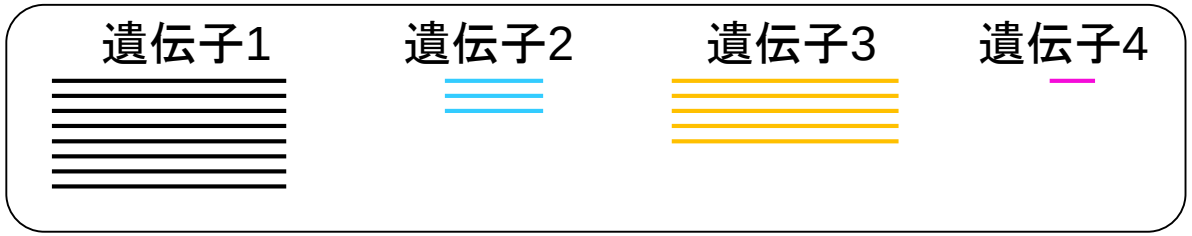
・どの染色体上のどの領域にどの遺伝子があるかは調べる個体が同じなら、目だろうが心臓だろうが不変

転写物全体（トランスクリプトーム）

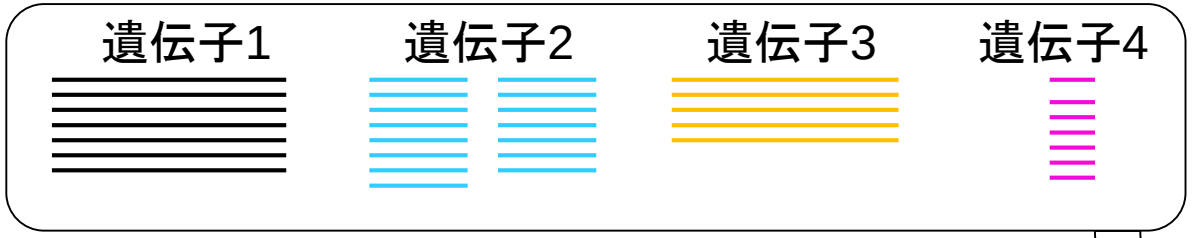
・遺伝子2は光刺激に応答して発現亢進
・遺伝子4も光刺激に応答して発現亢進

トランスクリプトーム情報を得る手段

- 光刺激前（T1）の目のトランスクリプトーム



- 光刺激後（T2）の目のトランスクリプトーム



これがいわゆる
遺伝子発現行列

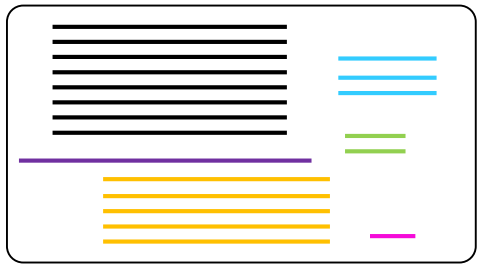
	T1	T2
遺伝子1	8	7
遺伝子2	3	15
遺伝子3	5	5
遺伝子4	1	7
…	…	…

・マイクロアレイ
・RNA-Seq
・…

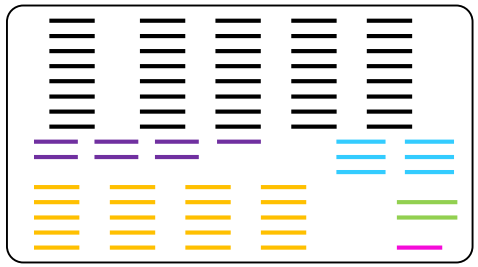
トランスクリプトーム取得

- 次世代シーケンサー：Illumina社の場合

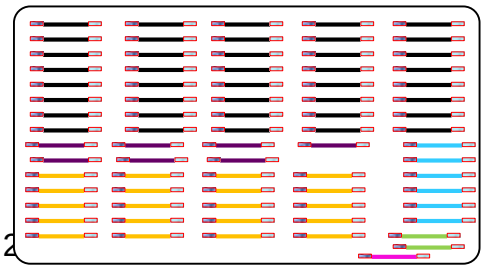
光刺激前（T1）の目のトランスクリプトーム



数百塩基程度
に断片化



二種類のアダプター
配列を両末端に付加

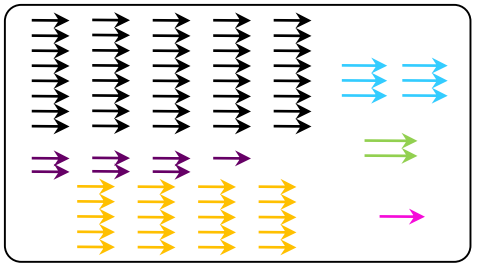


配列決定

・ペアードエンド法
断片配列の両末端が数百塩基以内の対の二種類の配列が得られる

・シングルエンド法

シングルエンド法
の場合



FASTA形式とFASTQ形式

- FASTA形式
  - 1行目 : “>”ではじまる一行のdescription行
  - 2行目 : 配列情報

```
>SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
```

- FASTQ形式
  - 1行目 : “@”ではじまる1行のdescription行
  - 2行目 : 配列情報
  - 3行目 : ”+”からはじまる1行（のdescription行）
  - 4行目 : クオリティ情報

```
@SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
+
!''*(((***+))%%%++)(%%%%).1***-+*''))**55CCF>>>>>>CCCCCCC65
```

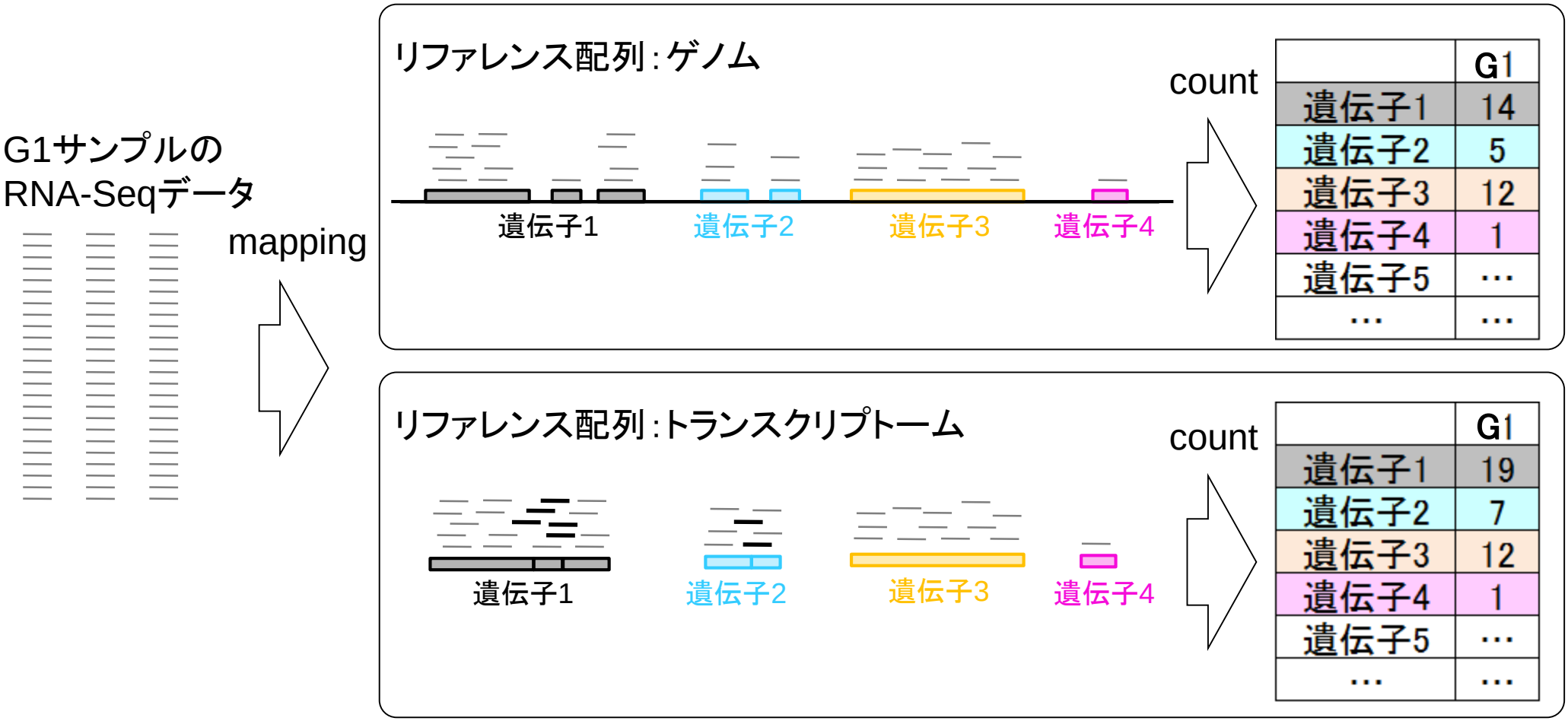
http://en.wikipedia.org/wiki/FASTQ_format

トランスクリプトーム解析の目的は様々

- トランスクリプトーム配列取得
  - ゲノム配列既知の場合、Cufflinksなどを用いて遺伝子構造推定（アノテーション）
  - ゲノム配列未知の場合、Trinityなどのトランスクリプトーム用アセンブラを実行
- 遺伝子またはisoformごとの発現量の正確な推定
  - RSEMなどを利用して発現量情報を得る
  - ある特定のサンプル内での遺伝子間の発現量の大小関係を知りたい
  - Length biasやGC biasなどの各種補正がポイント
- 比較するサンプル間で発現変動している遺伝子またはisoformの同定
  - *TCC*パッケージなどを利用して発現変動遺伝子（DEG）を得る
  - Sequence depthやサンプル間で発現している遺伝子のcomposition biasの補正がポイント
  - （GO解析など）DEG結果を用いる多くの下流解析結果に影響を及ぼす

マップされたリード数 = 発現量ではないが…

- 基本的なマッピングプログラム（bowtieなど）を用いた場合



マップされたリード数のカウント情報は、発現量推定の基本情報です

研究目的別留意点：遺伝子間比較

- **発現量補正の基本形**：カウント数×$\frac{\text{定数}}{\text{配列長}\times\text{総リード数}}$
  - RPK (Reads per kilobase)
  - RPM (Reads per million)
  - RPKM (Reads per kilobase per million)
- **同一サンプル内での異なる遺伝子間の発現レベル比較の場合**
  - 配列長由来bias：長いほど沢山sequenceされる
    - RPKMやFPKMなどの配列長を考慮して正規化されたデータで解析
  - GC含量由来bias：カウント数の分布がGC含量依存的である
    - Risso et al., *BMC Bioinformatics*, **12**: 480, 2011
    - Benjamini and Speed, *Nucleic Acids Res.*, **40**: e72, 2012

総リード数(ライブラリサイズ or sequence depth)補正は不必要
理由：遺伝子間の発現レベルの大小関係は定数倍しても不変

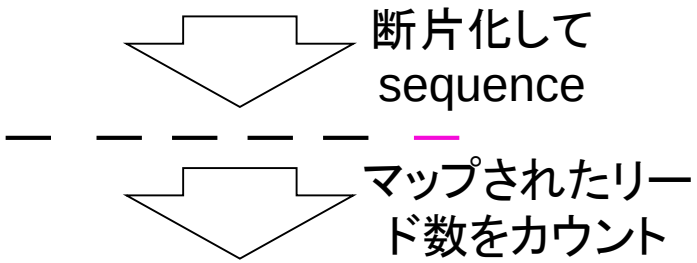
研究目的別留意点：サンプル間比較

- **発現量補正の基本形**：カウント数$\times \frac{\text{定数}}{\text{配列長}\times\text{総リード数}}$
  - RPK (Reads per kilobase)
  - RPM (Reads per million)
  - RPKM (Reads per kilobase per million)
- **異なるサンプル間での同一遺伝子間の発現レベル比較の場合**
  - **総リード数の違い**：総リード数が*x*倍違うと全体的に*x*倍変動…
    - RPM正規化で全体を揃えることは基本
  - **組成の違い**：サンプル特異的高発現遺伝子の存在で比較困難に…
    - TMM正規化法(Robinson and Oshlack, *Genome Biol.*, **11**: R25, 2010)
    - TbT正規化法(Kadota et al., *Algorithms Mol. Biol.*, **7**: 5, 2012)

Length biasやGC bias補正は少なくとも理論上は不必要
理由：同一遺伝子に対して掛かる係数はサンプル間で同じ

配列長の補正

- 配列長が長い遺伝子ほど沢山sequenceされる
  - それらの遺伝子上にマップされる生のリード数が増加傾向
  - 配列長が長い遺伝子ほど発現レベルが高い傾向になる



mRNA	リード数
AAAAAAA...	5
AAAAAAA...	1

1つのサンプル内で異なる遺伝子間の発現レベルの大小関係を配列長を考慮せずに比較することはできない

配列長を考慮した発現量推定のイメージ

- gene1: 3 exons (middle length), 14 reads mapped (**low** coverage)
- gene2: 3 exons (middle length), 56 reads mapped (**high** coverage)
- gene3: 2 exons (**short** length), 12 reads mapped (middle coverage)
- gene4: 2 exons (**long** length), 31 reads mapped (middle coverage)

マップされたリード分布　　生リードカウント結果　　補正度の発現量

「Garber et al., *Nat. Methods*, **8**: 469-477, 2011」のFig. 3a

・長さが同じならリード数の多い方が発現量高い（gene 1 vs. 2）
・長いほどマップされるリード数が多くなる効果を補正する必要がある（gene 3 vs. 4）

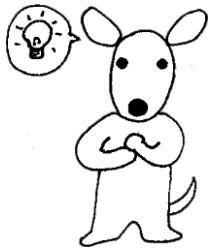
1つのサンプル内で転写物または遺伝子間の発現レベルの大小を比較したい場合には配列長を考慮すべきである

配列長の補正

mRNA	リード数	配列長(in bp)
 AAAAAAA…	5	1500
 AAAAAAA…	1	300

- 前提条件：配列長が既知
- 補正の基本戦略：配列長で割る
  - 「1 / 配列長」を掛ける場合
    → 「塩基あたりの平均のリード数」を計算しているのと等価
  - 「1000 / 配列長」を掛ける場合
    → 「その遺伝子の配列長が1000bpだったときのリード数(or カウント数)」と等価

Reads Per Kilobase (RPK)
Counts Per Kilobase (CPK)

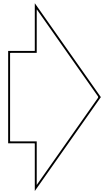


マイクロアレイデータの正規化

- 各サンプルから測定されたシグナル強度の和は一定
  - アレイ上の遺伝子数が少ない場合は非現実的だが、数千～数万種類の遺伝子が搭載されているので妥当という思想

	sample1	sample2
gene1	10.5	12.4
gene2	6.4	7.1
gene3	8.0	8.5
gene4	10.8	11.4
gene5	5.6	6.7
gene6	8.4	8.9
gene7	6.2	7.0
gene8	6.1	6.8
gene9	6.6	6.5
gene10	5.1	5.8
総和	73.7	81.1

グローバル正規化

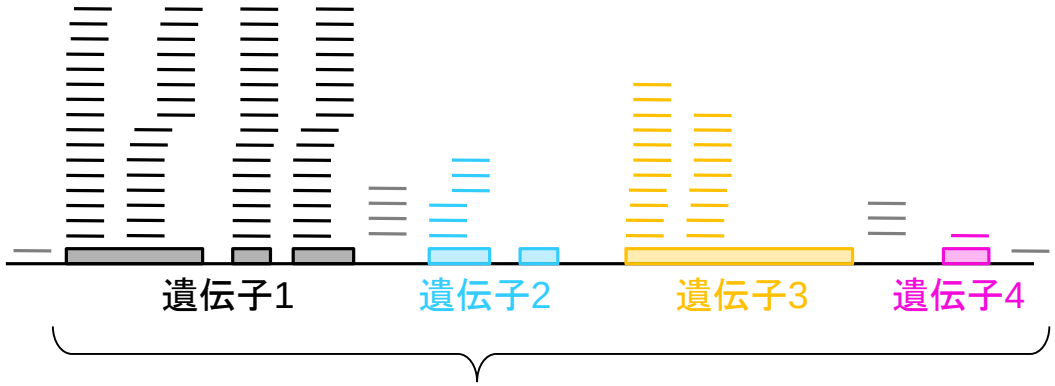


	sample1	sample2
gene1	14.2	15.3
gene2	8.7	8.8
gene3	10.9	10.5
gene4	14.7	14.1
gene5	7.6	8.3
gene6	11.4	11.0
gene7	8.4	8.6
gene8	8.3	8.4
gene9	9.0	8.0
gene10	6.9	7.2
総和	100.0	100.0

背景：サンプルごとにシグナル強度の総和は異なる
対策：総和が任意の値（例では100）になるような正規化係数を掛ける
例：sample1の正規化係数= 100 / **73.7**

RNA-Seqデータの正規化の一部

- 発現しているRNA量の総和はサンプル間で一定



	T1	T2
遺伝子1	40	7
遺伝子2	6	15
遺伝子3	20	5
遺伝子4	1	1
総リード数	67	28

RPM正規化

	T1	T2
遺伝子1	597014.9	250000.0
遺伝子2	89552.2	535714.3
遺伝子3	298507.5	178571.4
遺伝子4	14925.4	35714.3
総リード数	1000000	1000000

Reads Per Million mapped reads（RPM）
正規化後の総リード数が100万（one million）になるように補正
例：T1の正規化係数 = 1000000 / **67**

RPKM

- Reads per kilobase (of exon) per million (mapped reads)
- 配列長が1000 bpだったとき、かつ総リード数が100万だったときのカウント数

$$\mathrm{RPKM} = \text{カウント数} \times \frac{1{,}000}{\text{配列長}} \times \frac{1{,}000{,}000}{\text{総リード数}} = \text{カウント数} \times \frac{1{,}000{,}000{,}000}{\text{配列長} \times \text{総リード数}}$$

sample_length_count.txt

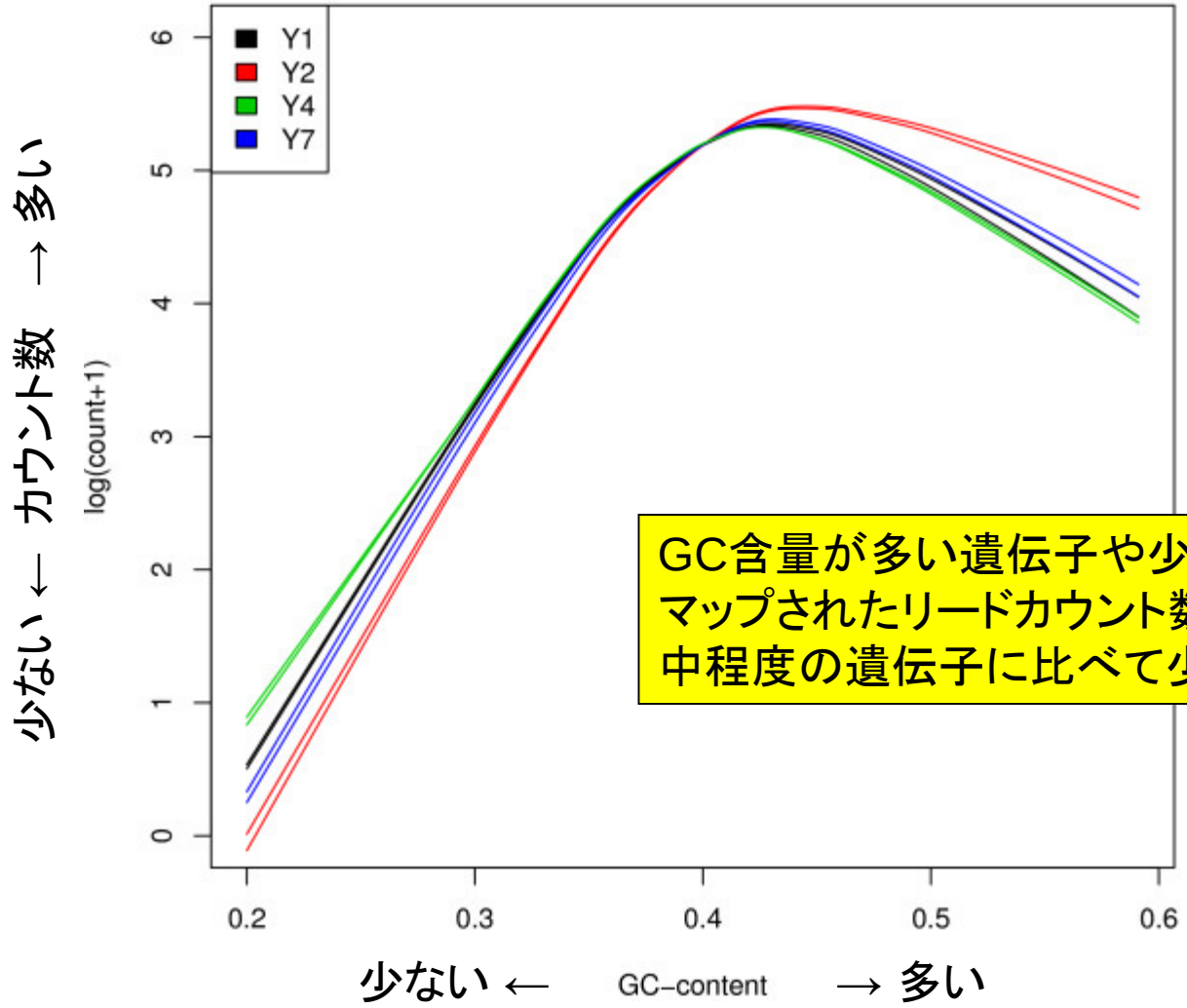
ID	Length	Count
NM_203348.1	3543	3
NM_001008737.1	1897	19
NM_001037228.1	537	7
NM_033183.2	886	0
NM_138368.3	4443	56
NM_152833.2	2844	85
NM_001100111.1	682	0
NM_001102659.1	1376	0
NM_001104548.1	888	3

総リード数 = 2385273

hoge1.txt

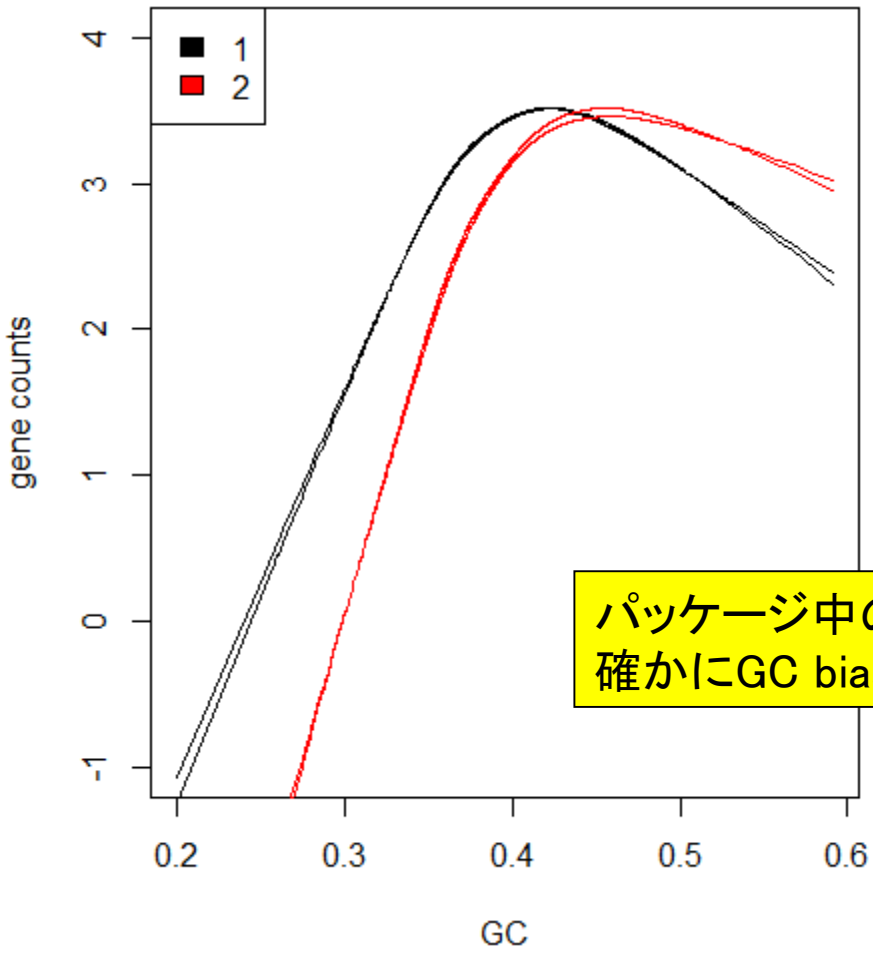
rownames(data)	Length	Count
NM_203348.1	3543	0.355
NM_001008737.1	1897	4.199
NM_001037228.1	537	5.465
NM_033183.2	886	0
NM_138368.3	4443	5.284
NM_152833.2	2844	12.53
NM_001100111.1	682	0
NM_001102659.1	1376	0
NM_001104548.1	888	1.416

GC bias補正の必要性

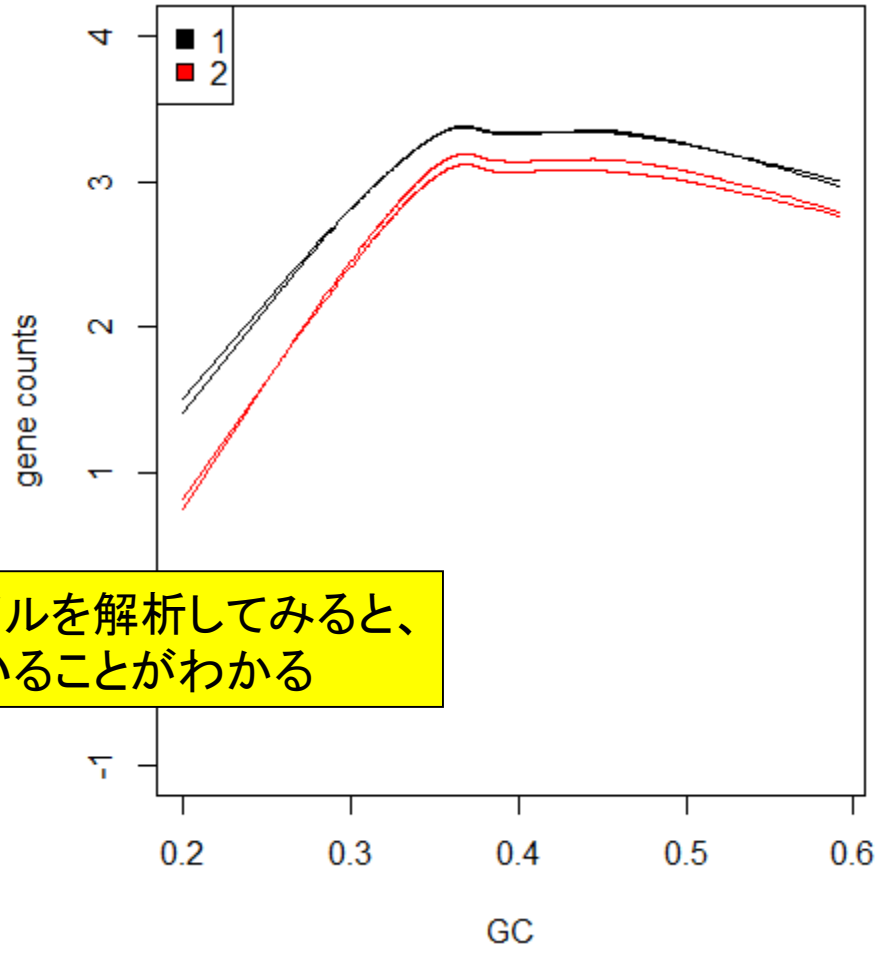


GC含量が多い遺伝子や少ない遺伝子上にマップされたリードカウント数は、GC含量が中程度の遺伝子に比べて少ない傾向にある

GC bias補正の必要性



Quantile
正規化



パッケージ中のサンプルファイルを解析してみると、
確かにGC biasが緩和されていることがわかる

Contents（Rで…）

- ゲノム解析
  - アノテーションファイルを読み込んで目的のキーワードを含む行のみ抽出
  - multi-FASTAファイルを自在に解析
    - 配列長分布、GC含量、フィルタリング、部分配列の切り出しなど
    - 連続塩基の出現頻度（CpG）解析、ゲノム配列取得など
- トランスクリプトーム解析
  - 研究目的別留意点：サンプル内とサンプル間の違い
  - **マッピング → カウント情報取得**
  - データを眺める：クラスタリングやM-A plot
  - 理想的な実験デザイン
  - なぜx倍発現変動という議論がだめなんですか？
  - モデルとか分布って、自分の解析結果にどういう影響を与えているの？
  - 多重比較問題：FDRって何？

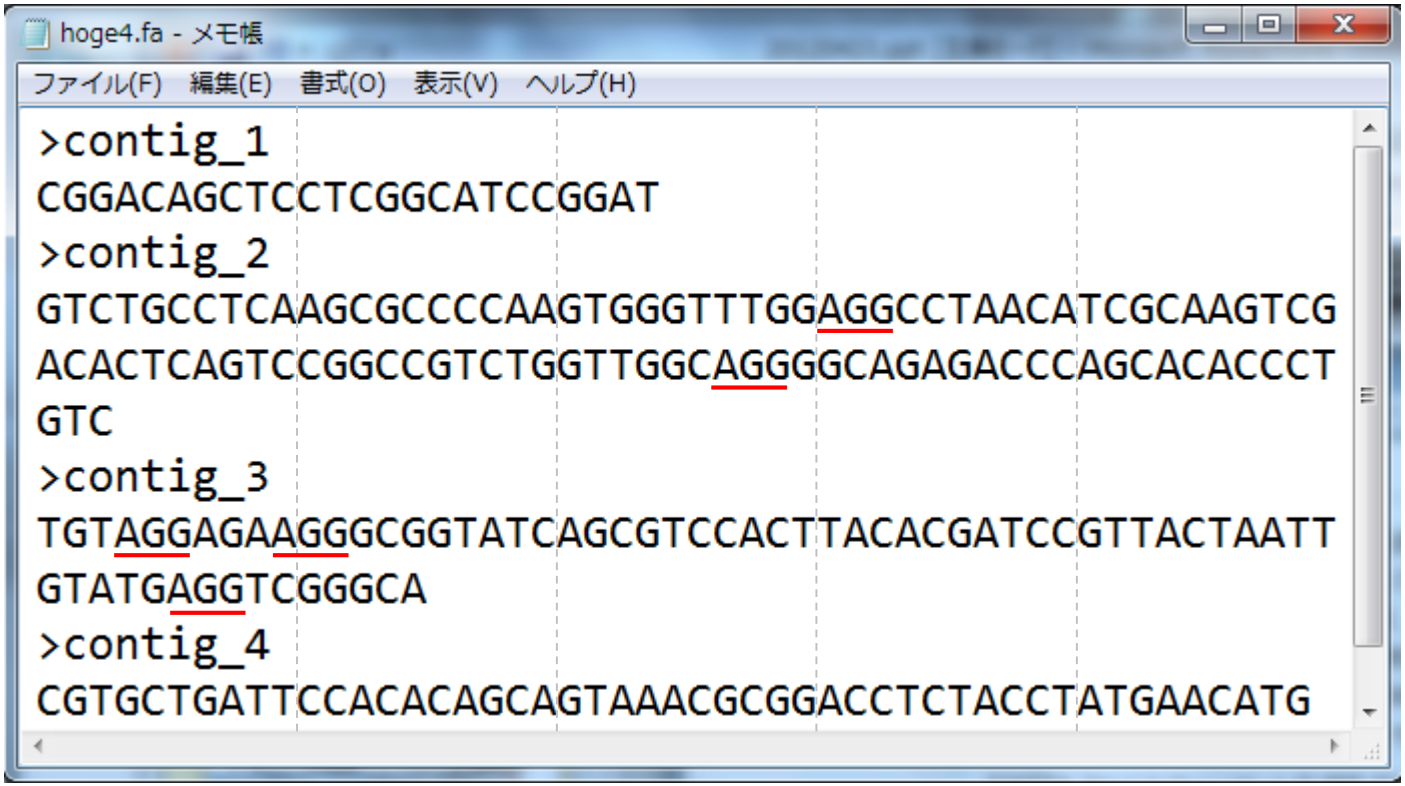
今はLinuxコマンド抜きで一通り解析可能

- *SRAdb* (Zhuら, *BMC Bioinformatics*, **14**: 19, 2013)
  - 公共DBからのRNA-seqデータ（FASTQファイル）取得
- ***QuasR* (Lerchら, unpublished)**
  - **リファレンス配列（ゲノム or トランスクリプトーム）へのマッピング**
    - **Bowtie (Langmeadら, 2009) or SpliceMap (Auら, 2010)を選択可能**
    - **出力はBAM形式ファイル、QCレポートも**
  - **遺伝子アノテーション情報をもとにカウントデータ取得**
    - ***GenomicFeatures* (Lawrenceら, 2013)で得られるTranscriptDbオブジェクトを利用**
    - **UCSC known genesやEnsembl genesのカウントデータなど**
- *TCC* (Sunら, *BMC Bioinformatics*, **14**: 219, 2013)
  - 内部的に*edgeR* (Robinsonら, 2010)や*DESeq* (Anders, 2010)などを用いて頑健な発現変動解析を実行

アセンブル以外ならWindows（のR）上でどうにかなる時代がやってきました

マッピング ＝ 大量高速文字列検索

- マップされる側のリファレンス配列：hoge4.fa
- マップする側のRNA-seqデータ（リードと呼ばれる）：”AGG”



出力ファイル

	start	end
contig_2	31	33
contig_2	77	79
contig_3	4	6
contig_3	10	12
contig_3	56	58

マッピングプログラムの出力：（どのリードが）リファレンス配列上のどの位置から転写されたものかという座標情報

*QuasR*パッケージを用いてマッピング

- Basic alignerの1つであるbowtie (Langmead et al., 2009)を利用
  - マッピング時に多くのオプションを指定可能
  - “-v”：許容するミスマッチ数を指定するオプション。”-v 0”は、リードがリファレンスに完全一致するもののみレポート。”-v 2”は、2塩基ミスマッチまで許容してマップされうる場所を探索。
  - “-m”：出力するリード条件を指定するオプション。”-m 1”は、複数個所にマップされるリードを除外して、1か所にのみマップされたリードをレポート。”-m 3”は、合計3か所にマップされるリードまでをレポート。
  - “--best --strata”：最も少ないミスマッチ数でマップされるもののみ出力する、という意思表示。これをつけずに”-v 2 -m 1”などと指定すると、たとえ完全一致(ミスマッチ数0)で1か所にのみマップされるリードがあったとしても、どこか別の場所で1塩基ミスマッチでマップされる個所があれば、マップされうる場所が2か所ということを意味し、そのリードは出力されなくなる。それを防ぐのが主な目的
  - …

デフォルトである程度よきに計らってくれるが…実際の挙動を完全に把握できる状況で様々なオプションを試したい

マッピング時に用いるオプションの理解

- マップされる側のリファレンス配列：ref_genome.fa

ref_genome.fa - メモ帳

ファイル(F)　編集(E)　書式(O)　表示(V)　ヘルプ(H)

```
>chr1
CGAGGAGGAACGCTTACGAGATCAGGCTAAGAGTGGATGCTGAGTGGG
>chr2
AGGGAGGGGGTCCAGTATCTATGGCCTAAAAACATAGACACCTTGAGGAG
ACGCAGGTAGGCTGAGGATAAAGCCGTTTGCACGCATCATGAAGGGGCTG
CTCGGGTATGGTTAGTCTTTGCCTCTAGATTTTCACGACGCTGCGGTTCA
TGACGCCCTG
>chr3
GGGGGGACTATTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGGC
AGCATCTAGTCGCATCAGAAGGGTGTAGTCAGCCTATAGTTAACTAGTTT
>chr4
CGAGACGAGCAAGTTATTCGCTCAGTGAATGGGTAGCAAAAGAATGTTGT
CGTCTGTATTGGGGCCTATGCTCGACAAGAGATTGTGTGTAGTATGAGCC
ACCAGACTTTACCGTACAAGATA
>chr5
GCGGGGTCTATTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGGC
AGCATCTAGTCGCATCAGAAGGGTGTAGTCAGCCTATAGTTAACTAGTTT
```

chr3とchr5の違いは、2番目と7番目の塩基のみ。主に”-m”オプションの違いの把握が可能。

マッピング時に用いるオプションの理解

- マップされる側のリファレンス配列：ref_genome.fa

サンプルデータ NEW

1. Marioni et al., Genome Res., 2008のSupplementary table 2のデータ。

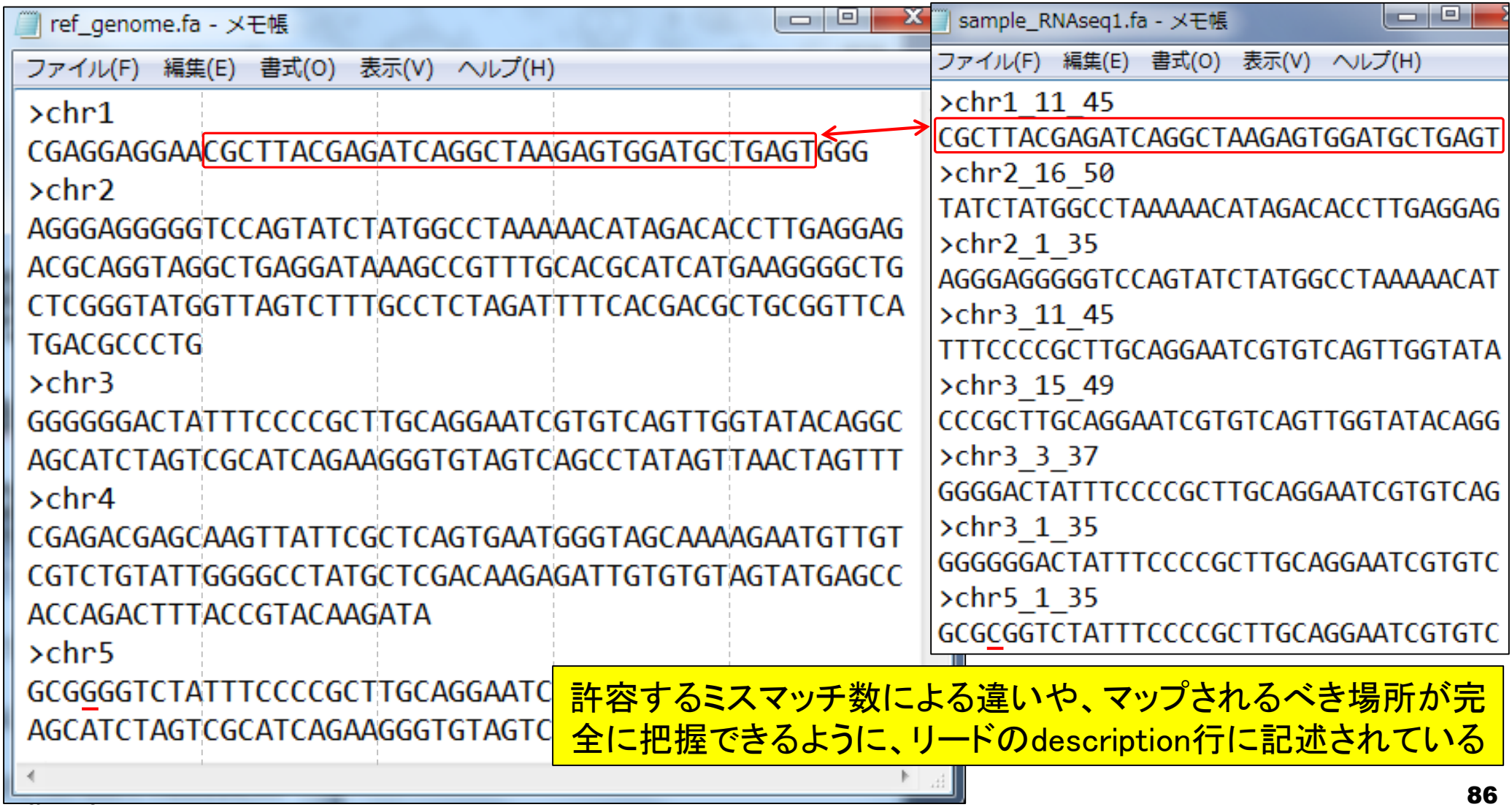
18. ランダムな塩基配列から生成したリファレンスゲノム配列データ(ref_genome.fa)。
48, 160, 100, 123, , 100の配列長をもつ、計5つの塩基配列を生成しています。descriptionは行"contig"という記述を基本としています。
塩基の存在比はAが28%, Cが22%, Gが26%, Tが24%にしています。
set.seed関数を利用し、chr3の配列と同じものをchr5としてコピーして作成したのち、2番目と7番目の塩基置換を行っています。そのため、実際に指定するのは最初の4つ分の配列長のみです。

```
out_f <- "ref_genome.fa"                #出力ファイル名を指定してout_fに格納
param1 <- c(48, 160, 100, 123)          #配列長を指定
narabi <- c("A","C","G","T")            #以下の数値指定時にACGTの並びを間違えないようにするため
param2 <- c(28, 22, 26, 24)             #(A,C,G,Tの並びで)各塩基の存在比率を指定
param3 <- "chr"                         #FASTA形式ファイルのdescription行に記述する内容
param4 <- 3                             #コピーを作成したい配列番号を指定

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

#塩基置換関数の作成
enkichikan <- function(fa, p) {         #関数名や引数の作成
  t <- substring(fa, p, p)              #置換したい位置の塩基を取りだす
  t_c <- chartr("CGAT", "GCTA", t)      #置換後の塩基を作成
  substring(fa, p, p) <- t_c            #置換
  return(fa)                            #置換後のデータを返す
}
```

コピペで作成しています

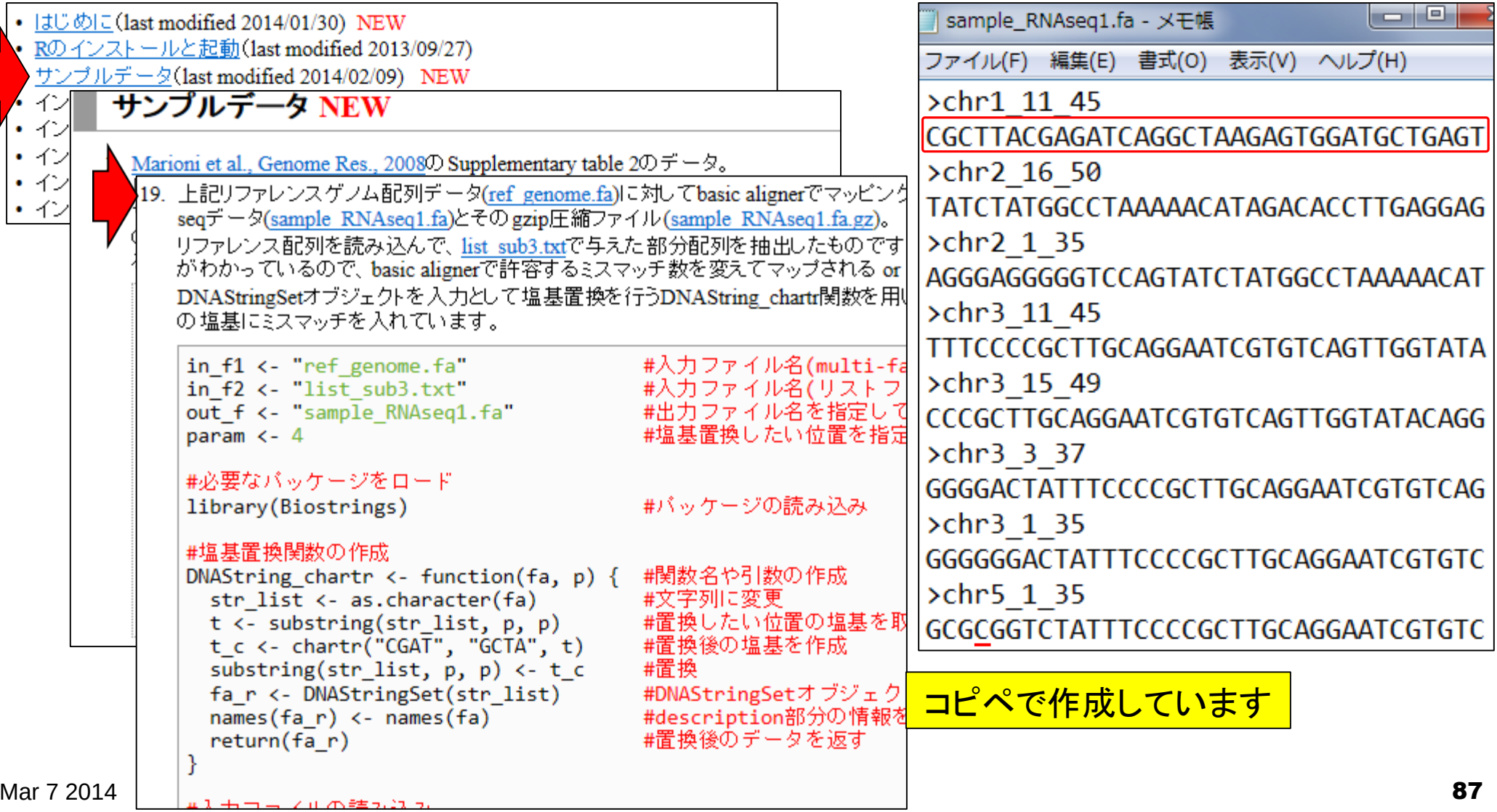
マッピング時に用いるオプションの理解

- マップする側のRNA-seqデータ：sample_RNAseq1.fa



マッピング時に用いるオプションの理解

- マップする側のRNA-seqデータ：sample_RNAseq1.fa



- マッピング | 備忘録 | について(last modified 2013/10/25)
- マッピング | 備忘録 | basic aligner(last modified 2013/10/17)
- マッピング | 備忘録 | splice-aware aligner(last modified 2014/02/04) NEW
- マッピング | 備忘録 | Bisulfite sequencing用(last modified 2014/02/04) NEW
- マッピング | 備忘録 | (ESTレベルの長さの)contig(last modified 2010/12/06)
- マッピング | 基礎(last modified 2013/06/19)
- マッピング | single-end | ゲノム | basic aligner(基礎) | QuasR(Lerch_XXX)(last modified 2013/10/25)
- マッピング | single-end | ゲノム | basic aligner(応用) | QuasR(Lerch_XXX)(last modified 2013/10/25)
- マッピング | single-end | ゲノム | splice-aware aligner | QuasR(Lerch_XXX)(last modified 2013/10/25)
- マップ後 | について(last modified 2013/06/19)
- マップ後 | 出力ファ
- マップ後 | 出力ファ

- マッピング | single-end | ゲノム | basic aligner(応用) | QuasR(Lerch_XXX)

マッピング | single-end | ゲノム | basic aligner(応用) | QuasR(Lerch_XXX)

QuasRパッケージを用いてsingle-end RNA-seqデータのリファレンスゲノム配列へのマッピングを行うやり方を示します。basic alignerの一つであるBowtie (Langmead et al., Genome Biol., 2009)を実装した Rbowtieパッケージを内部的に使っています。
Bowtie自体は、複数個所にマップされるリードの取り扱い(uniquely mapped reads or multi-mapped reads)を"-m"オプションで指定したり、許容するミスマッチ数を指定する"-v"などの様々なオプションを利用可能ですが、「基礎」のところではやり方を示しませんでした。ここでは、マッピングのオプションをいくつか変更して挙動を確認したり、複数のRNA-seqファイルを一度にマッピングするやり方を示します。
尚、出力ファイルは、"*.bam", "*_QC.pdf", "*.bed"の3つです。それ以外のファイルは基本無視で大丈夫です。
「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18,19のRNA-seqデータ(sample_RNAseq1.fa)のref_genome.faへの(mapping_single_genome1.txt):

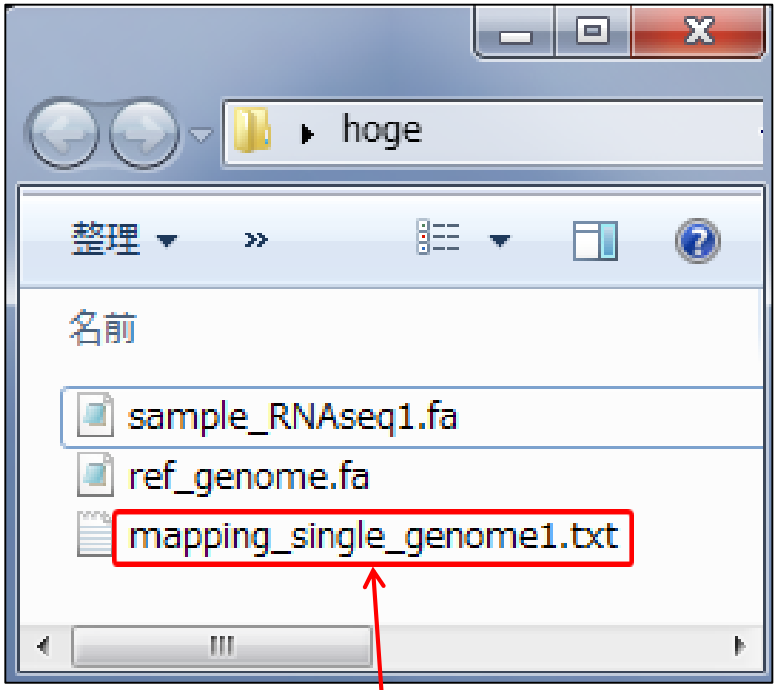
オプションを"-m 1 --best --strata -v 0"とした例です。
sample_RNAseq1.faでマップされないのは計3リードです。2リード("chr3_11_45"と"chr3_15_49")はchr5にもマップされるので、"-m 1"オプションで落とされます。1リード("chr5_1_35")は該当箇所と完全一致ではない(4番目の塩基にミスマッチをいれている)ので落とされます。

```
in_f1 <- "mapping_single_genome1.txt"  #入力ファイル名を指定してin_f1に格納(RNA-seqファイル)
in_f2 <- "ref_genome.fa"               #入力ファイル名を指定してin_f2に格納(リファレンス配列)
param_mapping <- "-m 1 --best --strata -v 0"#マッピング時のオプションを指定

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

#本番(マッピング)
time_s <- proc.time()                  #計算時間を計測するため
out <- qAlign(in_f1, in_f2, alignmentParameter=param_mapping)#マッピングを行うqAlign関数を実行
time_e <- proc.time()                  #計算時間を計測するため
time_e - time_s                        #計算時間を表示(一番右側の数字。単位はsecond)
out                                    #マッピングに用いたパラメータや入力ファイルの情報などを表
```

複数のRNA-seqサンプルを実行できるようにリストファイルとして与える

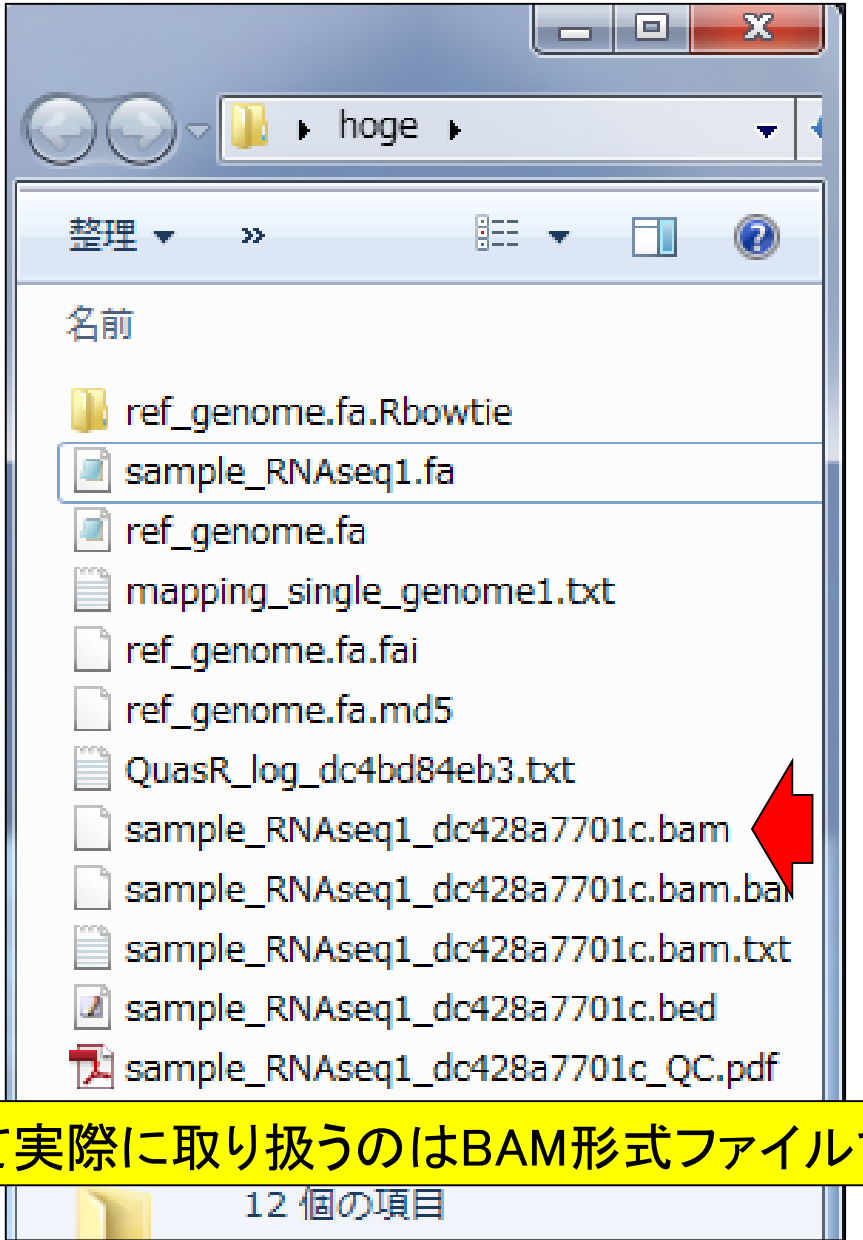
許容するミスマッチ数は0個(”-v 0”)、1か所にマップされるリードのみ出力(”-m 1”)

QuasRパッケージを用いてマッピング



	A	B
1	FileName	SampleName
2	sample_RNAseq1.fa	namae

実行後



出力ファイルとして実際に取り扱うのはBAM形式ファイルです

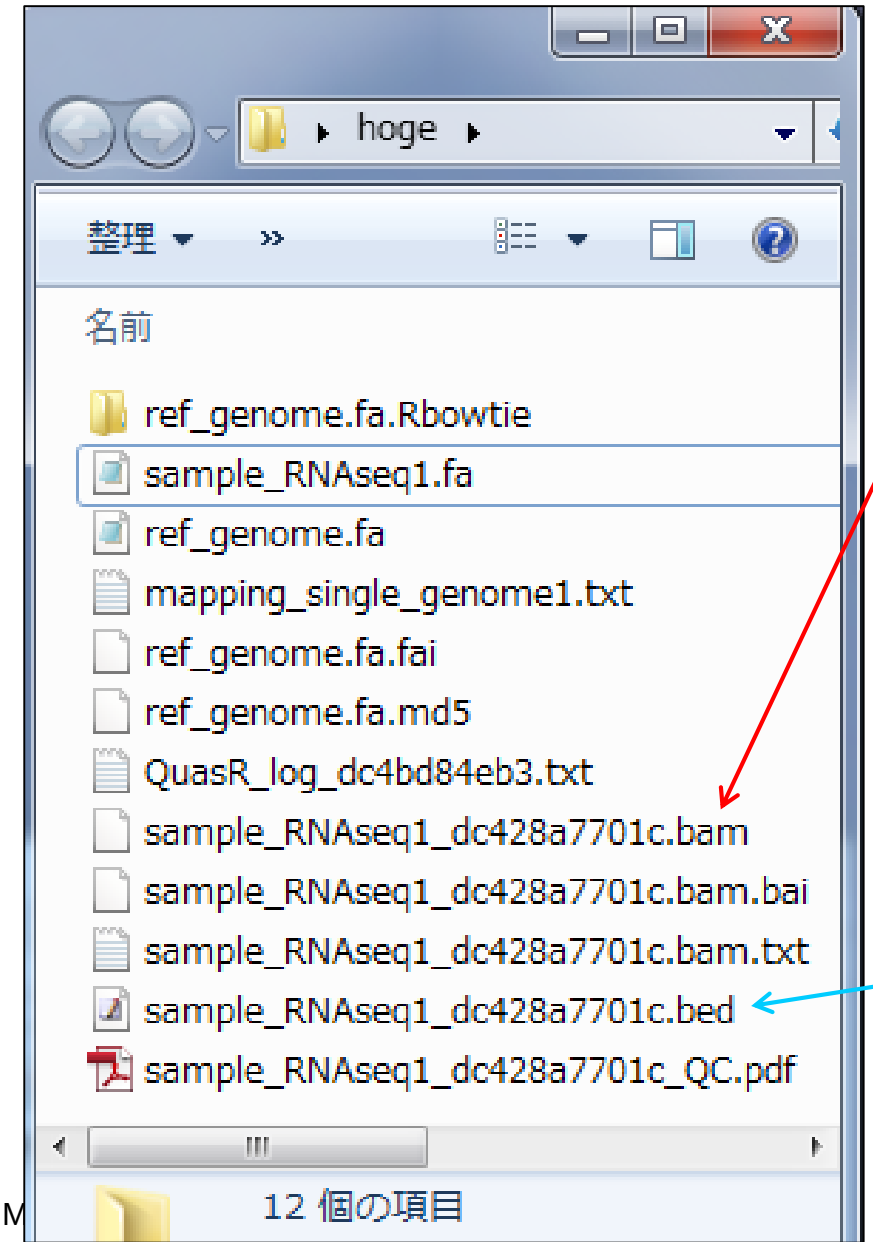
マッピング結果の出力ファイル形式

- ゲノム上のどの位置にどのリードがマッピングされたか（トランスクリプトームの場合どの転写物配列上のどの位置にどのリードがマッピングされたか）を表すファイル形式は複数あります。
  - SAM (Sequence Alignment/Map) format
    - SAMtools (Li et al., *Bioinformatics*, **25**: 2078-2079, 2009)
  - **BAM** (Binary Alignment/Map) format
    - SAMtools (Li et al., *Bioinformatics*, **25**: 2078-2079, 2009)
  - **BED** (Browser Extensible Data) format
    - BEDtools (Quinlan et al., *Bioinformatics*, **26**: 841-842, 2010)

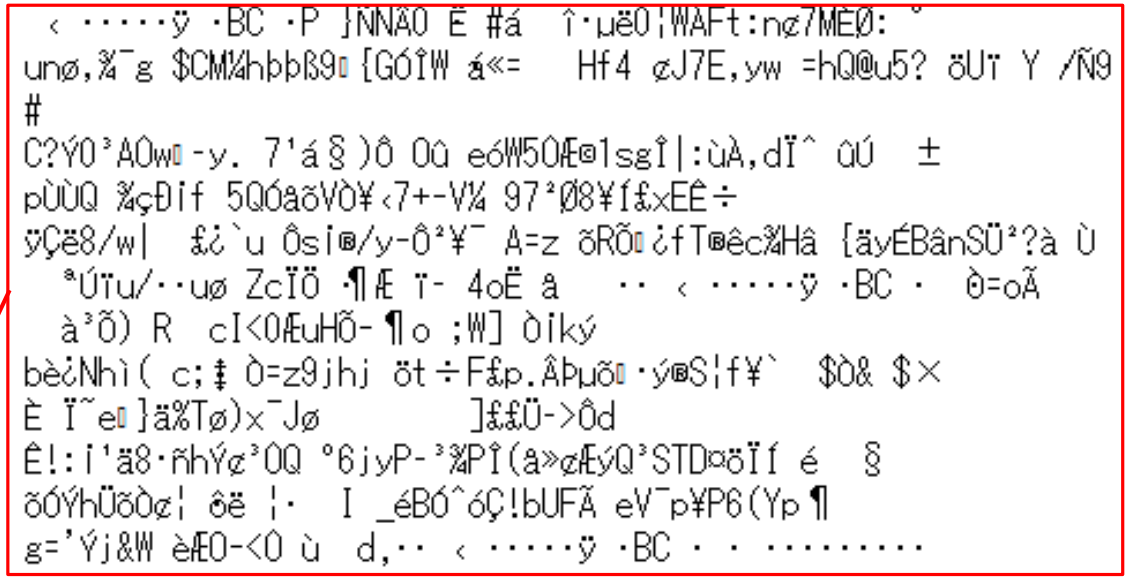
…

実用上はBAM形式、視覚上はBED形式

マッピング結果の出力ファイル形式



BAM形式ファイル



BED形式ファイル

chr1	11	45
chr2	1	35
chr2	16	50
chr3	1	35
chr3	3	37

BEDの最小限の情報は、リードIDを含まない

マッピングオプションと結果の解釈

- “-m 1 --best --strata -v 0”：0ミスマッチで1か所にのみマップされるリードを出力

ref_genome.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
>chr1
CGAGGAGGAACGCTTACGA
>chr2
AGGGAGGGGGTCCAGTATC
ACGCAGGTAGGCTGAGGAT
CTCGGGTATGGTTAGTCTT
TGACGCCCTG
>chr3
GGGGGGACTATTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGGC
AGCATCTAGTCGCATCAGAAGGGTGTAGTCAGCCTATAGTTAACTAGTTT
>chr4
CGAGACGAGCAAGTTATTCGCTCAGTGAATGGGTAGCAAAAGAATGTTGT
CGTCTGTATTGGGGCCTATGCTCGACAAGAGATTGTGTGTAGTATGAGCC
ACCAGACTTTACCGTACAAGATA
>chr5
GCGGGGTCTATTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGGC
AGCATCTAGTCGCATCAGAAGGGTGTAGTCAGCCTATAGTTAACTAGTTT
```

TGGG
GAGGAG
GGGCTG
GGTTCA

chr1	11	45	
chr2	1	35	
chr2	16	50	
chr3	1	35	
chr3	3	37	

sample_RNAseq1.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
>chr1_11_45
CGCTTACGAGATCAGGCTAAGAGTGGATGCTGAGT
>chr2_16_50
TATCTATGGCCTAAAAACATAGACACCTTGAGGAG
>chr2_1_35
AGGGAGGGGGTCCAGTATCTATGGCCTAAAAACAT
>chr3_11_45
TTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATA
>chr3_15_49
CCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGG
>chr3_3_37
GGGGACTATTTCCCCGCTTGCAGGAATCGTGTCAG
>chr3_1_35
GGGGGGACTATTTCCCCGCTTGCAGGAATCGTGTC
>chr5_1_35
GCGCGGTCTATTTCCCCGCTTGCAGGAATCGTGTC
```

マップされなかったのは、
計8リード中3リード

マッピングオプションと結果の解釈

- “-m 1 --best --strata -v 0”：0ミスマッチで1か所にのみマップされるリードを出力

ref_genome.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
>chr1
CGAGGAGGAACGCTTACGAGATCAGGCTAAGAGTGGATGCTGAGTGGG
>chr2
AGGGAGGGGGTCCAGTATCTATGGCCTAAAAACATAGACACCTTGAGGAG
ACGCAGGTAGGCTGAGGATAAAGCCGTTTGCACGCATCATGAAGGGGCTG
CTCGGGTATGGTTAGTCTTTGCCTCTAGATTTTCACGACGCTGCGGTTCA
TGACGCCCTG
>chr3
GGGGGGACTATTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGGC
AGCATCTAGTCGCATCAGAAGGGTGTAGTCAGCCTATAGTTAACTAGTTT
>chr4
CGAGACGAGCAAGTTATTCGCTCAGTGAATGGGTAGCAAAAGAATGTTGT
CGTCTGTATTGGGGCCTATGCTCGACAAGAGATTGTGTGTAGTATGAGCC
ACCAGACTTTACCGTACAAGATA
>chr5
GCGGGGTCTATTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGGC
AGCATCTAGTCGCATCAGAAGGGTGTAGTCAGCCTATAGTTAACTAGTTT
```

sample_RNAseq1.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
>chr1_11_45
CGCTTACGAGATCAGGCTAAGAGTGGATGCTGAGT
>chr2_16_50
TATCTATGGCCTAAAAACATAGACACCTTGAGGAG
>chr2_1_35
AGGGAGGGGGTCCAGTATCTATGGCCTAAAAACAT
>chr3_11_45
TTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATA
>chr3_15_49
CCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGG
>chr3_3_37
GGGGACTATTTCCCCGCTTGCAGGAATCGTGTCAG
>chr3_1_35
GGGGGGACTATTTCCCCGCTTGCAGGAATCGTGTC
>chr5_1_35
GCGCGGTCTATTTCCCCGCTTGCAGGAATCGTGTC
```

完全一致でも複数個所にマップされるために落とされたリード

マッピングオプションと結果の解釈

- “-m 1 --best --strata -v 0”：0ミスマッチで1か所にのみマップされるリードを出力

ref_genome.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
>chr1
CGAGGAGGAACGCTTACGAGATCAGGCTAAGAGTGGATGCTGAGTGGG
>chr2
AGGGAGGGGGTCCAGTATCTATGGCCTAAAAACATAGACACCTTGAGGAG
ACGCAGGTAGGCTGAGGATAAAGCCGTTTGCACGCATCATGAAGGGGCTG
CTCGGGTATGGTTAGTCTTTGCCTCTAGATTTTCACGACGCTGCGGTTCA
TGACGCCCTG
>chr3
GGGGGGACTATTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGGC
AGCATCTAGTCGCATCAGAAGGGTGTAGTCAGCCTATAGTTAACTAGTTT
>chr4
CGAGACGAGCAAGTTATTCGCTCAGTGAATGGGTAGCAAAAGAATGTTGT
CGTCTGTATTGGGGCCTATGCTCGACAAGAGATTGTGTGTAGTATGAGCC
ACCAGACTTTACCGTACAAGATA
>chr5
GCGGGGTCTATTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGGC
AGCATCTAGTCGCATCAGAAGGGTGTAGTCAGCCTATAGTTAACTAGTTT
```

sample_RNAseq1.fa - メモ帳

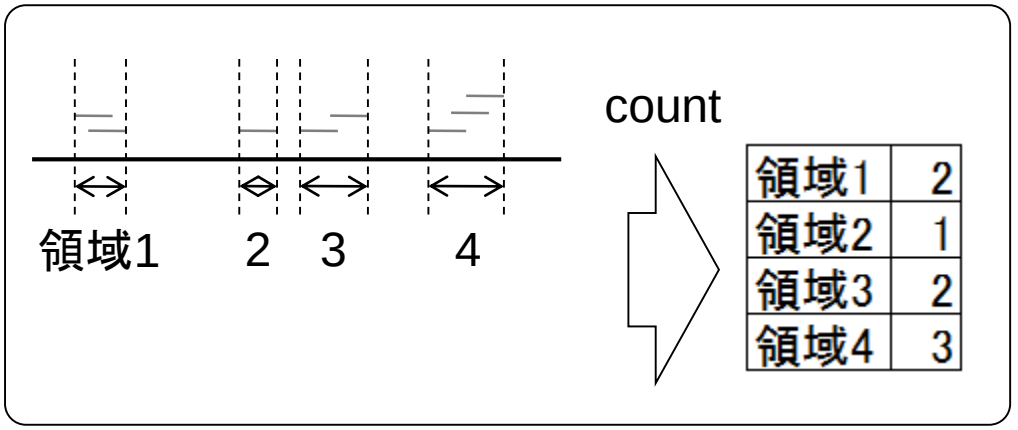
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
>chr1_11_45
CGCTTACGAGATCAGGCTAAGAGTGGATGCTGAGT
>chr2_16_50
TATCTATGGCCTAAAAACATAGACACCTTGAGGAG
>chr2_1_35
AGGGAGGGGGTCCAGTATCTATGGCCTAAAAACAT
>chr3_11_45
TTTCCCCGCTTGCAGGAATCGTGTCAGTTGGTATA
>chr3_15_49
CCCGCTTGCAGGAATCGTGTCAGTTGGTATACAGG
>chr3_3_37
GGGGACTATTTCCCCGCTTGCAGGAATCGTGTCAG
>chr3_1_35
GGGGGGACTATTTCCCCGCTTGCAGGAATCGTGTC
>chr5_1_35
GCGCGGTCTATTTCCCCGCTTGCAGGAATCGTGTC
```

1か所にのみマップされるがミスマッチのため落とされたリード

マッピング結果からのカウント情報取得

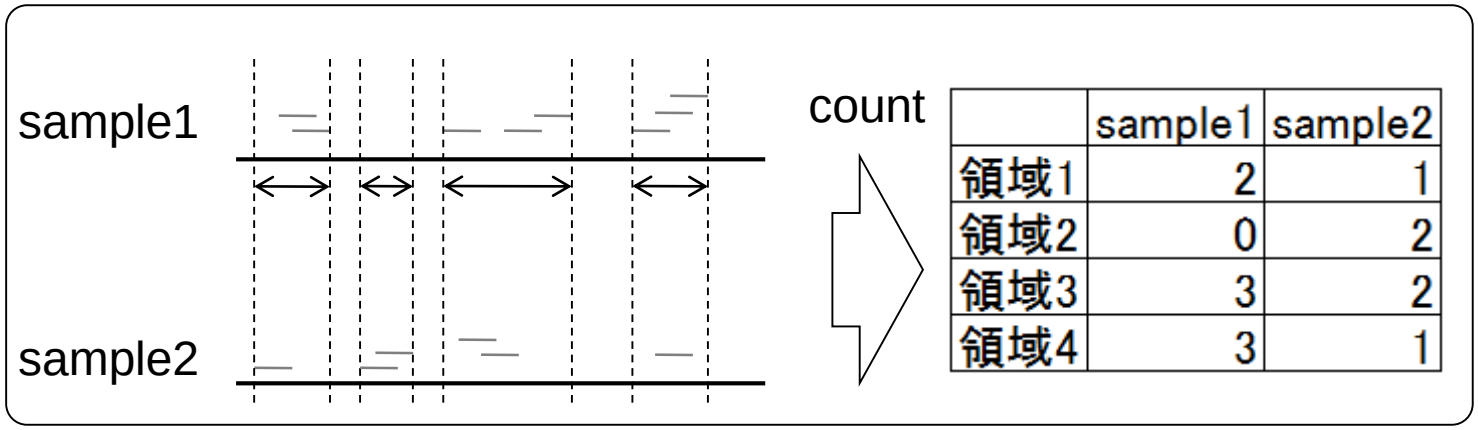
- アノテーション情報を利用する場合
  - UCSC Genes, Ensembl Genesなど様々なテーブル名を指定可能
  - gene, exon, promoter, junctionなど様々なレベルを指定可能
- アノテーション情報がない場合
  - マップされたリードの和集合領域を同定したのち、領域ごとのリード数をカウント
  - BEDtools (Quinlan et al., 2010)中のmergeBedプログラムを実行して和集合領域同定後、intersectBedプログラムを実行してリード数をカウントする作業に相当



基本的なイメージ

マッピング結果からのカウント情報取得

- アノテーション情報を利用する場合
  - UCSC Genes, Ensembl Genesなど様々なテーブル名を指定可能
  - gene, exon, promoter, junctionなど様々なレベルを指定可能
- アノテーション情報がない場合
  - マップされたリードの和集合領域を同定したのち、領域ごとのリード数をカウント
  - BEDtools (Quinlan et al., 2010)中のmergeBedプログラムを実行して和集合領域同定後、intersectBedプログラムを実行してリード数をカウントする作業に相当



	sample1	sample2
領域1	2	1
領域2	0	2
領域3	3	2
領域4	3	1

複数サンプルの場合には領域が変わりうる

- マップ後 | 出力ファイルの読み込み | BAM形式 (last modified 2013/09/30)
- マップ後 | 出力ファイルの読み込み | Bowtie形式 (last modified 2013/06/18)
- マップ後 | 出力ファイルの読み込み | SOAP形式 (last modified 2013/06/19)
- マップ後 | 出力ファイルの読み込み | htSeqTools (Planet 2012) (last modified 2013/06/19)
- マップ後 | カウント情報取得 | ゲノム | アノテーション有 | QuasR(Lerch XXX) (last modified 2013/11/06)
- マップ後 | カウント情報取得 | ゲノム | アノテーション無 | QuasR(Lerch XXX) (last modified 2013/11/06)
- マップ後 | カウント情報取得 | トランスクリプトーム | BEDファイルから (last modified 2013/10/13)
- マップ後 | 配列長とカウント数の関係 (last modified 2013/10/27)
- 正規化 | につ
- 正規化 | 基礎
- 正規化 | 基礎
- 正規化 | 基礎
- 正規化 | 基礎
- 正規化 | サン

- マップ後 | カウント情報取得 | ゲノム | アノテーション無 | QuasR(Lerch XXX)

マップ後 | カウント情報取得 | ゲノム | アノテーション無 | QuasR(Lerch_XXX) NEW

QuasRパッケージを用いたsingle-end RNA-seqデータのリファレンスゲノム配列へのBowtieによるマッピングから、カウントデータ取得までの一連の流れを示します。アノテーション情報がない場合を想定しているので、GenomicRangesパッケージを利用して、マップされたリードの和集合領域(union range)を得たのち、領域ごとにマップされたリード数をカウントしています。

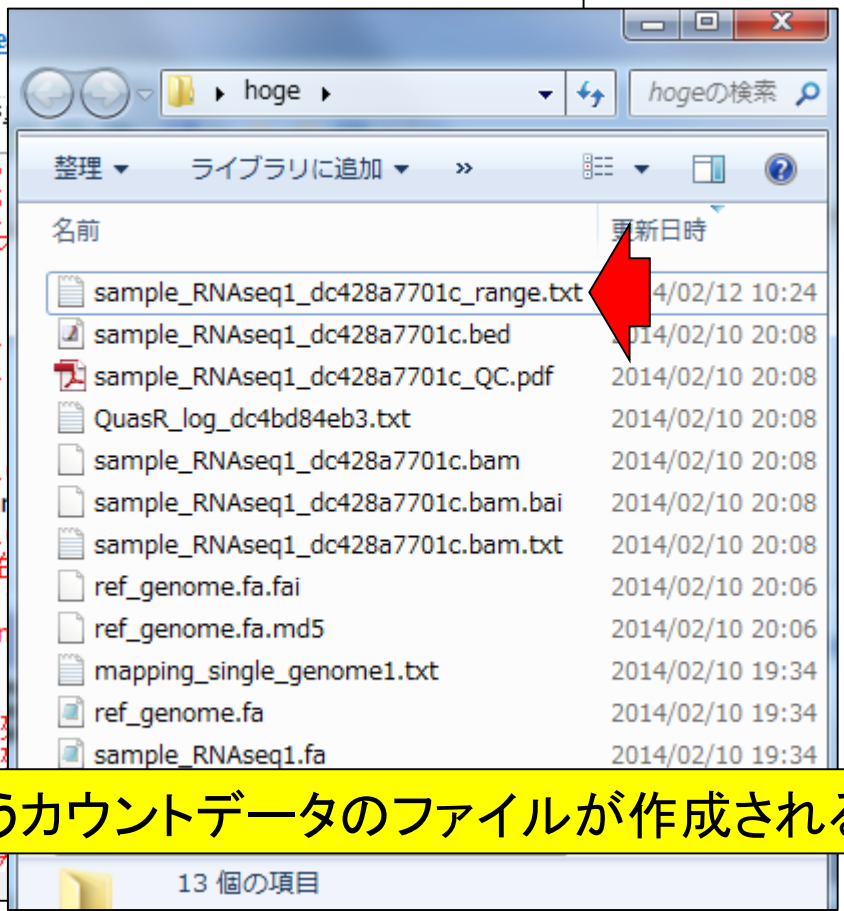
RNA-seqデータのほうのリード数は少ないですが、リファレンス配列の前処理でかなり時間がかかるようです(2時間とか...)。

「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18-20の複数のRNA-seqデータ(sample_RNAseq1.fa)をref_ge(mapping_single_genome1.txt):

複数サンプルのマッピング結果をまとめて和集合領域を定め、カウント情報を得

```
in_f1 <- "mapping_single_genome1.txt"  #入力ファイル名を指定
in_f2 <- "ref_genome.fa"               #入力ファイル名を指定
param_mapping <- "-m 1 --best --strata -v 0"#マッピング時の

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

#前処理(マッピング)
time_s <- proc.time()                  #計算時間を計測するた
out <- qAlign(in_f1, in_f2, alignmentParameter=param_mappi
time_e <- proc.time()                  #計算時間を計測するた
time_e - time_s                        #計算時間を表示(一番を
out                                    #マッピングに用いたパ
alignmentStats(out)                    #マッピング結果(align

#本番(マップされたリードの和集合領域同定)
tmpfname <- out@alignments[,1]         #ファイル名(in_f1の1
tmpsname <- out@alignments[,2]         #サンプル名(in_f1の2
for(i in 1:length(tmpfname)){
  if(i == 1){
    k <- readGAlignments(tmpfname[i])  #BAM形式ファイルを読
```



*_range.txtというカウントデータのファイルが作成される

マッピング結果からのカウント情報取得

マップ後 | カウント情報取得 | ゲノム | アノテーション無 | QuasR(Lerch_XXX)

QuasRパッケージを用いたsingle-end RNA-seqデータのリファレンスゲノム配列へのBowtieによるマッピングから、カ
タ取得までの一連の流れを示します。アノテーション情報がない場合を想定しているので、GenomicRanges パッケー
て、マップされたリードの和集合領域(union range)を得たのち、領域ごとにマップされたリード数をカウントしていま
RNA-seqデータのほうのリード数は少ないですが、リファレンス配列の前処理でかなり時間がかかるようです(2時間
「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18-20の複数のRNA-seqデータ(sample_RNAseq1.fa)をref_genome.faにマッピングする場合(mapping_single_genome1.txt):

複数サンプルのマッピング結果をまとめて和集合領域を定め、カウント情報を得るやり方です。サンプル間比較の

```
in_f1 <- "mapping_single_genome1.txt"  #入力ファイル名を指定してin_f1に格納(RNA-seqファ
in_f2 <- "ref_genome.fa"               #入力ファイル名を指定してin_f2に格納(リファレンス
param_mapping <- "-m 1 --best --strata -v 0"#マッピング時のオプションを指定
```

*.bed

chr1	11	45
chr2	1	35
chr2	16	50
chr3	1	35
chr3	3	37

	A	B
1	FileName	SampleName
2	sample_RNAseq1.fa	namae

*_range.txt

seqnames	start	end	width	strand	namae
chr1	11	45	35	+	1
chr2	1	50	50	+	2
chr3	1	37	37	+	2

カウント数はこちら

マッピング結果からのカウント情報取得

5.サンプルデータ18-20の複数のRNA-seqデータ(sample_RNAseq1.faとsample_RNAseq2.fa)をref_genome.faにマッピングする場合(mapping_single_genome4.txt):

全部のマッピング結果をまとめて和集合領域を定め、カウント情報を得るやり方です。一般的なカウントデータ行列の形式(2列目以降がカウント情報)にし、配列長情報と別々のファイルにして保存するやり方です。

```
in_f1 <- "mapping_single_genome4.txt"  #入力ファイル名を指定してin_f1に格納(RNA-seqファイル)
in_f2 <- "ref_genome.fa"               #入力ファイル名を指定してin_f2に格納(リファレンス配列)
out_f1 <- "hoge4_count.txt"            #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge4_genelength.txt"       #出力ファイル名を指定してout_f2に格納
param_mapping <- "-m 1 --best --strata -v 1"#マッピング時のオプションを指定
```

FileName	SampleName
sample_RNAseq1.fa	sample1
sample_RNAseq2.fa	sample2

tmp	sample1	sample2
chr1_11_45_35_+	1	0
chr2_1_60_60_+	2	1
chr3_1_37_37_+	2	0
chr4_6_65_60_+	0	1
chr5_1_35_35_+	1	0

リストファイル中で指定したサンプル名がカウントデータ行列の列名となる

よく見かけるカウントデータ取得手段

- basic alignerの1つであるBowtieを利用
- 最大2塩基ミスマッチまで許容してリファレンス配列の1か所とのみ一致するリード（uniquely mapped reads or unique mapper）数をカウント
  - Marioni et al., *Genome Res*., **18**:1509-1517, 2008
  - Bullard et al., *BMC Bioinformatics*, **11**:94, 2010
  - Risso et al., *BMC Bioinformatics*, **12**:480, 2011
  - ReCount (Frazee et al., *BMC Bioinformatics*, **12**:449, 2011)
  - …

SpliceMap (Au et al., 2010)などのsplice-aware alignerだと相当時間がかかるという現実的な問題もあるのだろう。講義や講習会では到底無理。
→ ユーザの記憶に残らない → 実際に使われない…

定量化：遺伝子レベル ⇔ isoformレベル

- 全体的な流れとしては遺伝子レベル → isoformレベル
  - 例：新規splice variantの発見（Twine et al., *PLoS One*, **6**: e16266, 2011）
- 遺伝子セット解析（Gene Ontology解析やパスウェイ解析など）のための基本情報は遺伝子レベルの解像度
- 複数エクソン　→　遺伝子レベルの要約統計量
  - exon union method (Mortazavi et al., *Nat. Methods*, **5**: 621-628, 2008)
    - **全て**のisoforms間で用いられているexonの情報（**union**：和集合）を利用
  - exon intersection method (Bullard et al., *BMC Bioinformatics*, **11**: 94, 2010)
    - 複数isoforms間で**共通**して用いられているexonの情報のみ（**intersection**：積集合）を利用

count情報を得る際に、どのexonの情報を用いるか?

遺伝子のカウント数の定義

- 算出された生リードカウント結果
  - exon union method（和集合）の場合：20 reads
  - Exon intersection method（積集合）の場合：11 reads

「Garber et al., *Nat. Methods*, **8**: 469-477, 2011」のFig. 3c

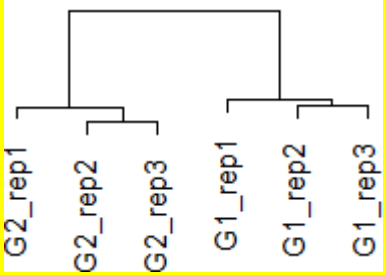
様々な思想があり、当然その後の解析結果に影響を及ぼします

Contents（Rで…）

- ゲノム解析
  - アノテーションファイルを読み込んで目的のキーワードを含む行のみ抽出
  - multi-FASTAファイルを自在に解析
    - 配列長分布、GC含量、フィルタリング、部分配列の切り出しなど
    - 連続塩基の出現頻度（CpG）解析、ゲノム配列取得など
- トランスクリプトーム解析
  - 研究目的別留意点：サンプル内とサンプル間の違い
  - マッピング → カウント情報取得
  - **データを眺める：クラスタリングやM-A plot**
  - 理想的な実験デザイン
  - なぜx倍発現変動という議論がだめなんですか？
  - モデルとか分布って、自分の解析結果にどういう影響を与えているの？
  - 多重比較問題：FDRって何？

サンプル間クラスタリング

出力:hoge1.png



解析 | クラスタリング | サンプル間 | hclust NEW

RNA-seqカウントデータのクラスタリング結果は、特にゼロカウント(0カウント; zero count)を多く含む場合に(もちろん距離の定義の仕方によっても変わってきますが)低発現データのフィルタリングの閾値次第で結果が変わる傾向にあります。ここでは、上記閾値問題に悩まされることなく頑健なサンプル間クラスタリングを行うやり方を示します。内部的に行っていることは、以下の通りです:

1. 全サンプルで0カウントとなる行(遺伝子)をフィルタリングした後、unique関数を用いて同一発現パターンのものを1つのパターンとしてまとめる、2.「1 - Spearman順位相関係数」でサンプル間距離を定義、3. Average-linkage clusteringの実行、です。

順位相関係数を用いてサンプルベクトル間の類似度として定義するので、サンプル間正規化の問題に悩まされません。また、低発現遺伝子にありがちな同一発現パターンの遺伝子をまとめることで、(変動しやすい)同順位となる大量の遺伝子が集約されるため、結果的に「総カウント数がx個以下のものをフィルタリング...」という閾値問題をクリアしたことになります。近いうち(2014年中)にTCCパッケージ中でwrapper functionを提供する予定です。

「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13の10,000 genes×6 samplesのカウントデータ(data_hypodata_3vs3.txt)の場合:

Biological replicatesを模倣したシミュレーションデータ(G1群3サンプル vs. G2群3サンプル)です。gene_1～gene_2000までがDEG(最初の1800個がG1群で高発現、残りの200個がG2群で高発現) gene_2001～gene_10000までがnon-DEGであることが既知です。

```
in_f <- "data_hypodata_3vs3.txt"          #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.png"                      #出力ファイル名(クラスタリング結果ファイル)を指定
param_fig <- c(500, 400)                  #ファイル出力時の横幅と縦幅を指定(単位はピクセル)

#入力ファイルの読み込み
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="")#
dim(data)                                 #オブジェクトdataの行数と列数を表示

#前処理(フィルタリング)
obj <- as.logical(rowSums(data) > 0)      #条件を満たすかどうかを判定した結果をobjに格納
```

ゼロカウントを含む低発現データのフィルタリングは重要です

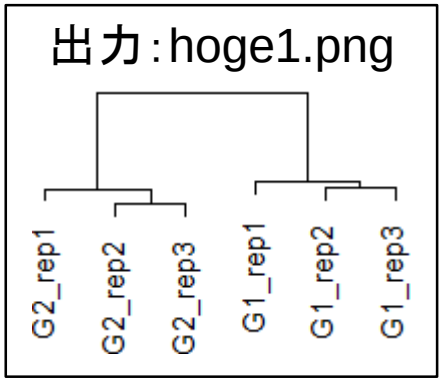
サンプル間クラスタリング

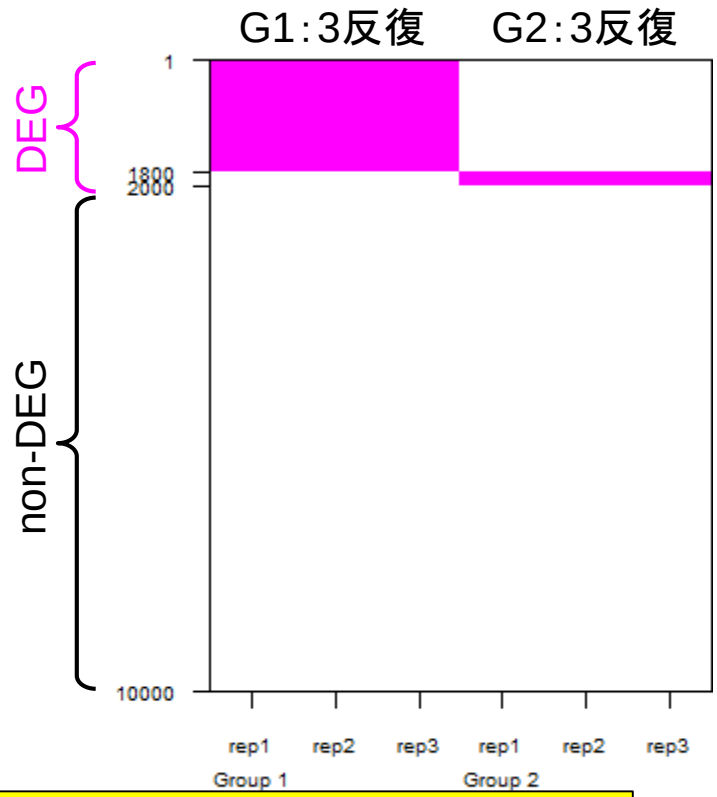
- data_hypodata_3vs3.txt（2群間比較用）
  - G1群：3サンプル、G2群：3サンプル
  - 全部で10,000行×6列。最初の2,000行分が発現変動遺伝子（DEG）

DEG
G1で高発現
G2で高発現
non-DEG

	G1_rep1	G1_rep2	G1_rep3	G2_rep1	G2_rep2	G2_rep3
gene_1	36	56	144	2	1	0
gene_2	84	152	124	52	37	28
gene_3	592	840	800	151	257	200
gene_4	0	8	4	1	1	3
gene_5	32	32	0	1	1	0
…						
gene_1801	34	86	24	284	180	364
gene_1802	5	1	3	0	160	24
gene_1803	57	56	51	248	192	220
gene_1804	29	25	32	128	204	160
gene_1805	42	29	44	184	156	92
…						
gene_2001	4	8	9	13	12	4
gene_2002	88	139	40	22	44	21
gene_2003	933	667	462	889	396	443
gene_2004	48	37	14	36	57	71
gene_2005	290	338	553	319	210	504
…						
gene_9996	107	67	104	35	65	45
gene_9997	145					
gene_9998	42					
gene_9999	5					
gene_10000	2					

出力：hoge1.png





DEGが多く存在するほど群間で明瞭なクラスターに分かれる傾向
→クラスタリング結果からDEGの有無をある程度把握可能です

サンプル間クラスタリング

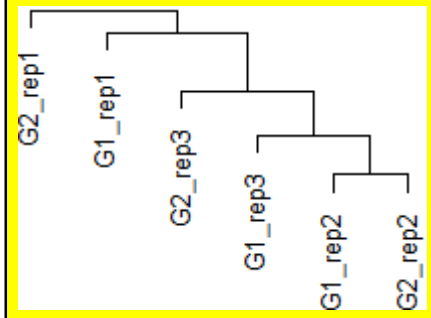
2. サンプルデータ13の10,000 genes×6 samplesのカウントデータ(data_hypodata_3vs3.txt)の場合:

Biological replicatesを模倣したシミュレーションデータ(G1群3サンプル vs. G2群3サンプル)です。gene_1～gene_2000までがDEG(最初の1800個がG1群で高発現、残りの200個がG2群で高発現) gene_2001～gene_10000までがnon-DEGであることが既知です。

non-DEGデータのみでクラスタリングを行っています。

```
in_f <- "data_hypodata_3vs3.txt"          #入力ファイル名を指定してin_fに格納
out_f <- "hoge2.png"                      #出力ファイル名(クラスタリング結果ファイル)を指定
param_fig <- c(500, 400)                  #ファイル出力時の横幅と縦幅を指定(単位はピクセル)
param_nonDEG <- 2001:10000                #non-DEGの位置を指定

#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で指定したファイルの読み
dim(data)                                 #オブジェクトdataの行数と列数を表示

#前処理(サブセットの抽出)
data <- data[param_nonDEG,]               #指定した行のみ抽出した結果をdataに格納

#前処理(フィルタリング)
obj <- as.logical(rowSums(data) > 0)      #条件を満たすかどうかを判定した結果をobjに格納
data <- unique(data[obj,])                #objがTRUEとなる行のみ抽出し、ユニークパターンのみにした結果をd
dim(data)                                 #オブジェクトdataの行数と列数を表示

#本番
data.dist <- as.dist(1 - cor(data, method="spearman"))#サンプル間の距離を計算し、結果をdata.distに格
out <- hclust(data.dist, method="average")#階層的クラスタリングを実行し、結果をoutに格納
png(out_f, pointsize=13, width=param_fig[1], height=param_fig[2])#出力ファイルの各種パラメータを指定
plot(out)                                 #樹形図（デンドログラム）の表示
```

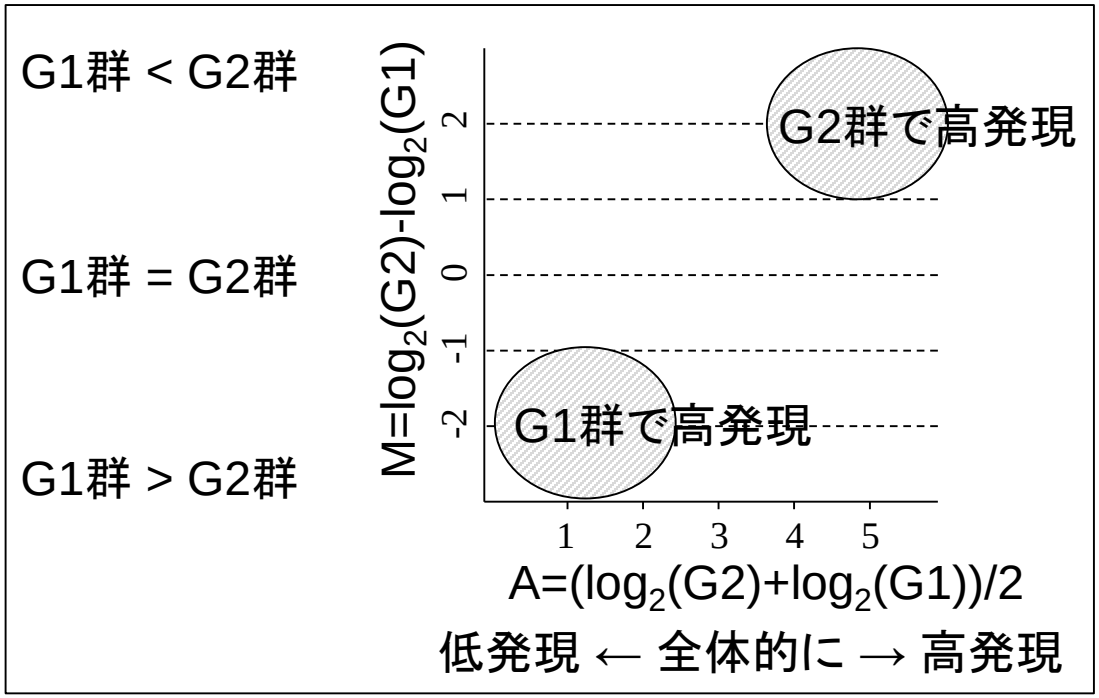
出力:hoge2.png



DEGが存在しないデータの典型的なクラスタリング結果です

M-A plot

- 2群間比較用
- 横軸が全体的な発現レベル、縦軸がlog比からなるプロット
- 名前の由来は、おそらく対数の世界での縦軸が引き算（Minus）、横軸が平均（Average）



DEGが存在しないデータのM−A plotを眺めることで、縦軸の閾値のみに相当する**倍率変化**を用いたDEG同定の危険性が分かります

- マップ後 | 配列長とカウント数の関係(last modified 2013/10/27)
- 正規化 | について(last modified 2013/06/21)
- 正規化 | 基礎 | RPK or CPK (配列長補正)(last modified 2013/07/03)
- 正規化 | 基礎 | RPM or CPM (総リード数補正)(last modified 2013/12/17)
- 正規化 | 基礎 | RPKM | トランスクリプトーム(last modified 2013/06/23)
- 正規化 | 基礎 | RPKM | ゲノム(last modified 2013/09/18)
- 正規化
- 正規化
- 正規化

• 正規化 | 基礎 | RPM or CPM (総リード数補正)

正規化 | 基礎 | RPM or CPM (総リード数補正)

カウントデータファイルを読み込んで、転写物ごとのリード数を「総リード数が100万 (million)だったときのリード数; Reads per million (RPM)」に変換するやり方を示します。「リード数＝カウント数」なのでReadsのところをCountsに置き換えた表現(Counts per million; CPM)もときどき見受けられます。

「ファイル」

1. 3列目に力

```
in_f <- "s
out_f <- "
param1 <-

#入力ファイ
data <- re
head(data)
sum(data[,

#本番(正規
nf <- para
out <- dat
head(out)
sum(out)

#ファイルに
tmp <- cbi
colnames(t
write.tabl
```

4. サンプルデータ13の10,000 genes×6 samplesのカウントデータ(data_hypodata_3vs3.txt)の場合:

Biological replicatesを模倣したシミュレーションデータ(G1群3サンプル vs. G2群3サンプル)です。gene_1～gene_2000までがDEG (最初の1800個がG1群で高発現、残りの200個がG2群で高発現) gene_2001～gene_10000までがnon-DEGであることが既知です。

non-DEGデータのみを正規化してM-A plotを作成していろいろとハイライトさせるやり方です。

```
in_f <- "data_hypodata_3vs3.txt"        #入力ファイル名を指定してin_fに格納
param1 <- 1000000                       #補正後の総リード数を指定(RPMにしたい場合はここの数値
param_nonDEG <- 2001:10000              #non-DEGの位置を指定
param_G1 <- 3                           #G1群のサンプル数を指定
param_G2 <- 3

#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

#前処理(non-DEGデータのみ抽出)
data <- data[param_nonDEG,]             #param_nonDEGで指定した行の情報のみ抽出

#本番(正規化)
nf <- param1/colSums(data)              #正規化係数を計算した結果をnfに格納
data <- sweep(data, 2, nf, "*")         #正規化係数を各列に掛けた結果をdataに格納

#本番(M-A plot)
data.cl <- c(rep(1, param_G1), rep(2, param_G2))#G1群を1、G2群を2としたベクトルdata.clを作
G1 <- apply(as.matrix(data[,data.cl==1]), 1, mean)#遺伝子ごとにG1群の平均を計算した結果をG1
G2 <- apply(as.matrix(data[,data.cl==2]), 1, mean)#遺伝子ごとにG2群の平均を計算した結果をG2
M <- log2(G2) - log2(G1)                #M-A plotのM(y軸の値)に相当するものをMに格納)
A <- (log2(G1) + log2(G2))/2            #M-A plotのA(x軸の値)に相当するものをAに格納)
```

RPM補正後のnon-DEGデータを用いてM-A plotを描画します

4. サンプルデータ13の10,000 genes×6 samplesのカウントデータ(data_hypodata_3vs3.txt)の場合:

Biological replicatesを模倣したシミュレーションデータ(G1群3サンプル vs. G2群3サンプル)です。gene_1～gene_2000までがDEG (最初の1800個がG1群で高発現、残りの200個がG2群で高発現) gene_2001～gene_10000までがnon-DEGであることが既知です。
non-DEGデータのみを正規化してM-A plotを作成していろいろとハイライトさせるやり方です。

```
in_f <- "data_hypodata_3vs3.txt"                    #入力ファイルを指定して、dに格納
param1 <- 1000000
param_nonDEG <- 2001:10000
param_G1 <- 3
param_G2 <- 3

#入力ファイルの読み込み
data <- read.table(in_f, header=T                   ="")#in_fで指定したファ

#前処理(non-DEGデータのみ抽出)
data <- data[param_nonDEG,]                         情報のみ抽出

#本番(正規化)
nf <- param1/colSums(dat
data <- sweep(data, 2, n

#本番(M-A plot)
data.cl <- c(rep(1, para
G1 <- apply(as.matrix(da
G2 <- apply(as.matrix(da
M <- log2(G2) - log2(G1)
A <- (log2(G1) + log2(G2
```

切り取り(T)
コピー(C)
貼り付け
すべて選択(A)
印刷(I)...
印刷プレビュー(N)...
Bing でマップ
Bing で翻訳

R Console

```
> dim(data)
[1] 8000    6
> head(data, n=5)
              G1_rep1   G1_rep2   G1_rep3    G2_rep1   G2_rep2   G2_rep3
gene_2001    5.292335  10.86076  11.62478   16.90668  15.24977   5.42827
gene_2002  116.431378 188.70563  51.66570   28.61130  55.91583  28.49842
gene_2003 1234.437226 905.51550 596.73886 1156.15657 503.24248 601.18092
gene_2004   63.508025  50.23099  18.08300   46.81849  72.43642  96.35180
gene_2005  383.694315 458.86692 714.27833  414.86383 266.87101 683.96204
> colSums(data)
G1_rep1 G1_rep2 G1_rep3 G2_rep1 G2_rep2 G2_rep3
  1e+06   1e+06   1e+06   1e+06   1e+06   1e+06
>
```

ここまでは100万に揃えるという総リード数補正結果の確認

4. サンプルデータ13の10,000 genes×6 samplesのカウントデータ(data_hypodata_3vs3.txt)の場合:

Biological replicatesを模倣したシミュレーションデータ(G1群3サンプル vs. G2群3サンプル)です。gene_1～gene_2000までがDEG (最初の1800個がG1群で高発現、残りの200個がG2群で高発現) gene_2001～gene_10000までがnon-DEGであることが既知です。
non-DEGデータのみを正規化してM-A plotを作成していろいろとハイライトさせるやり方です。

```
#本番(正規化)
nf <- param1/colSums(data)              #正規化係数を計算した結果をnfに格納
data <- sweep(data, 2, nf, "*")         #正規化係数を各列に掛けた結果をdataに格納

#本番(M-A plot)
data.cl <- c(rep(1, param_G1), non(2, param
G1 <- apply(as.matrix(data[,da
G2 <- apply(as.matrix(data[,da
M <- log2(G2) - log2(G1)
A <- (log2(G1) + log2(G2))/2
plot(A, M, xlab="A = (log2(G2)
grid(col="gray", lty="dotted")

#後処理(いろいろな条件を満たすも
obj <- "gene_2002"
points(A[obj], M[obj], col="re

obj <- (M >= log2(4))
points(A[obj], M[obj], col="bl
sum(obj, na.rm=TRUE)

obj <- (A > 10)
```

切り取り(T)
コピー(C)
貼り付け
すべて選択(A)
印刷(I)...
印刷プレビュー
Bing でマップ
Bing で翻訳
Google で検索
電子メール (W
すべてのアク
Send to OneN

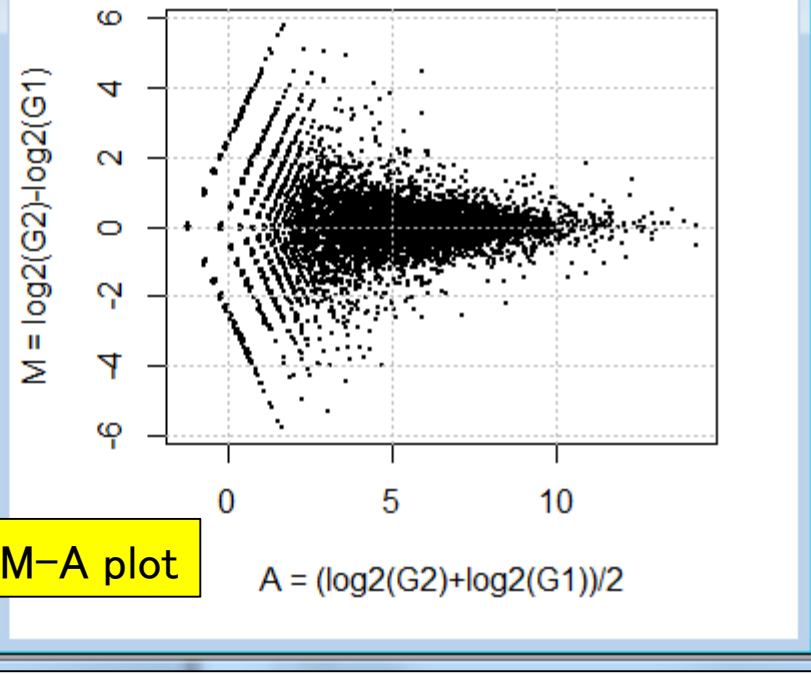
RGui (64-bit)

ファイル　履歴　サイズ変更　ウインドウ

R Console

```
gene_2005  383.694315 458.866
> colSums(data)
G1_rep1 G1_rep2 G1_rep3 G2_re
  1e+06   1e+06   1e+06   1e+
> data.cl <- c(rep(1, param_G
> G1 <- apply(as.matrix(data[
> G2 <- apply(as.matrix(data[
> M <- log2(G2) - log2(G1)
> A <- (log2(G1) + log2(G2))/
> plot(A, M, xlab="A = (log2(
> grid(col="gray", lty="dotte
>
>
```

R Graphics: Device 2 (ACTIVE)



RPM補正後のデータを用いてM-A plot

M-A plot作成手順

non-DEGのデータのみで
RPM正規化したデータ

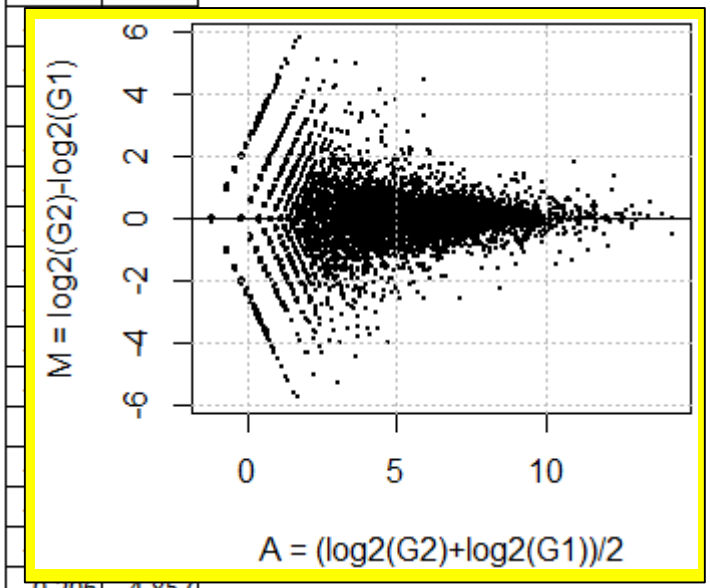
各群の
平均値

log_2変換

log_2(G2/G1)
log比

平均発現量

	G1_rep1	G1_rep2	G1_rep3	G2_rep1	G2_rep2	G2_rep3	G1	G2	logG1	logG2	M	A
gene_2001	5.3	10.9	11.6	16.9	15.2	5.4	9.3	12.5	3.211	3.647	0.436	3.429
gene_2002	116.4	188.7	51.7	28.6	55.9	28.5	118.9	37.7	6.894	5.236	-1.658	6.065
gene_2003	1234.4	905.5	596.7	1156.2	503.2	601.2	912.2	753.5	9.833	9.558	-0.276	9.695
gene_2004	63.5	50.2	18.1	46.8	72.4	96.4	43.9	71.9	5.457	6.167	0.710	5.812
gene_2005	383.7	458.9	714.3	414.9	266.9	684.0	518.9	455.2	9.019	8.830	-0.189	8.925
gene_2006	23.8	33.9	20.7	35.1	25.4	69.2	26.1	43.2	4.708	5.435	0.726	5.071
gene_2007	129.7	126.3	59.4	70.2	75.0	88.2	105.1	77.8	6.716	6.282		
gene_2008	909.0	590.6	1034.6	926.0	560.4	348.8	844.7	611.7	9.722	9.257		
gene_2009	0.0	0.0	11.6	15.6	0.0	0.0	3.9	5.2	1.954	2.379		
gene_2010	398.2	313.6	337.1	330.3	294.8	416.6	349.7	347.3	8.450	8.440		
gene_2011	64.8	111.3	100.7	76.7	59.7	71.9	92.3	69.5	6.528	6.118		
gene_2012	47.6	86.9	60.7	40.3	38.1	70.6	65.1	49.7	6.024	5.634		
gene_2013	0.0	36.7	0.0	13.0	6.4	2.7	12.2	7.4	3.611	2.879		
gene_2014	1.3	2.7	1.3	0.0	0.0	2.7	1.8	0.9	0.829	-0.144		
gene_2015	1.3	92.3	0.0	2.6	15.2	0.0	31.2	6.0	4.964	2.573		
gene_2016	5.3	1.4	0.0	5.2	0.0	0.0	2.2	1.7	1.148	0.794		
gene_2017	6.6	13.6	20.7	16.9	15.2	13.6	13.6	15.2	3.768	3.930		
gene_2018	2.6	6.8	16.8	5.2	8.9	5.4	8.7	6.5	3.128	2.702		
gene_2019	328.1	296.0	229.9	317.3	242.7	270.1	284.7	276.7	8.153	8.112		
gene_2020	2.6	0.0	2.6	0.0	0.0	1.4	1.7	0.5	0.802	-1.144		
gene_2021	23.8	32.6	24.5	20.8	30.5	42.1	27.0	31.1	4.754	4.960	0.206	4.857
gene_2022	307.0	186.0	222.2	352.4								
gene_2023	25.1	25.8	27.1	41.6								
gene_2024	60.9	54.3	20.7	37.7	47.0	33.9	45.3	39.6	5.501	5.306	-0.195	5.403
gene_2025	15.9	42.1	6.5	27.3	49.6	27.1	21.5	34.7	4.424	5.116	0.691	4.770
gene_2026	2.6	1.4	10.3	0.0	0.0	0.0	4.8	0.0	2.257	#####	#######	#######
gene_2027	365.2	272.9	313.9	273.1	331.7	479.0	317.3	361.3	8.310	8.497	0.187	8.403



任意の遺伝子gene_2002をハイライトさせたいときは？

4. サンプルデータ13の10,000 genes×6 samplesのカウントデータ(data_hypodata_3vs3.txt)の場合:

Biological replicatesを模倣したシミュレーションデータ(G1群3サンプル vs. G2群3サンプル)です。gene_1～gene_2000までがDEG (最初の1800個がG1群で高発現、残りの200個がG2群で高発現) gene_2001～gene_10000までがnon-DEGであることが既知です。
non-DEGデータのみを正規化してM-A plotを作成していろいろとハイライトさせるやり方です。

```
#本番(M-A plot)
data.cl <- c(rep(1, param_G1), rep(2, param_
G1 <- apply(as.matrix(data[,data.cl==1]), 1,
G2 <- apply(as.matrix(data[,data.cl==2]), 1,
M <- log2(G2) - log2(G1)                #M-A
A <- (log2(G1) + log2(G2))/2            #M-A
plot(A, M, xlab="A = (log2(G2)+log2(G1))/2",
grid(col="gray", lty="dotted")          #指定

#後処理(いろいろな条件を満たすものを描画している
obj <- "gene_2002"
points(A[obj], M[obj

obj <- (M >= log2(4)
points(A[obj], M[obj
sum(obj, na.rm=TRUE)

obj <- (A > 10)
points(A[obj], M[obj
sum(obj, na.rm=TRUE)
```

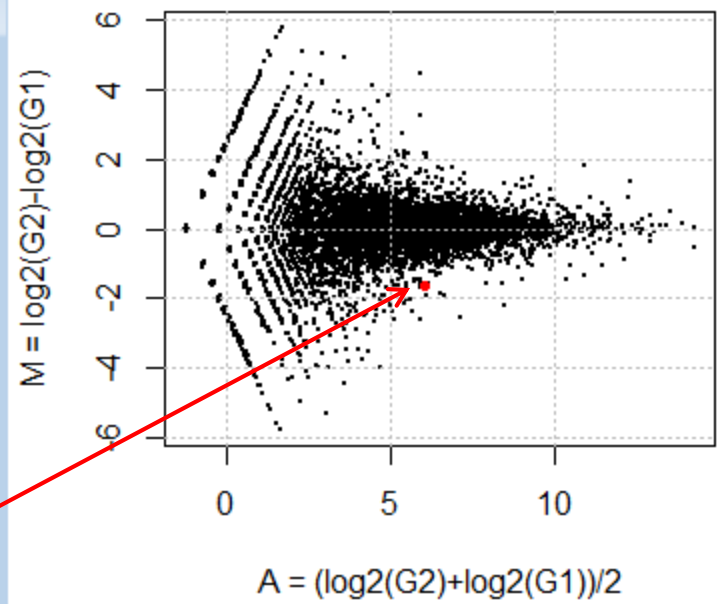
切り取り(T)
コピー(C)
貼り付け
すべて選択(A)
印刷(I)...
印刷プレビュー(N)...
Bing でマップ
Bing で翻訳

RGui (64-bit)
ファイル　履歴　サイズ変更　ウインドウ

R Console

```
   1e+06    1e+06    1e+06    1e+
> data.cl <- c(rep(1, param_G
> G1 <- apply(as.matrix(data[
> G2 <- apply(as.matrix(data[
> M <- log2(G2) - log2(G1)
> A <- (log2(G1) + log2(G2))/
> plot(A, M, xlab="A = (log2(
> grid(col="gray", lty="dotte
>
> obj <- "gene_2002"
> points(A[obj], M[obj], col=
>
>
```

R Graphics: Device 2 (ACTIVE)



gene_2002のようなG1群で高発現遺伝子のM値はマイナスの位置にプロットされます

4. サンプルデータ13の10,000 genes×6 samplesのカウントデータ(data_hypodata_3vs3.txt)の場合:

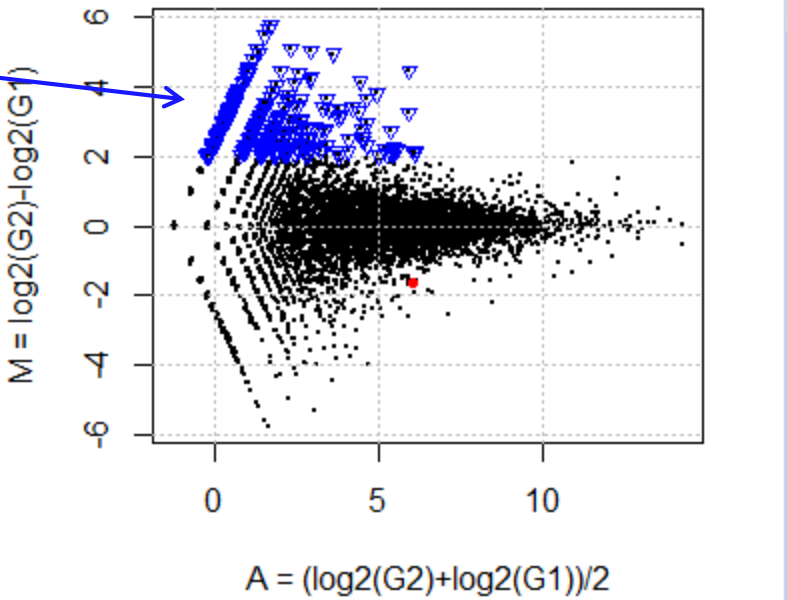
Biological replicatesを模倣したシミュレーションデータ(G1群3サンプル vs. G2群3サンプル)です。gene_1～gene_2000までがDEG (最初の1800個がG1群で高発現、残りの200個がG2群で高発現) gene_2001～gene_10000までがnon-DEGであることが既知です。
non-DEGデータのみを正規化してM-A plotを作成していろいろとハイライトさせるやり方です。

```
#本番(M-A plot)
data.cl <- c(rep(1, param_G1), rep(2, param_G2))#G1群を1、G2群を2としたベクトルdata.clを作
G1 <- apply(as.matrix(data[,data.cl==1]), 1, mean)#遺伝子ごとにG1群の平均を計算した結果をG1
G2 <- apply(as.matrix(data[,data.cl==2]), 1, mean)#遺伝子ごとにG2群の平均を計算した結果をG2
M <- log2(G2) - log2(G1)                #M-A plotのM(y軸の値)に相当するものをMに格納)
A <- (log2(G1) + log2(G2))/2            #M-A plotのA(x軸の
plot(A, M, xlab="A = (log2(G2)+log2(G1))/2", ylab="M = lo
grid(col="gray", lty="dotted")          #指定したパラメータ

#後処理(いろいろな条件を満たすものを描画している)
obj <- "gene_2002"                      #ハイライトさせたい遺
points(A[obj], M[obj], col="red", cex=1.0, pch=20)#objがTR

obj <- (M >= log2(4))                   #条件を満たすかどうか
points(A[obj], M[obj], col="blue", cex=0.8, pch=6)#objがTR
sum(obj, na.rm=TRUE)                    #objがTRUEとなる要素

obj <- (A > 10)                         #条件を満たすかどうか
points(A[obj], M[obj], col="lightblue", cex=0.8, pch=17)#o
sum(obj, na.rm=TRUE)                    #objがTRUEとなる要素
```

G2群で4倍以上高発現という条件(499個)も可能です

R Console

```
> A <- (log2(G1)
> plot(A, M, xlab
> grid(col="gray"
>
> obj <- "gene_20
> points(A[obj],
>
> obj <- (M >= lo
> points(A[obj],
> sum(obj, na.rm=
[1] 499
>
>
```

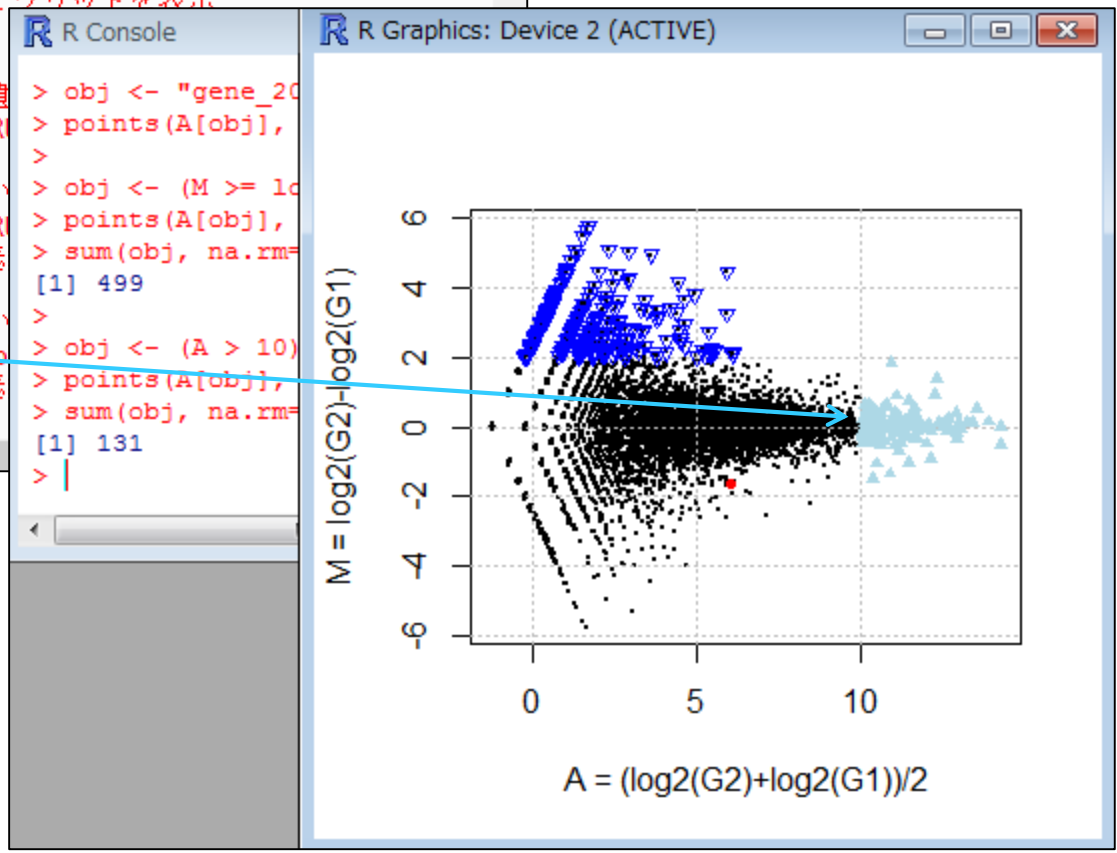


4. サンプルデータ13の10,000 genes×6 samplesのカウントデータ(data_hypodata_3vs3.txt)の場合:

Biological replicatesを模倣したシミュレーションデータ(G1群3サンプル vs. G2群3サンプル)です。gene_1～gene_2000までがDEG (最初の1800個がG1群で高発現、残りの200個がG2群で高発現) gene_2001～gene_10000までがnon-DEGであることが既知です。
non-DEGデータのみを正規化してM-A plotを作成していろいろとハイライトさせるやり方です。

```
#本番(M-A plot)
data.cl <- c(rep(1, param_G1), rep(2, param_G2))#G1群を1、G2群を2としたベクトルdata.clを作
G1 <- apply(as.matrix(data[,data.cl==1]), 1, mean)#遺伝子ごとにG1群の平均を計算した結果をG1
G2 <- apply(as.matrix(data[,data.cl==2]), 1, mean)#遺伝子ごとにG2群の平均を計算した結果をG2
M <- log2(G2) - log2(G1)                #M-A plotのM(y軸の値)に相当するものをMに格納)
A <- (log2(G1) + log2(G2))/2            #M-A plotのA(x軸の
plot(A, M, xlab="A = (log2(G2)+log2(G1))/2", ylab="M = lo
grid(col="gray", lty="dotted")          #指定したパラメータ

#後処理(いろいろな条件を満たすものを描画している)
obj <- "gene_2002"                      #ハイライトさせたい遺
points(A[obj], M[obj], col="red", cex=1.0, pch=20)#objがTR

obj <- (M >= log2(4))                   #条件を満たすかどうか
points(A[obj], M[obj], col="blue", cex=0.8, pch=6)#objがTR
sum(obj, na.rm=TRUE)                    #objがTRUEとなる要素

obj <- (A > 10)                         #条件を満たすかどうか
points(A[obj], M[obj], col="lightblue", cex=0.8, pch=17)#o
sum(obj, na.rm=TRUE)                    #objがTRUEとなる要素
```

A値が10より大きいという条件(131個)も可能です



M-A plot作成手順

non-DEGのデータのみで
RPM正規化したデータ

	G1_rep1	G1_rep2	G1_rep3	G2_rep1	G2_rep2	G2_rep3
gene_2001	5.3	10.9	11.6	16.9	15.2	5.4
gene_2002	116.4	188.7	51.7	28.6	55.9	28.5
gene_2003	1234.4	905.5	596.7	1156.2	503.2	601.2
gene_2004	63.5	50.2	18.1	46.8	72.4	96.4
gene_2005	383.7	458.9	714.3	414.9	266.9	684.0
gene_2006	23.8	33.9	20.7	35.1	25.4	69.2
gene_2007	129.7	126.3	59.4	70.2	75.0	88.2
gene_2008	909.0	590.6	1034.6	926.0	560.4	348.8
gene_2009	0.0	0.0	11.6	15.6	0.0	0.0
gene_2010	398.2	313.6	337.1	330.3	294.8	416.6
gene_2011	64.8	111.3	100.7	76.7	59.7	71.9
gene_2012	47.6	86.9	60.7	40.3	38.1	70.6
gene_2013	0.0	36.7	0.0	13.0	6.4	2.7
gene_2014	1.3	2.7	1.3	0.0	0.0	2.7
gene_2015	1.3	92.3	0.0	2.6	15.2	0.0
gene_2016						
gene_2017						
gene_2018						
gene_2019						
gene_2020	2.6	0.0	2.6	0.0	0.0	1.4
gene_2021	23.8	32.6	24.5	20.8	30.5	42.1
gene_2022	307.0	186.0	222.2	352.4	346.9	350.1
gene_2023	25.1	25.8	27.1	41.6	12.7	19.0
gene_2024	60.9	54.3	20.7	37.7	47.0	33.9
gene_2025	15.9	42.1	6.5	27.3	49.6	27.1
gene_2026	2.6	1.4	10.3	0.0	0.0	0.0
gene_2027	365.2	272.9	313.9	273.1	331.7	479.0

各群の
平均値

G1	G2
9.3	12.5
118.9	37.7
912.2	753.5
43.9	71.9
518.9	455.2
26.1	43.2
105.1	77.8
844.7	611.7
3.9	5.2
349.7	347.3
92.3	69.5
65.1	49.7
12.2	7.4
1.8	0.9
31.2	6.0
1.7	0.5
27.0	31.1
238.4	349.8
26.0	24.4
45.3	39.6
21.5	34.7
4.8	0.0
317.3	361.3

$\log_2$変換

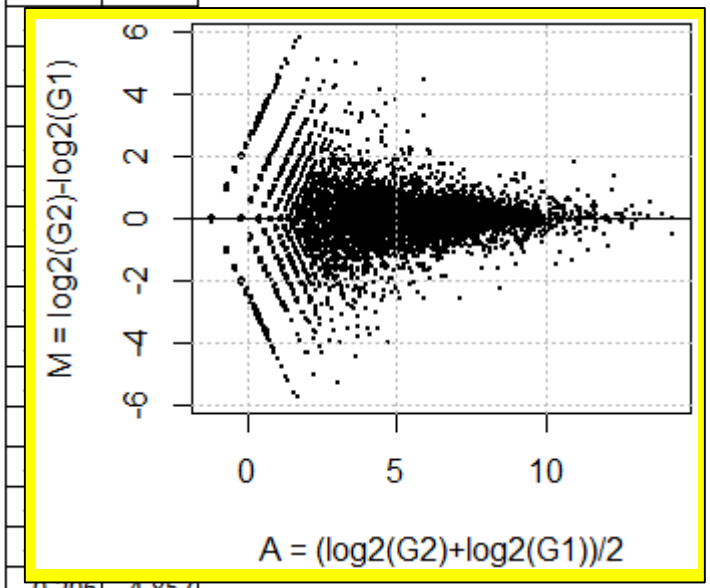
logG1	logG2
3.211	3.647
6.894	5.236
9.833	9.558
5.457	6.167
9.019	8.830
4.708	5.435
6.716	6.282
9.722	9.257
1.954	2.379
8.450	8.440
6.528	6.118
6.024	5.634
3.611	2.879
0.829	-0.144
4.964	2.573
0.802	-1.144
4.754	4.960
7.897	8.451
4.702	4.611
5.501	5.306
4.424	5.116
2.257	######
8.310	8.497

$\log_2$(G2/G1)
log比

平均発現量

M	A
0.436	3.429
-1.658	6.065
-0.276	9.695
0.710	5.812
-0.189	8.925
0.726	5.071
0.206	4.857
0.553	8.174
-0.090	4.656
-0.195	5.403
0.691	4.770
#######	#######
0.187	8.403

どちらか一方の群で平均値がゼロカウントになる遺伝子は通常のM-A plot上にはプロットできません

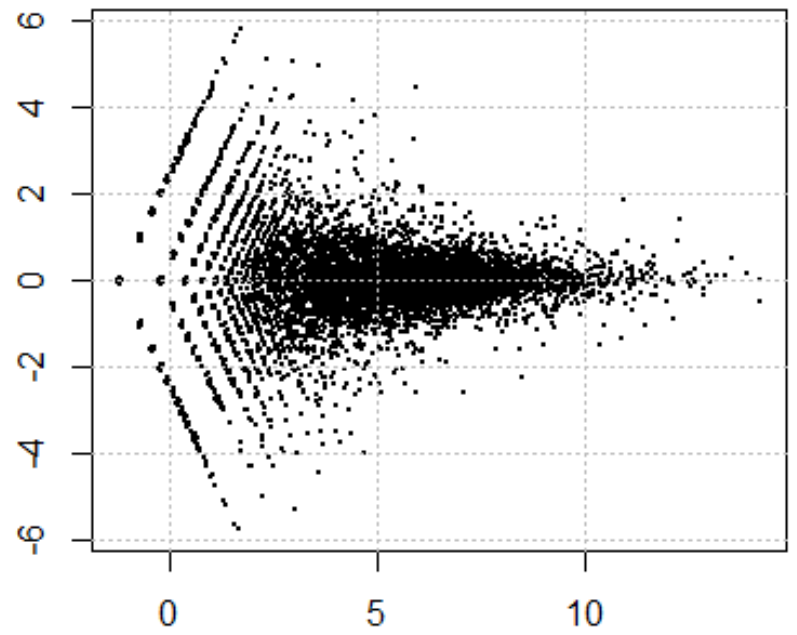


M-A plot作成手順

R Console

```
> obj <- "gene_2026"
> points(A[obj],M[obj], col="red", cex=1.0, pch=20)
> data[obj,]
            G1_rep1  G1_rep2  G1_rep3 G2_rep1 G2_rep2 G2_rep3
gene_2026 2.646168 1.357594 10.33314       0       0       0
> G1[obj]
gene_2026
 4.778968
> G2[obj]
gene_2026
        0
> log2(G1[obj])
gene_2026
 2.256699
> log2(G2[obj])
gene_2026
     -Inf
> M[obj]
gene_2026
     -Inf
> A[obj]
gene_2026
     -Inf
>
```

R Graphics: Device 2 (ACTIVE)



どちらか一方の群で平均値がゼロカウントになる遺伝子は通常のM-A plot上にはプロットできません

gene_2022											0.553	8.174
gene_2023	25.1	25.8	27.1	41.6	12.7	19.0	26.0	24.4	4.702	4.611	-0.090	4.656
gene_2024	60.9	54.3	20.7	37.7	47.0	33.9	45.3	39.6	5.501	5.306	-0.195	5.403
gene_2025	15.9	42.1	6.5	27.3	49.6	27.2	21.5	34.7	4.424	5.116	0.691	4.770
gene_2026	2.6	1.4	10.3	0.0	0.0	0.0	4.8	0.0	2.257	#####	#######	#######
gene_2027	365.2	272.9	313.9	273.1	331.7	479.0	317.3	361.3	8.310	8.497	0.187	8.403

- マップ後 | 配列長とカウント数の関係(last modified 2013/10/27)
- 正規化 | について(last modified 2013/06/21)
- 正規化 | 基礎 | RPK or CPK (配列長補正)(last modified 2013/07/03)
- 正規化 | 基礎 | RPM or CPM (総リード数補正)(last modified 2013/12/17)
- 正規化 | 基礎 | RPKM | トランスクリプトーム(last modified 2013/06/23)
- 正規化 | 基礎 | RPKM | ゲノム(last modified 2013/09/18)
- 正規化
- 正規化
- 正規化

参考

- 正規化 | 基礎 | RPM or CPM (総リード数補正)

正規化 | 基礎 | RPM or CPM (総リード数補正)

カウントデータファイルを読み込んで、転写物ごとのリード数を「総リード数が100万 (million)だったときのリード数; Reads per million (RPM)」に変換するやり方を示します。「リード数＝カウント数」なのでReadsのところをCountsに置き換えた表現(Counts per million; CPM)もときどき見受けられます。

「ファイル」

1. 3列目に力

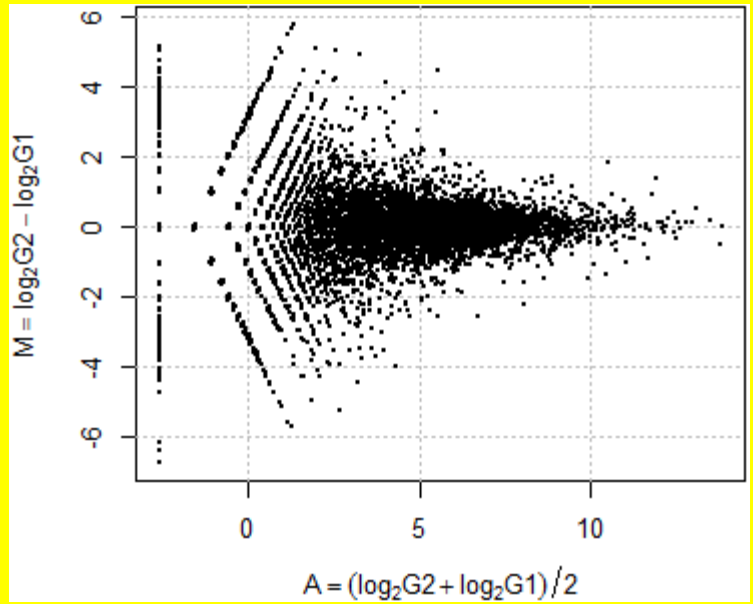
9. サンプルデータ13の10,000 genes×6 samplesの力

*TCC*を含むRNA-seq用パッケージはプロットできます

Biological replicatesを模倣したシミュレーションデー
(最初の1800個がG1群で高発現、残りの200個がG2群で高発現) gene_2001～gene_10000までがnon-DEGであることが既知です。
2.をTCCパッケージを利用して行うやり方です。各サンプルのリード数を100万ではなく平均リード数に揃えています。
non-DEGデータのみを正規化してM-A plotを作成するやり方です。

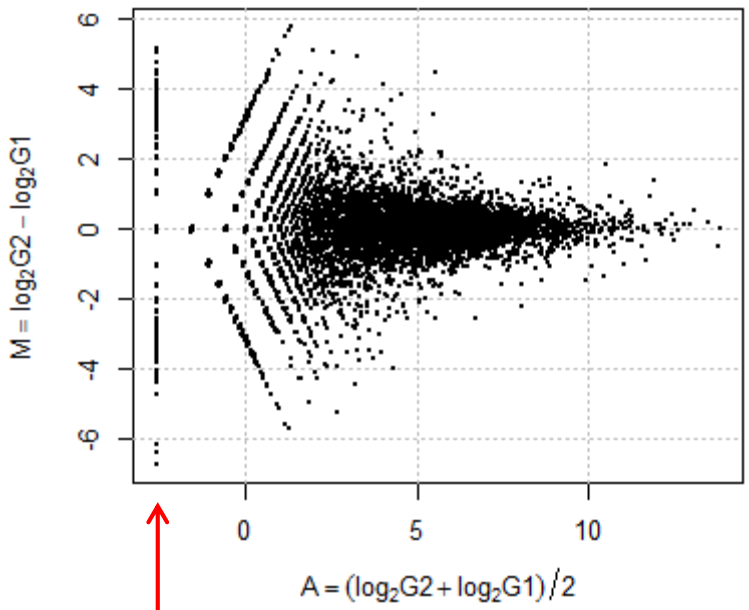
```
in_f <- "data_hypodata_3vs3.txt"          #入力ファイル名を指定
out_f1 <- "hoge9.txt"                     #出力ファイル名を指定
out_f2 <- "hoge9.png"                     #出力ファイル名を指定
param_G1 <- 3                             #G1群のサンプル数を
param_G2 <- 3                             #G2群のサンプル数を
param_nonDEG <- 2001:10000                #non-DEGの位置を指定
param_fig <- c(400, 380)                  #MA-plot描画時の横幅


#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="\

#前処理(サブセットの抽出とTCCクラスオブジェクトの作成)
data <- data[param_nonDEG,]               #param_nonDEGで指定
data.cl <- c(rep(1, param_G1), rep(2, param_G2))#G1群を1、
tcc <- new("TCC", data, data.cl)          #TCCクラスオブジェク

#ファイルに保存(テキストファイル)
data <- getNormalizedData(tcc)            #正規化後のデータを取り出してdataに格納
```

出力：hoge9.png



M-A plot：*TCC*パッケージの0カウント対策

出力：hoge9.png



①各群について、ゼロでない平均発現量の最小値を取得
②0だったところをその値で置換
③M値を再計算
④M-A plotの左側に、再計算して得られたM値をプロット

Contents（Rで…）

- ゲノム解析
  - アノテーションファイルを読み込んで目的のキーワードを含む行のみ抽出
  - multi-FASTAファイルを自在に解析
    - 配列長分布、GC含量、フィルタリング、部分配列の切り出しなど
    - 連続塩基の出現頻度（CpG）解析、ゲノム配列取得など
- トランスクリプトーム解析
  - 研究目的別留意点：サンプル内とサンプル間の違い
  - マッピング → カウント情報取得
  - データを眺める：クラスタリングやM-A plot
  - **理想的な実験デザイン**
  - なぜx倍発現変動という議論がだめなんですか？
  - モデルとか分布って、自分の解析結果にどういう影響を与えているの？
  - 多重比較問題：FDRって何？

理想的な実験デザイン

- 腎臓 vs. 肝臓のようなG1群 vs. G2群の比較の場合
  - 生のリードカウントのデータ（基本的には整数値）

Gene ID	A1	A2	A3	A4	…	B1	B2	B3	B4	…
Gene1										
Gene2										
Gene3										
Gene4										
Gene5										
Gene6										
Gene7										
…										

G1_rep1：ある生物の**腎臓**
G1_rep2：同じ生物種の別個体の**腎臓**
G1_rep3：同じ生物種のさらに別個体の**腎臓**
…
G2_rep1：ある生物の**肝臓**
G2_rep2：同じ生物種の別個体の**肝臓**
…

Biological replicatesのデータ
生物学的なばらつき（個体間の違い）を考慮すべし

2群間比較：technical replicatesデータ

- data_marioni.txt（ヒトのデータ）


kidney（腎臓）


liver（肝臓）

18,110 genes

rownames(data)	R1L1Kidney	R1L3Kidney	R1L7Kidney	R2L2Kidney	R2L6Kidney	R1L2Liver	R1L4Liver	R1L6Liver	R1L8Liver	R2L3Liver
ENSG00000000003	178	167	179	172	151	138	178	175	187	169
ENSG00000000005	0	0	0	0	1	0	0	0	0	0
ENSG00000000419	53	78	64	72	71	30	42	41	33	43
ENSG00000000457	22	33	30	27	30	47	60	37	42	62
ENSG00000000460	9	7	18	14	9	19	9	13	14	19
ENSG00000000938	14	18	7	27	15	42	34	39	37	45
ENSG00000000971	28									
ENSG00000001036	154									
ENSG00000001084	77									
ENSG00000001167	41									
ENSG00000001460	24									
ENSG00000001461	23									
ENSG00000001497	55									
ENSG00000001561	139									
ENSG00000001617	136									

Technical replicatesのデータ

レーン間の違いなどサンプル内の技術的なばらつきの度合いを調べるための同一個体由来データ。このようなデータで2群間比較し、**発現変動遺伝子がどの程度あるかといった数に関する議論はほぼ無意味。**
理由：Biological variation > Technical variation。**得られた結果はその個体内のみで成立**するものであり、同じ生物種の別個体においても同様な事象が観測されるわけではない。

2群間比較：technical replicatesデータ

- data_marioni.txt (ヒトのデータ)


kidney（腎臓）


liver（肝臓）

18,110 genes

rownames(data)	R1L1Kidney	R1L3Kidney	R1L7Kidney	R2L2Kidney	R2L6Kidney	R1L2Liver	R1L4Liver	R1L6Liver	R1L8Liver	R2L3Liver
ENSG00000000003	178	167	179	172	151	138	178	175	187	169
ENSG00000000005	0	0	0	0	1	0	0	0	0	0
ENSG00000000419	53	78	64	72	71	30	42	41	33	43
ENSG00000000457	22	33	30	27	30	47	60	37	42	62
ENSG00000000460	9	7	18	14	9	19	9	13	14	19
ENSG00000000938	14	18	7	27	15	42	34	39	37	45
ENSG00000000971	28	27	28	34	42	470	502	500	490	536
ENSG00000001036	154	195	140	168	187	56	67	55	64	67
ENSG00000001084	77	72	71	93	84	305	330	322	292	325
ENSG00000001167	41	37	35	44	40	40	26	30	33	32
ENSG00000001460	24	29	30	31	23	4	5	3	6	4
ENSG00000001461	23	26	32	35	32	3	4	4	2	6
ENSG00000001497	55	48	52	59	70	34	32	30	38	50
ENSG00000001561	139	166	136	154	152	30	33	44	39	38
ENSG00000001617	136	130	112	153	156	12	5	7	5	4

G1群　　G2群

発現変動遺伝子(DEG)がないデータで2群間比較をしてみると…

- 解析 | 発現変動 | について(last modified 2013/08/29)
- 解析 | 発現変動 | 2群間 | 対応なし | について(last modified 2013/0[illegible]12)
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC (Sun_2013)([illegible]modified 2014/02/04)推奨 NEW
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | edgeR (Robinson_2010)(last modified 2014/01/30) NEW
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | SAMseq (Li_2013)(last modified 2014/01/30) NEW
- 解析 | 発現変動 | 2群間 | 対応なし | 複製なし | TCC (Sun_2013)(last modified 2014/02/04)推奨 NEW
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC (Sun_2013)

解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC (Sun_2013) NEW

TCCを用いたやり方を示します。
内部的にiDEGES/edgeR(Sun_2013)正規化を実行したのち、edgeRパッケージ中のexact testで発現変動遺伝子(Differentially expressed Genes; DEGs)検出を行っています。TCC原著論文中のiDEGES/edgeR-edgeRという解析パイプラインに相当します。全てTCCパッケージ(Sun et al., BMC Bioinformatics, 2013)内で完結します。
「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1つの10,000 genes×6 samplesのカウントデータ(data_hypodata_3vs3.txt)の場合:

Biological replic
gene_2000までが
gene_10000までが"no

```
in_f <- "data_h
out_f1 <- "hoge
out_f2 <- "hoge
param_G1 <- 3
param_G2 <- 3
param_FDR <- 0.
param_fig <- c(

#必要なパッケージ
library(TCC)

#入力ファイルの読
data <- read.ta

#前処理(TCCクラス
data.cl <- c(re
tcc <- new("TCC
```

5. サンプルデータ4の18,110 genes×10 samplesのリアルデータ(data_marioni.txt; kidney 5サンプル vs. liver 5サンプル)の最初の4サンプルを比較する場合:

「FDR閾値を満たすもの」と「fold-change閾値を満たすもの」それぞれのM-A plotを作成しています。

```
in_f <- "data_marioni.txt"              #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge5.txt"                   #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge5_FDR.png"               #出力ファイル名を指定してout_f2に格納
out_f3 <- "hoge5_FC.png"                #出力ファイル名を指定してout_f3に格納
param_G1 <- 2                           #G1群(kidney)のサンプル数を指定
param_G2 <- 2                           #G2群(liver)のサンプル数を指定
param_FDR <- 0.05                       #DEG検出時のfalse discovery rate (FDR)閾値を指
param_FC <- 2                           #fold-change閾値(param_FC倍)を指定
param_fig <- c(400, 380)                #MA-plot描画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(TCC)                            #パッケージ

#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

#前処理(サブセットの抽出)
data <- data[,1:(param_G1+param_G2)]     #最初の(param_G1+param_G2)列分のデータを抽出し

#前処理(TCCクラスオブジェクトの作成)
data.cl <- c(rep(1, param_G1), rep(2, param_G2))#G1群を1、G2群を2としたベクトルdata.cl
tcc <- new("TCC", data, data.cl)        #TCCクラスオブジェクトtccを作成
```

左から(2+2)列分のサブセットを抽出したうえでDEG同定を行っています

5. サンプルデータ4の 18,110 genes×10 samplesのリアルデータ(data_marioni.txt; kidney 5サンプル vs. liver 5サンプル)の最初の4サンプルを比較する場合:

• 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC (Sun 2013)

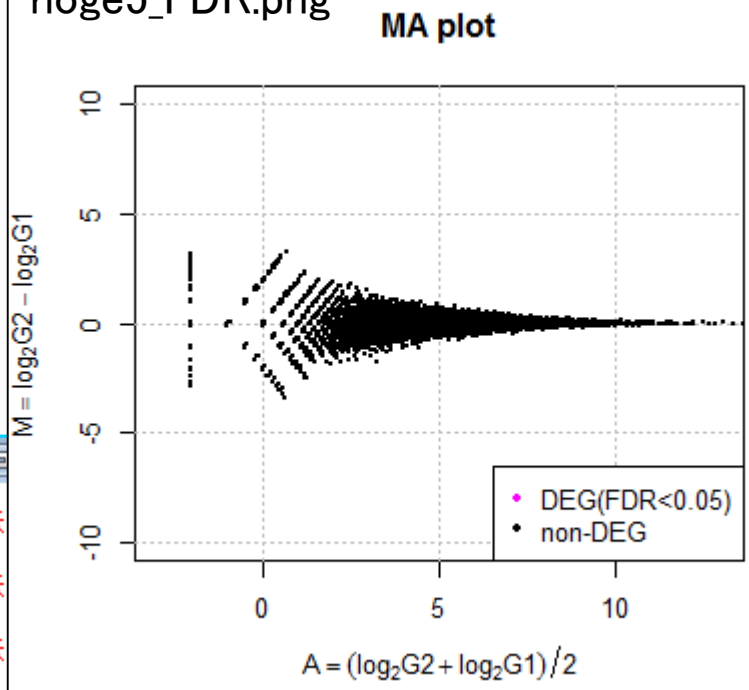
「FDR閾値を満たすもの」と「fold-change閾値を満たすもの」それぞれのM-A plotを作成しています。

```
#ファイルに保存(M-A plot; FDR)
png(out_f2, pointsize=13, width=param_fig[1], height=param_fig[2])#出力ファイルの各種
plot(tcc, FDR=param_FDR, xlim=c(-3, 13), ylim=c(-10, 10))#param_FDRで指定した閾値を満
legend("bottomright", c(paste("DEG(FDR<", param_FDR, ")", sep=""), "non-DEG")
       col=c("magenta", "black"), pch=20)#凡例を作成している
dev.off()                                  #おまじない
sum(tcc$stat$q.value < 0.05)               #FDR < 0.05を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.10)               #FDR < 0.10を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.20)               #FDR < 0.20を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.30)               #FDR < 0.30を満たす遺伝子数を表示

#ファイルに保存(M-A plot; fold-change)
param_FC <- 2                              #fold-change閾値(param_FC倍)を指定
M <- getResult(tcc)$m.value                #M-A plotのM値を抽出
hoge <- rep(1, length(M))                  #初期値を1にしたベクトルhogeを作成
hoge[abs(M) > log2(param_FC)] <- 2          #条件を満たす位置に2を代入
cols <- c("black", "magenta")              #色情報を指定してcolsに格納

png(o
plot(
legen
```

R Console

```
> sum(tcc$stat$q.value < 0.05)          #FDR < 0.05を満たす遺伝子数を表示
[1] 0
> sum(tcc$stat$q.value < 0.10)          #FDR < 0.10を満たす遺伝子数を表示
[1] 0
> sum(tcc$stat$q.value < 0.20)          #FDR < 0.20を満たす遺伝子数を表示
[1] 0
> sum(tcc$stat$q.value < 0.30)          #FDR < 0.30を満たす遺伝子数を表示
[1] 0
>
```

hoge5_FDR.png



統計的手法のFDR < 0.05を満たすDEG検出結果は0個

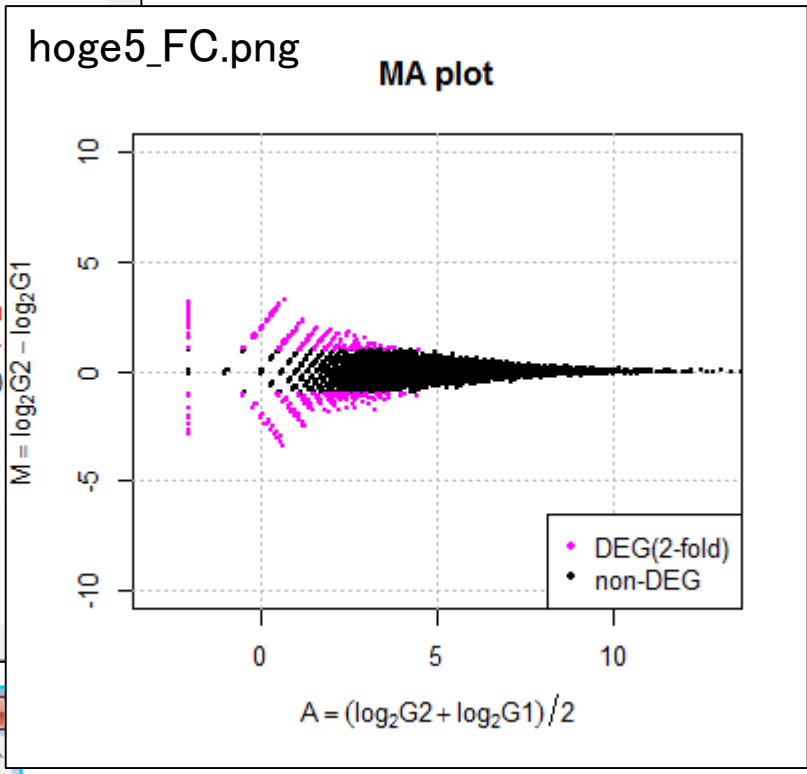
5. サンプルデータ4の18,110 genes×10 samplesのリアルデータ(data_marioni.txt; kidney 5サンプル vs. liver 5サンプル)の最初の4サンプルを比較する場合:

• 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC (Sun 2013)

「FDR閾値を満たすもの」と「fold-change閾値を満たすもの」それぞれのM-A plotを作成しています。

```
sum(tcc$stat$q.value < 0.05)          #FDR < 0.05を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.10)          #FDR < 0.10を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.20)          #FDR < 0.20を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.30)          #FDR < 0.30を満たす遺伝子数を表示

#ファイルに保存(M-A plot; fold-change)
param_FC <- 2                         #fold-change閾値(param_FC倍)を指定
M <- getResult(tcc)$m.value           #M-A plotのM値を抽出
hoge <- rep(1, length(M))             #初期値を1にしたベクトルhogeを作成
hoge[abs(M) > log2(param_FC)] <- 2     #条件を満たす位置に2を代入
cols <- c("black", "magenta")         #色情報を指定してcolsに格納

png(out_f3, pointsize=13, width=param_fig[1], height=param_fig[2])#出力ファイル
plot(tcc, col=cols, col.tag=hoge, xlim=c(-3, 13), ylim=c(-10, 10))#M-A plotを
legend("bottomright", c(paste("DEG(", param_FC, "-fold)", sep=""), "non-DEG"),
       col=c("magenta", "black"), pch=20)#凡例を作成している
dev.off()                             #おまじない
sum(abs(M) > log2(16))                #16倍以上発現変動する遺伝子数を表示
sum(abs(M) > log2(8))                 #8倍以上発現変動する遺伝子数を表示
sum(abs(M) > log2(4))                 #4倍以上発現変動する遺伝子数を表示
sum(abs(M) > log2(2))                 #2倍以上発現変動する遺伝子数を表示
```

hoge5_FC.png



R Console

```
> sum(abs(M) > log2(16))              #16倍以上発現変動する遺伝子数を$
[1] 0
> sum(abs(M) > log2(8))               #8倍以上発現変動する遺伝子数を表$
[1] 10
> sum(abs(M) > log2(4))               #4倍以上発現変動する遺伝子数を表$
[1] 214
> sum(abs(M) > log2(2))               #2倍以上発現変動する遺伝子数を表$
[1] 1402
>
```

2倍以上発現変動しているものをDEGとみなすと18,110遺伝子中1,402個も!!

Contents（Rで…）

- ゲノム解析
  - アノテーションファイルを読み込んで目的のキーワードを含む行のみ抽出
  - multi-FASTAファイルを自在に解析
    - 配列長分布、GC含量、フィルタリング、部分配列の切り出しなど
    - 連続塩基の出現頻度（CpG）解析、ゲノム配列取得など
- トランスクリプトーム解析
  - 研究目的別留意点：サンプル内とサンプル間の違い
  - マッピング → カウント情報取得
  - データを眺める：クラスタリングやM-A plot
  - 理想的な実験デザイン
  - **なぜ*x*倍発現変動という議論がだめなんですか？**
  - **モデルとか分布って、自分の解析結果にどういう影響を与えているの？**
  - 多重比較問題：FDRって何？

2群間比較：technical replicatesデータ

- data_marioni.txt (ヒトのデータ)


kidney（腎臓）


liver（肝臓）

18,110 genes

rownames(data)	R1L1Kidney	R1L3Kidney	R1L7Kidney	R2L2Kidney	R2L6Kidney	R1L2Liver	R1L4Liver	R1L6Liver	R1L8Liver	R2L3Liver
ENSG00000000003	178	167	179	172	151	138	178	175	187	169
ENSG00000000005	0	0	0	0	1	0	0	0	0	0
ENSG00000000419	53	78	64	72	71	30	42	41	33	43
ENSG00000000457	22	33	30	27	30	47	60	37	42	62
ENSG00000000460	9	7	18	14	9	19	9	13	14	19
ENSG00000000938	14	18	7	27	15	42	34	39	37	45
ENSG00000000971	28	27	28	34	42	470	502	500	490	536
ENSG00000001036	154	195	140	168	187	56	67	55	64	67
ENSG00000001084	77	72	71	93	84	305	330	322	292	325
ENSG00000001167	41	37	35	44	40	40	26	30	33	32
ENSG00000001460	24	29	30	31	23	4	5	3	6	4
ENSG00000001461	23	26	32	35	32	3	4	4	2	6
ENSG00000001497	55	48	52	59	70	34	32	30	38	50
ENSG00000001561	139	166	136	154	152	30	33	44	39	38
ENSG00000001617	136	130	112	153	156	12	5	7	5	4

G1群　　G2群

通常の2群間比較をしてみると…

- 解析 | 発現変動 | について(last modified 2013/08/29)
- 解析 | 発現変動 | 2群間 | 対応なし | について(last modified 2013/0[illegible]12)
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC (Sun 2013)([illegible]modified 2014/02/04)推奨 NEW
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | edgeR (Robinson [illegible]r0)(last modified 2014/01/30) NEW
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | SAMseq (Li 2013)(last modified 2014/01/30) NEW
- 解析 | 発現変動 | 2群間 | 対応なし | 複製なし | TCC (Sun 2013)(last modified 2014/02/04)推奨 NEW

- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC (Sun 2013)

解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC (Sun_2013) NEW

TCCを用いたやり方を示します。
内部的にiDEGES/edgeR(Sun 2013)正規化を実行したのち、edgeRパッケージ中のexact testで発現変動遺伝子(Differentially expressed Genes; DEGs)検出を行っています。TCC原著論文中のiDEGES/edgeR-edgeRという解析パイプラインに相当します。全てTCCパッケージ(Sun et al., BMC Bioinformatics, 2013)内で完結します。
「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1の10,000 genes×6 samplesのカウントデータ(data_hypodata_3vs3.txt)の場合:

Biological replic
gene_2000までか
gene_10000までか"no

```
in_f <- "data_h
out_f1 <- "hoge
out_f2 <- "hoge
param_G1 <- 3
param_G2 <- 3
param_FDR <- 0.
param_fig <- c(

#必要なパッケージ
library(TCC)

#入力ファイルの読
data <- read.ta

#前処理(TCCクラ
data.cl <- c(re
tcc <- new("TCC
```

4. サンプルデータ4の18,110 genes×10 samplesのリアルデータ(data_marioni.txt; kidney 5サンプル vs. liver 5サンプル)の場合:

「FDR閾値を満たすもの」と「fold-change閾値を満たすもの」それぞれのM-A plotを作成しています。

```
in_f <- "data_marioni.txt"              #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge4.txt"                   #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge4_FDR.png"               #出力ファイル名を指定してout_f2に格納
out_f3 <- "hoge4_FC.png"                #出力ファイル名を指定してout_f3に格納
param_G1 <- 5                           #G1群(kidney)のサンプル数を指定
param_G2 <- 5                           #G2群(liver)のサンプル数を指定
param_FDR <- 0.05                       #DEG検出時のfalse discovery rate (FDR)閾値を指
param_FC <- 2                           #fold-change閾値(param_FC倍)を指定
param_fig <- c(400, 380)                #MA-plot描画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(TCC)                            #パッケージの読み

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

#前処理(TCCクラスオブジェクトの作成)
data.cl <- c(rep(1, param_G1), rep(2, param_G2))#G1群を1、G2群を2としたベクトルdata.cl
tcc <- new("TCC", data, data.cl)        #TCCクラスオブジェクトtccを作成

#本番(iDEGES/edgeR正規化)
tcc <- calcNormFactors(tcc, norm.method="tmm", test.method="edger",#正規化を実行した結
```

サブセットの抽出を行わずにDEG同定を行っています

4. サンプルデータ4の 18,110 genes×10 samplesのリアルデータ(data_marioni.txt; kidney 5サンプル vs. liver 5サンプル)の場合:

• 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC (Sun 2013)

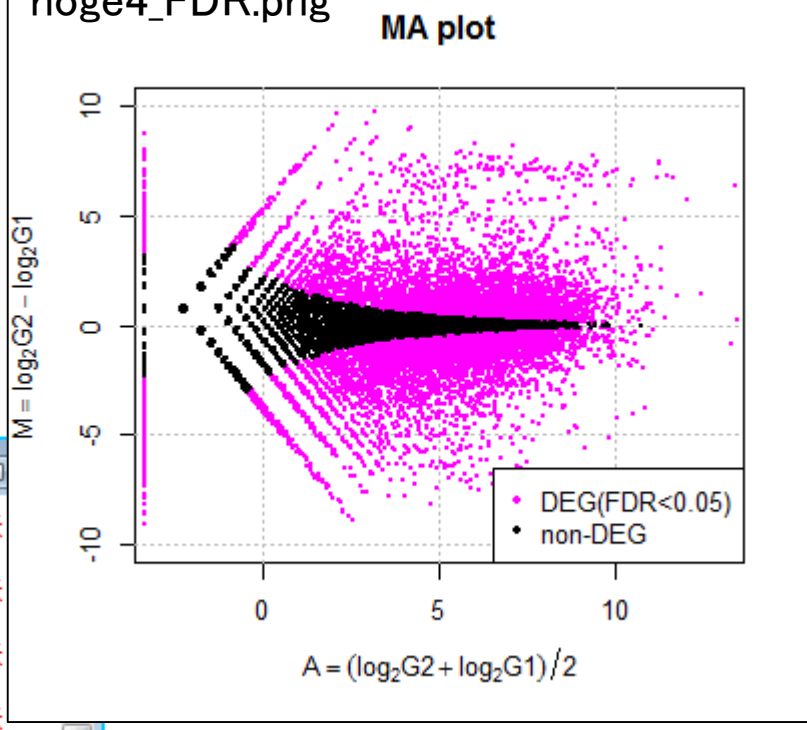
「FDR閾値を満たすもの」と「fold-change閾値を満たすもの」それぞれのM-A plotを作成しています。

```
#ファイルに保存(M-A plot; FDR)
png(out_f2, pointsize=13, width=param_fig[1], height=param_fig[2])#出力ファイル
plot(tcc, FDR=param_FDR, xlim=c(-3, 13), ylim=c(-10, 10))#param_FDRで指定した閾
legend("bottomright", c(paste("DEG(FDR<", param_FDR, ")", sep=""), "non-DEG")
       col=c("magenta", "black"), pch=20)#凡例を作成している
dev.off()                                #おまじない
sum(tcc$stat$q.value < 0.05)             #FDR < 0.05を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.10)             #FDR < 0.10を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.20)             #FDR < 0.20を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.30)             #FDR < 0.30を満たす遺伝子数を表示

#ファイルに保存(M-A plot; fold-change)
param_FC <- 2                            #fold-change閾値(param_FC倍)を指定
M <- getResult(tcc)$m.value              #M-A plotのM値を抽出
hoge <- rep(1, length(M))                #初期値を1にしたベクトルhogeを作成
hoge[abs(M) > log2(param_FC)] <- 2        #条件を満たす位置に2を代入
cols <- c("black", "magenta")            #色情報を指定してcolsに格納

png(o
plot(
legen
```

R Console

```
> sum(tcc$stat$q.value < 0.05)          #FDR < 0.05を満たす遺伝子数を表示
[1] 10831
> sum(tcc$stat$q.value < 0.10)          #FDR < 0.10を満たす遺伝子数を表示
[1] 11668
> sum(tcc$stat$q.value < 0.20)          #FDR < 0.20を満たす遺伝子数を表示
[1] 12609
> sum(tcc$stat$q.value < 0.30)          #FDR < 0.30を満たす遺伝子数を表示
[1] 13223
>
```

hoge4_FDR.png



統計的手法のFDR < 0.05を満たすDEG検出結果は全18,110個中10,831個

4. サンプルデータ4の 18,110 genes×10 samplesのリアルデータ(data_marioni.txt; kidney 5サンプル vs. liver 5サンプル)の場合:

• 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC (Sun 2013)

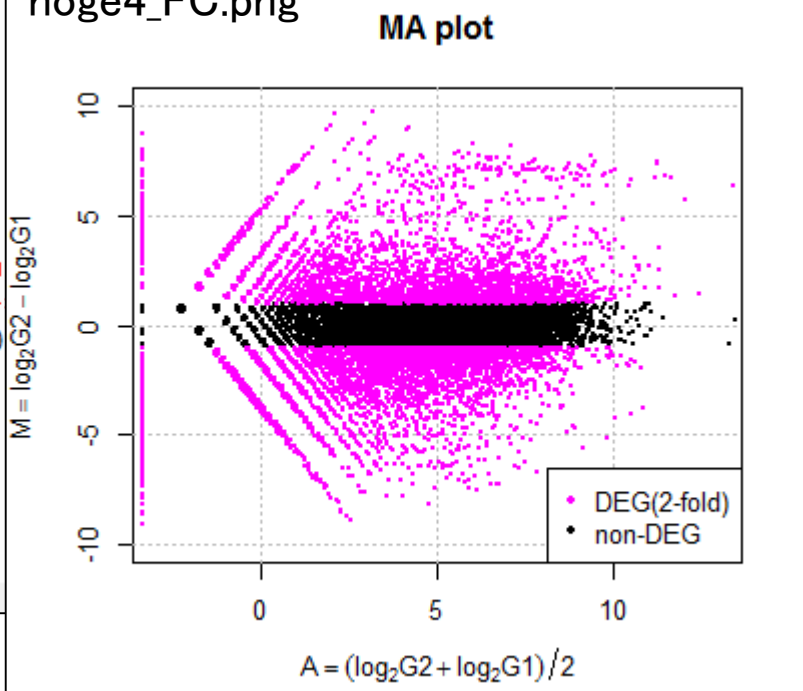
「FDR閾値を満たすもの」と「fold-change閾値を満たすもの」それぞれのM-A plotを作成しています。

```
sum(tcc$stat$q.value < 0.05)          #FDR < 0.05を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.10)          #FDR < 0.10を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.20)          #FDR < 0.20を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.30)          #FDR < 0.30を満たす遺伝子数を表示

#ファイルに保存(M-A plot; fold-change)
param_FC <- 2                         #fold-change閾値(param_FC倍)を指定
M <- getResult(tcc)$m.value           #M-A plotのM値を抽出
hoge <- rep(1, length(M))             #初期値を1にしたベクトルhogeを作成
hoge[abs(M) > log2(param_FC)] <- 2     #条件を満たす位置に2を代入
cols <- c("black", "magenta")         #色情報を指定してcolsに格納

png(out_f3, pointsize=13, width=param_fig[1], height=param_fig[2])#出力ファイル
plot(tcc, col=cols, col.tag=hoge, xlim=c(-3, 13), ylim=c(-10, 10))#M-A plotを
legend("bottomright", c(paste("DEG(", param_FC, "-fold)", sep=""), "non-DEG")
       col=c("magenta", "black"), pch=20)#凡例を作成している
dev.off()                             #おまじない
sum(abs(M) > log2(16))                #16倍以上発現変動する遺伝子数を表示
sum(abs(M) > log2(8))                 #8倍以上発現変動する遺伝子数を表示
sum(abs(M) > log2(4))                 #4倍以上発現変動する遺伝子数を表示
sum(abs(M) > log2(2))                 #2倍以上発現変動する遺伝子数を表示
```

R Console

```
> sum(abs(M) > log2(16))              #16倍以上発現変動する遺伝子数を$
[1] 1105
> sum(abs(M) > log2(8))               #8倍以上発現変動する遺伝子数を表$
[1] 2053
> sum(abs(M) > log2(4))               #4倍以上発現変動する遺伝子数を表$
[1] 3810
> sum(abs(M) > log2(2))               #2倍以上発現変動する遺伝子数を表$
[1] 7807
>
```

hoge4_FC.png



2倍以上発現変動しているものをDEGとみなすと18,110個中7,807個も!!

2群間比較：technical replicatesデータ

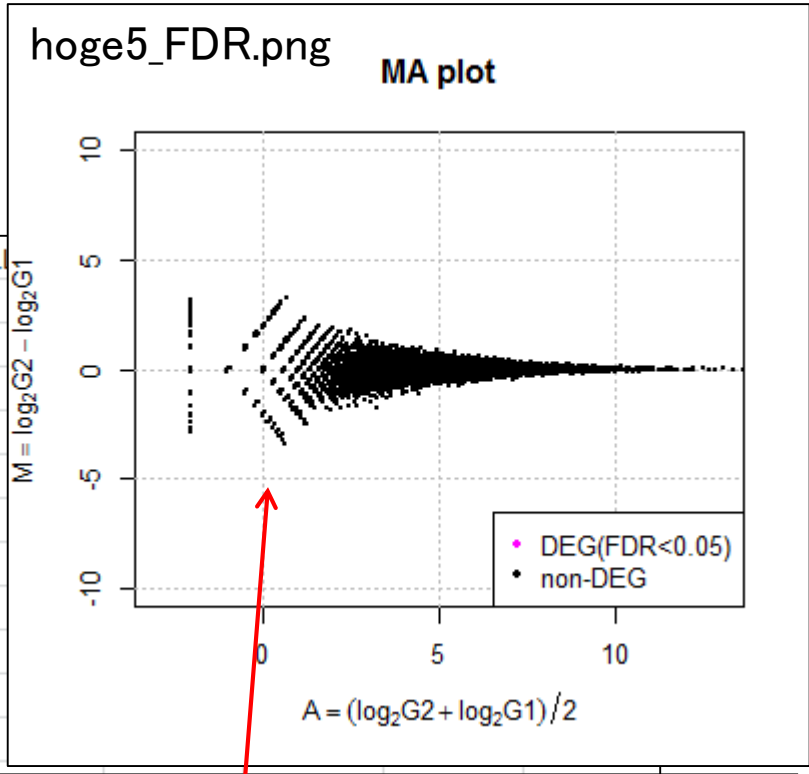
- data_marioni.txt（ヒトのデータ）

kidney（腎臓）

18,110 genes

rownames(data)	R1L1Kidney	R1L3Kidney	R1L7Kidney	R2L2Kidney	R2L6Kidney	R1I
ENSG00000000003	178	167	179	172	151	
ENSG00000000005	0	0	0	0	1	
ENSG00000000419	53	78	64	72	71	
ENSG00000000457	22	33	30	27	30	
ENSG00000000460	9	7	18	14	9	
ENSG00000000938	14	18	7	27	15	
ENSG00000000971	28	27	28	34	42	
ENSG00000001036	154	195	140	168	187	
ENSG00000001084	77	72	71	93	84	
ENSG00000001167	41	37	35	44	40	
ENSG00000001460	24	29	30	31	23	
ENSG00000001461	23	26	32			
ENSG00000001497	55	48	52			
ENSG00000001561	139	166	136			
ENSG00000001617	136	130	112	153	156	12

G1群　　G2群



この分布は同一群内のばらつきの程度を表している
→ この分布のど真ん中に位置する遺伝子のp値 = 1

2群間比較：technical replicatesデータ

- data_marioni.txt（ヒトのデータ）

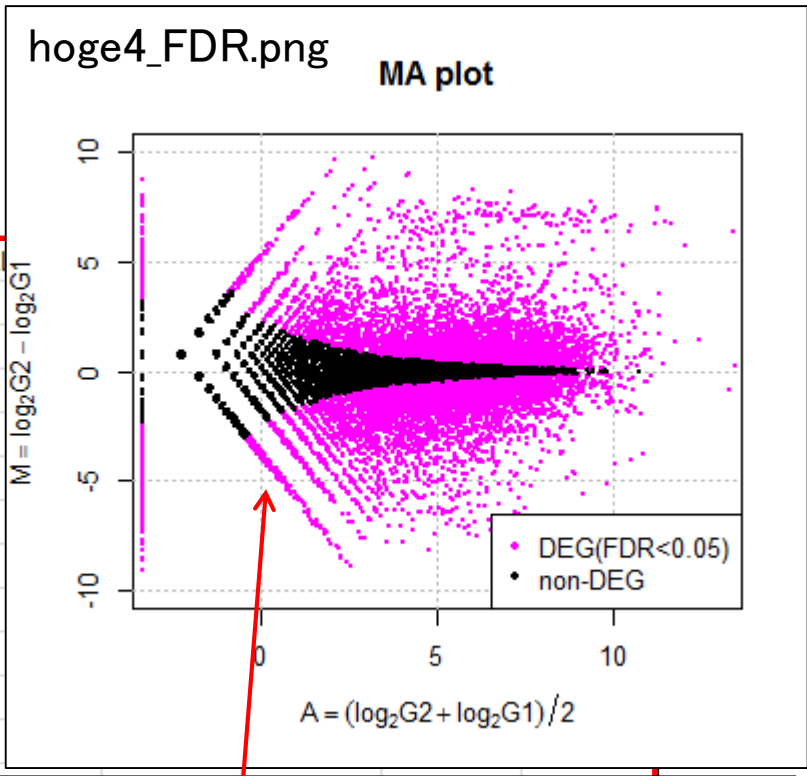
kidney（腎臓）

18,110 genes

rownames(data)	R1L1Kidney	R1L3Kidney	R1L7Kidney	R2L2Kidney	R2L6Kidney	R1I
ENSG00000000003	178	167	179	172	151	
ENSG00000000005	0	0	0	0	1	
ENSG00000000419	53	78	64	72	71	
ENSG00000000457	22	33	30	27	30	
ENSG00000000460	9	7	18	14	9	
ENSG00000000938	14	18	7	27	15	
ENSG00000000971	28	27	28	34	42	
ENSG00000001036	154	195	140	168	187	
ENSG00000001084	77	72	71	93	84	
ENSG00000001167	41	37	35	44	40	
ENSG00000001460	24	29	30	31	23	
ENSG00000001461	23	26	32			
ENSG00000001497	55	48	52			
ENSG00000001561	139	166	136			
ENSG00000001617	136	130	112	153	156	12 5 7 5 4

G1群　　G2群

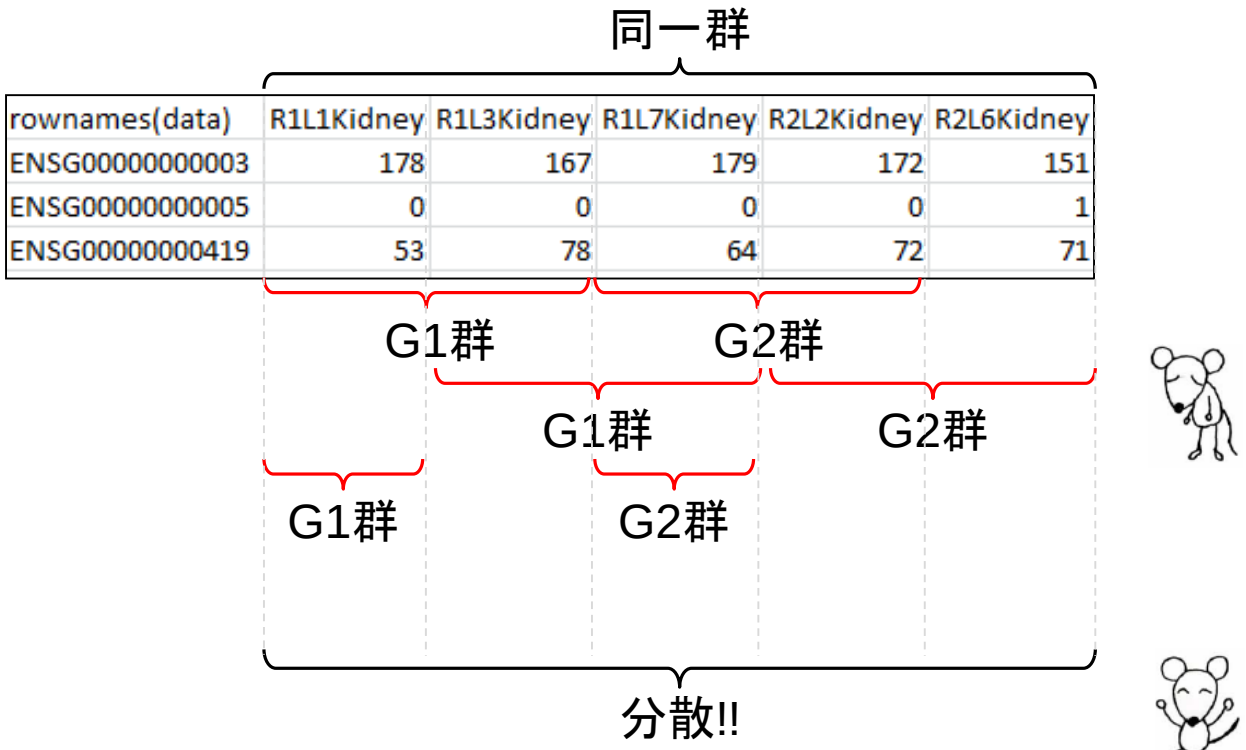
hoge4_FDR.png



同一群内のばらつきの範囲内は正しくnon-DEG、それ以外の位置がDEGと判定されていることがわかる

モデル構築

- 同一群に属する反復実験データのばらつきの程度を把握すること
- 平均−分散（MEAN−VARIANCE）プロットがよく用いられる
  - M−A plotの場合は、縦軸のM値（log比）がばらつきの指標に相当するが、無数の組合せが存在する



ばらつきの程度をより直接的に表現する指標は**分散**

平均-分散プロット

- 解析 | 一般 | Sequence logos(Schneider 1990)(last modified 2012/06/27)
- 解析 | 一般 | 上流配列解析 | LDSS(Yamamoto 2007)(last modified 2012/07/17)
- 解析 | 一般 | 上流配列解析 | Relative Appearance Rat (Yamamoto 2011)(last modified 2
- 解析 | 基礎 | 平均-分散プロット(Technical replicates) modified 2013/12/27)
- 解析 | 基礎 | 平均-分散プロット(Biological replicates)(st modified 2013/12/27)
- 解析 | クラスタリング | について(last modified 2014/02/05) NEW
- 解析 | クラスタリ
- 解析 | クラスタリ
- 解析 | 発現変動
- 解析 | 発現変動
- 解析 | 発現変動
- 解析 | 発現変動
- 解析 | 発現変動

解析 | 基礎 | 平均-分散プロット(Technical replicates)

Marioniらは Technical replicatesのデータがポアソン分布(Poisson distribution)に従うことを報告しています(Marioni et al., Genome Res., 2008)。つまり「各遺伝子の technical replicatesデータの平均と分散が全体として同じ」だと言っているわけです。
ここでは、サンプルデータ4の 18,110 genes×10 samplesの technical replicatesのカウントデータ(data_marioni.txt; kidney 5サンプル vs. liver 5サンプル)を読み込んで平均-分散プロットを作成します。
「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4の 18,110 genes×10 samplesのリアルデータ(data_marioni.txt; kidney 5サンプル vs. liver 5サンプル)の場合:

```
in_f <- "data_marioni.txt"              #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge1_G1.txt"                #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge1_G1.png"                #出力ファイル名を指定してout_f2に格納
out_f3 <- "hoge1_G2.txt"                #出力ファイル名を指定してout_f3に格納
out_f4 <- "hoge1_G2.png"                #出力ファイル名を指定してout_f4に格納
out_f5 <- "hoge1_all.png"               #出力ファイル名を指定してout_f5に格納
param_G1 <- 5                           #G1群(kidney)のサンプル数を指定
param_G2 <- 5                           #G2群(liver)のサンプル数を指定
param_fig <- c(380, 420)                #ファイル出力時の横幅と縦幅を指定(単位はピクセル)

#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で指定したファイルの読
colSums(data)                           #総リード数を表示

#前処理(データ正規化; 群ごとに総リード数の平均値を揃えている)
data.cl <- c(rep(1, param_G1), rep(2, param_G2))#G1群を1、G2群を2としたベクトルdata.clを作成
hoge <- data[,data.cl==1]               #G1群のデータのみ抽出している
nf <- mean(colSums(hoge))/colSums(hoge)#G1群の正規化係数を計算した結果をnfに格納
```

実行してみましょう

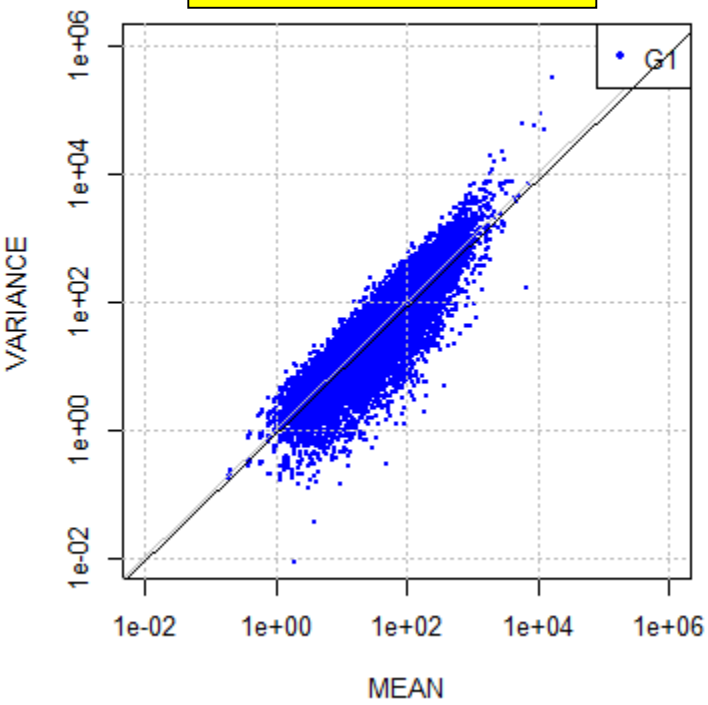
解析 | 基礎 | 平均-分散プロット(Technical replicates)

Marioniらは Technical replicatesのデータがポアソン分布(Poisson distribution)に従うことを報告しています(
Res., 2008)。つまり「各遺伝子のtechnical replicatesデータの平均と分散が全体として同じ」だと言っているわ
ここでは、サンプルデータ4の18,110 genes×10 samplesのtechnical replicatesのカウントデータ(data_marioni.tx
vs. liver 5サンプル)を読み込んで平均-分散プロットを作成します。
「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4の18,110 genes×10 samplesのリアルデータ(data_marioni.txt; kidney 5サンプル vs. liv

```
in_f <- "data_marioni.txt"          #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge1_G1.txt"            #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge1_G1.png"            #出力ファイル名を指定してout_f2に格納
out_f3 <- "hoge1_G2.txt"            #出力ファイル名を指定してout_f3に格納
out_f4 <- "hoge1_G2.png"            #出力ファイル名を指定してout_f4に格納
out_f5 <- "hoge1_all.png"           #出力ファイル名を指定してout_f5に格納
param_G1 <- 5                       #G1群(kidney)のサンプル数を指定
param_G2 <- 5                       #G2群(liver)のサンプル数を指定
param_fig <- c(380, 420)            #ファイル出力時の横幅と縦幅を指定(単位はピク
```

$y = x$の直線上にプロットされている



入力データ(G1群)

rownames(data)	R1 L1 Kidney	R1 L3 Kidney	R1 L7 Kidney	R2 L2 Kidney	R2 L6 Kidney
ENSG00000000003	178	167	179	172	151
ENSG00000000005	0	0	0	0	1
ENSG00000000419	53	78	64	72	71
ENSG00000000457	22	33	30	27	30
ENSG00000000460	9	7	18	14	9
ENSG00000000938	14	18	7	27	15
ENSG00000000971	28	27	28	34	42
...	...	...	...	...	...
sum	1574608	1622037	1525718	1677983	1711483

総リード数補正後のデータ

R1 L1 Kidney	R1 L3 Kidney	R1 L7 Kidney	R2 L2 Kidney	R2 L6 Kidney
183.4	167.0	190.3	166.3	143.1
0.0	0.0	0.0	0.0	0.9
54.6	78.0	68.1	69.6	67.3
22.7	33.0	31.9	26.1	28.4
9.3	7.0	19.1	13.5	8.5
14.4	18.0	7.4	26.1	14.2
28.8	27.0	29.8	32.9	39.8
...	...	...	...	...
1622366	1622366	1622366	1622366	1622366

MEAN	VARIANCE
170.0	334.3
0.2	0.2
67.5	70.4
28.4	17.9
11.5	24.1
16.0	46.2
31.7	25.3
...	...

解析 | 基礎 | 平均-分散プロット(Technical replicates)

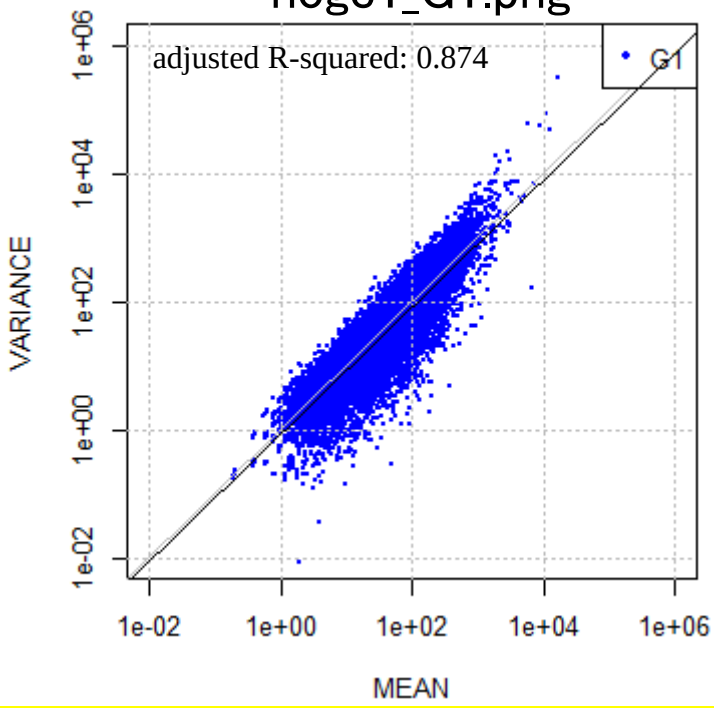
Marioniらは Technical replicatesのデータがポアソン分布(Poisson distribution)に従うことを報告しています(Marioni et al., Genome Res., 2008)。つまり「各遺伝子のtechnical replicatesデータの平均と分散が全体として同じ」だと言っているわけです。
ここでは、サンプルデータ4の18,110 genes×10 samplesのtechnical replicatesのカウントデータ(data_marioni.txt; kidney 5サンプル vs. liver 5サンプル)を読み込んで平均-分散プロットを作成します。
「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4の18,110 genes×10 samplesのリアルデータ(data_marioni.txt; kidney 5サンプル

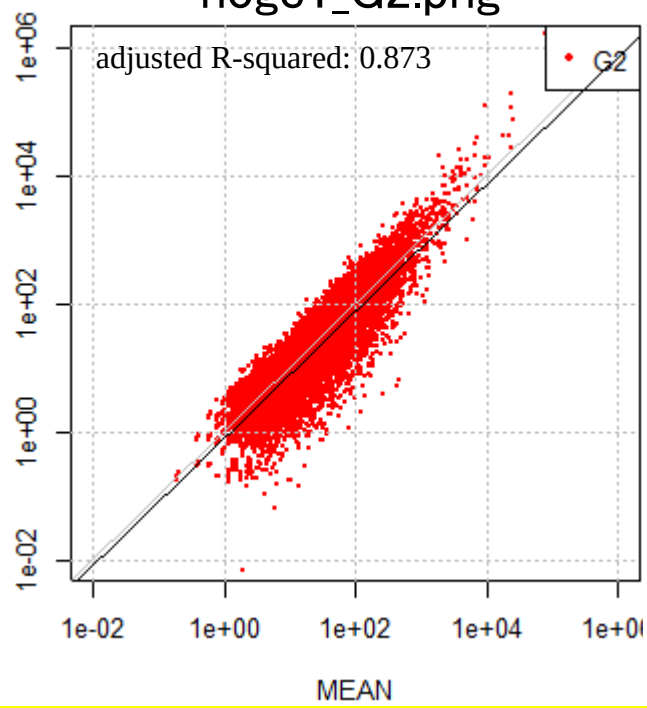
```
in_f <- "data_marioni.txt"          #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge1_G1.txt"            #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge1_G1.png"            #出力ファイル名を指定してout_f2に格納
out_f3 <- "hoge1_G2.txt"            #出力ファイル名を指定してout_f3に格納
out_f4 <- "hoge1_G2.png"            #出力ファイル名を指定してout_f4に格納
out_f5 <- "hoge1_all.png"           #出力ファイル名を指定してout_f5に格納
param_G1 <- 5                       #G1群(kidney)のサンプル数を指定
```

Technical replicatesのデータは:
・VARIANCEはそのMEANで説明可能である(VARIANCE ≒ MEAN)
・ポアソン分布に従う
・ポアソンモデルが適用可能

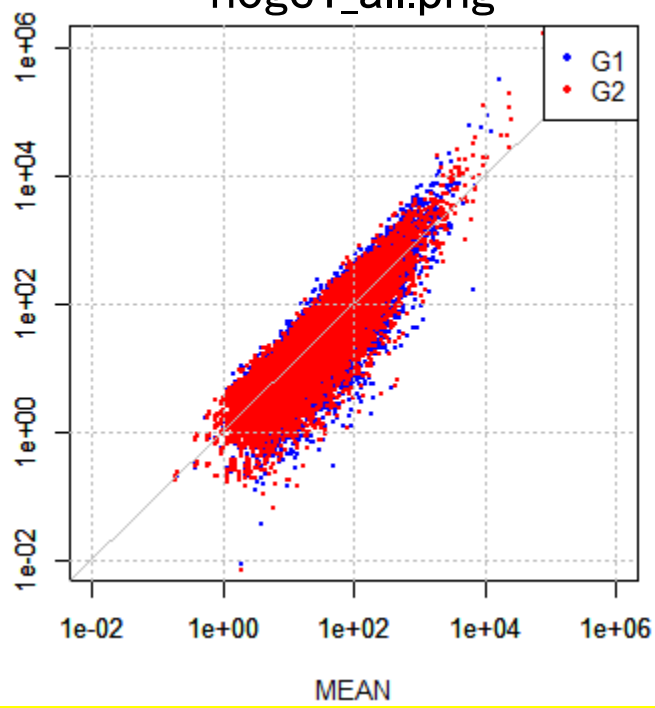
hoge1_G1.png



hoge1_G2.png



hoge1_all.png

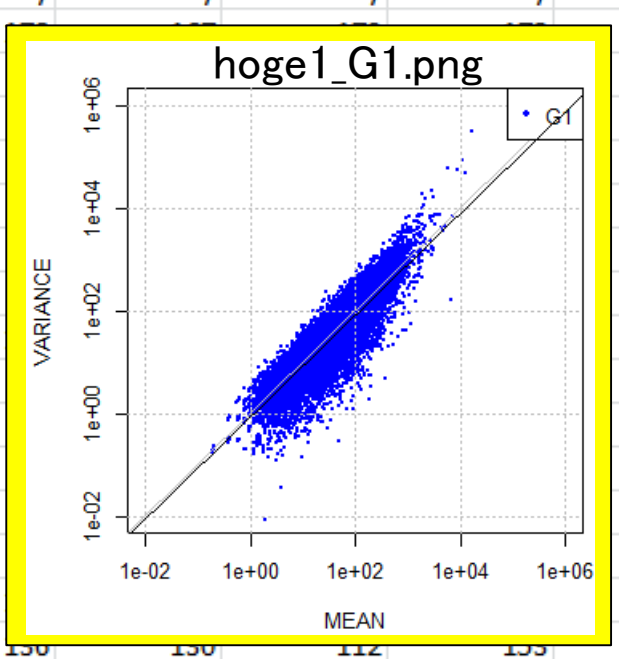


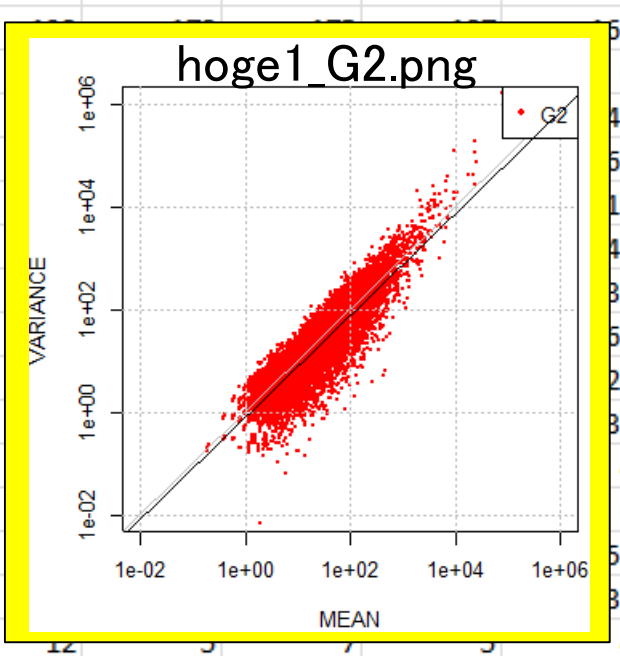
平均-分散プロット：non-DEG分布を把握

kidney（腎臓）　liver（肝臓）

18,110 genes

rownames(data)	R1L1Kidney	R1L3Kidney	R1L7Kidney	R2L2Kidney	R2L6Kidney	R1L2Liver	R1L4Liver	R1L6Liver	R1L8Liver	R2L3Liver
ENSG00000000003					151					69
ENSG00000000005					1					0
ENSG00000000419					71					43
ENSG00000000457					30					52
ENSG00000000460					9					19
ENSG00000000938					15					45
ENSG00000000971					42					86
ENSG00000001036					187					67
ENSG00000001084					84					25
ENSG00000001167					40					32
ENSG00000001460					23					4
ENSG00000001461					32					6
ENSG00000001497					70					50
ENSG00000001561					152					38
ENSG00000001617	136	130	112	153	156	12	5	7	5	4





G1群　G2群

（G1群 + G2群）の平均-分散プロットを眺めると...

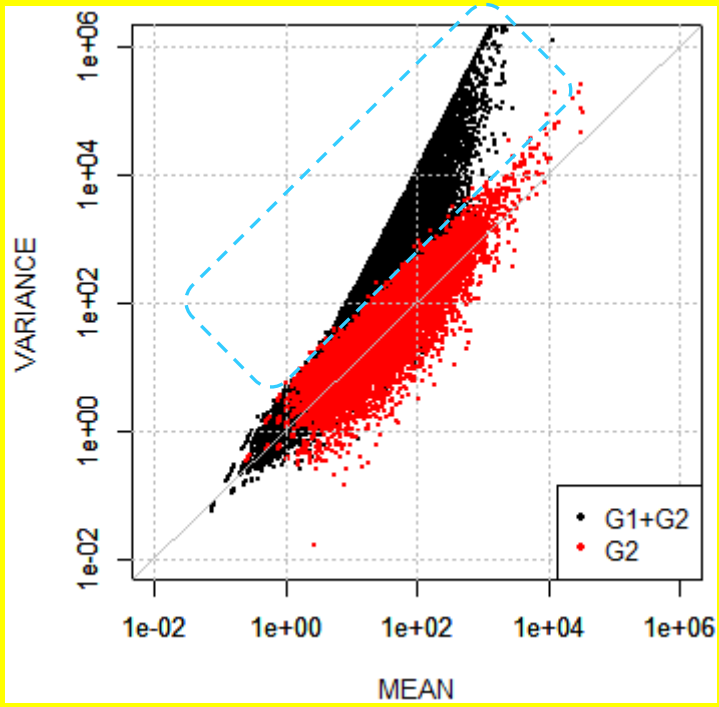
平均-分散プロット

4. サンプルデータ4の18,110 genes×10 samplesのリアルデータ(data_marioni.txt; kidney 5サンプル vs. liver 5サンプル)の場合:

TCCパッケージ中のiDEGES/edgeR正規化後のデータを用いて、2つの群をまとめてプロットしています。また、G2群のみのプロットも重ね書きしています。

```
in_f <- "data_marioni.txt"                #入力ファイル名を指定してin_fに格納
out_f <- "hoge4.png"                      #出力ファイル名を指定してout_fに格納
param_G1 <- 5                             #G1群(kidney)のサンプル数を指定
param_G2 <- 5                             #G2群(liver)のサンプル数を指定
param_fig <- c(380, 420)                  #ファイル出力時の横幅と縦幅を指定(単位は

#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

#前処理(TCCクラスオブジェクトの作成)
data.cl <- c(rep(1, param_G1), rep(2, param_G2))#G1群を1、G2群を2としたベクト
tcc <- new("TCC", data, data.cl)          #TCCクラスオブジェクトtccを作成

#本番(iDEGES/edgeR正規化)
tcc <- calcNormFactors(tcc, norm.method="tmm", test.method="edger",#正規化を
                       iteration=3, FDR=0.1, floorPDEG=0.05)#正規化を実行した
normalized.count <- getNormalizedData(tcc)#正規化後のデータを取り出してnormali

#ファイルに保存(Mean-Variance plot)
```

hoge4.png



同一群内のばらつきの範囲(non-DEG分布)外に多数の遺伝子が存在

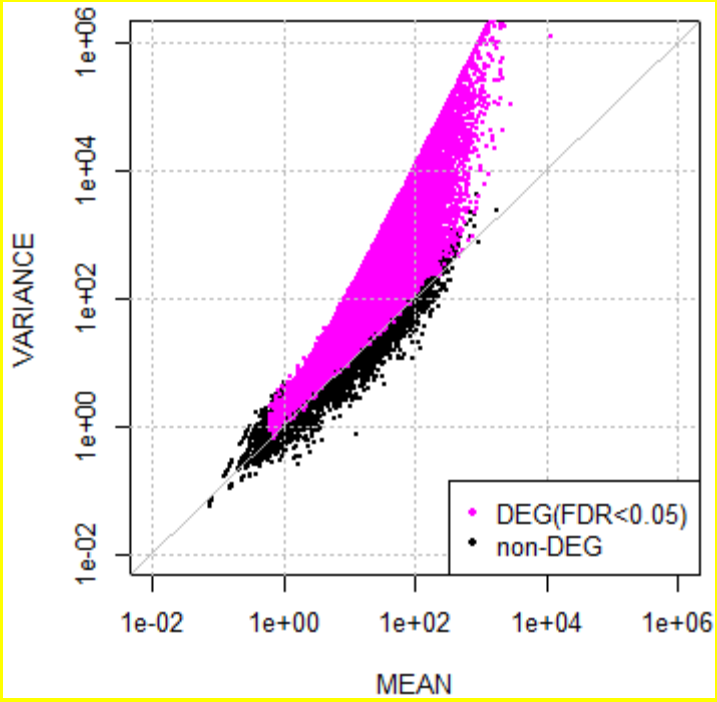
平均-分散プロット

5. サンプルデータ4の 18,110 genes×10 samplesのリアルデータ(data_marioni.txt; kidney 5サンプル vs. liver 5サンプル)の場合:

TCCパッケージ中のiDEGES/edgeR正規化後のデータを用いて、2つの群をまとめてプロットしています。
iDEGES/edgeR-edgeR解析パイプライン適用後のFDR＜0.05を満たす遺伝子をマゼンタで色づけしています。

```
in_f <- "data_marioni.txt"                #入力ファイル名を指定してin_fに格納
out_f <- "hoge5.png"                      #出力ファイル名を指定してout_fに格納
param_G1 <- 5                             #G1群(kidney)のサンプル数を指定
param_G2 <- 5                             #G2群(liver)のサンプル数を指定
param_FDR <- 0.05                         #DEG検出時のfalse discovery rate (FDR)
param_fig <- c(380, 420)                  #ファイル出力時の横幅と縦幅を指定(単位は

#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

#前処理(TCCクラスオブジェクトの作成)
data.cl <- c(rep(1, param_G1), rep(2, param_G2))#G1群を1、G2群を2としたベクト
tcc <- new("TCC", data, data.cl)          #TCCクラスオブジェクトtccを作成

#本番(iDEGES/edgeR正規化)
tcc <- calcNormFactors(tcc, norm.method="tmm", test.method="edger",#正規化を
                       iteration=3, FDR=0.1, floorPDEG=0.05)#正規化を実行した
normalized.count <- getNormalizedData(tcc)#正規化後のデータを取り出してnormaliz
```

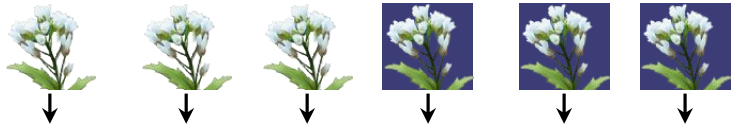
hoge5.png



一般にDEGはnon-DEGに比べ（G1群 + G2群）の分散が大きいので妥当

2群間比較：biological replicatesデータ

- data_arab.txt（植物のデータ）



identifier	mock1	mock2	mock3	hrcc1	hrcc2	hrcc3
AT1G01010	35	77	40	46	64	60
AT1G01020	43	45	32	43	39	49
AT1G01030	16	24	26	27	35	20
AT1G01040	72	43	64	66	25	90
AT1G01050	49	78	90	67	45	60
AT1G01060	0	15	2	0	21	8
AT1G01070	16	34	6	9	20	1
AT1G01080	170	191	382	127	98	184
AT1G01090	291	346	563	171	116	453
AT1G01100	113	125	246	78	27	361
AT1G01110	0	1	1	0	0	0

26,221 genes

G1群　　G2群

オリジナルは**AT4G32850**が重複して存在していたため、19,520行目のデータを予め除去している

発現変動遺伝子(DEG)がないデータで2群間比較をしてみると…

- 解析 | 発現変動 | について(last modified 2013/08/29)
- 解析 | 発現変動 | 2群間 | 対応なし | について(last modified 2013/09/12)
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC (Sun 2013)(last modified 2014/02/04)推奨 NEW
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | edgeR (Robinson 2010)(last modified 2014/01/30) NEW
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | SAMseq (Li 2013)(last modified 2014/01/30) NEW
- 解析 | 発現変動 | 2群間 | 対応なし | 複製なし | TCC (Sun 2013)(last modified 2014/02/20)推奨 NEW
- 解析 | 発現変動 | 2群間 | 対応なし | 複製なし | DESeq (Anders 2010)(last modified 2014/01/30) NEW
- 解析 | 発現変動 | 2群間 | 対応なし | BitSeq (Glaus 2012) (last modified 2013/01/08)

- 解析 | 発現変動 | 2群間 | 対応なし | 複製なし | TCC (Sun 2013)

解析 | 発現変動 | 2群間 | 対応なし | 複製なし | TCC (Sun_2013) NEW

TCCを用いたやり方を示します。
内部的にiDEGES/DESeq(Sun 2013)正規化を実行したのち、DESeqパッケージ中のnegative binomial testで発現変動遺伝子(Differentially expressed Genes; DEGs)検出を行っています。TCC原著論文中のiDEGES/DESeq-DESeqという解析パイプラインに相当します。全てTCCパッケージ内で完結します。
「ファイルと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。

3. サンプルデータ8の26,221 genes×6 samplesのリアルデータ(data_arab.txt; mock 3サンプル vs. hrcc 3サンプル)の場合:

mock群の1番目(mock1)と3番目(mock3)のサブセットを抽出して複製なしデータとして取り扱っています。
「FDR閾値を満たすもの」と「fold-change閾値を満たすもの」それぞれのM-A plotを作成しています。

```
in_f <- "data_arab.txt"                    #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge3.txt"                      #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge3_FDR.png"                  #出力ファイル名を指定してout_f2に格納
out_f3 <- "hoge3_FC.png"                   #出力ファイル名を指定してout_f3に格納
param_subset <- c(1, 3)                    #取り扱いたいサブセット情報を指定
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_FDR <- 0.05                          #DEG検出時のfalse discovery rate (FDR)閾値を指定
param_FC <- 2                              #fold change閾値(param_FC倍)を指定
param_fig <- c(400, 380)                   #MA-plot描画時の横幅と縦幅

#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で指定したファイル

#前処理(サブセットの抽出とTCCクラスオブジェクトの作成)
data <-data[,param_subset]                 #param_subsetで指定した列の情報のみ抽出
data.cl <- c(rep(1, param_G1), rep(2, param_G2))#G1群を1、G2群を2としたベクトルdata.clを作成
tcc <- new("TCC", data, data.cl)           #TCCクラスオブジェクトtccを作成
```

計6列からなるデータから(1, 3)列のサブセットを抽出したうえでDEG同定を行っています

3. サンプルデータ8の 26,221 genes×6 samplesのリアルデータ(data_arab.txt; mock 3サンプル vs. hrcc 3サンプル)の場合:

解析 | 発現変動 | 2群間 | 対応なし | 複製なし | TCC (Sun 2013)

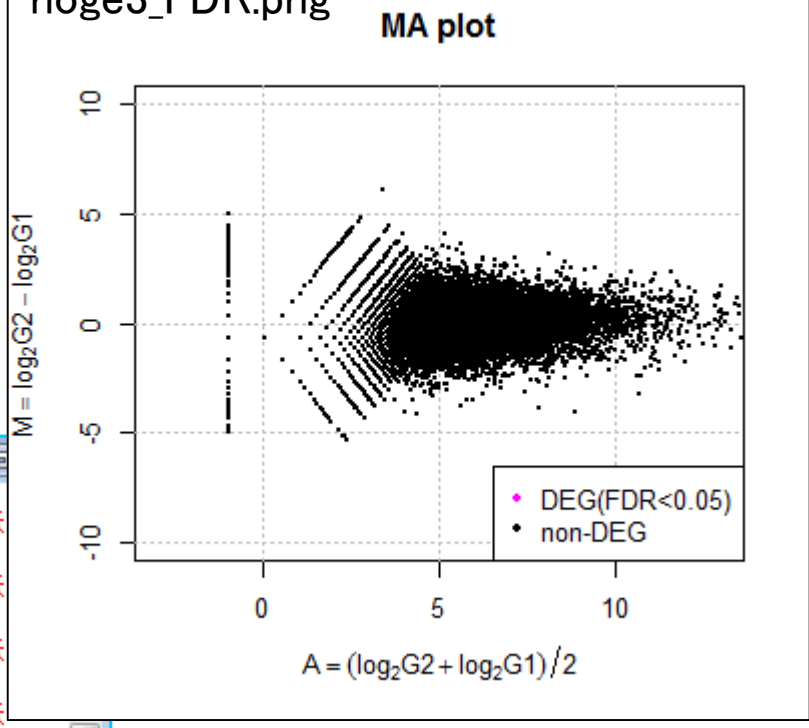
mock群の1番目(mock1)と3番目(mock3)のサブセットを抽出して複製なしデータとして取り扱っています。
「FDR閾値を満たすもの」と「fold-change閾値を満たすもの」それぞれのM-A plotを作成しています。

```
#ファイルに保存(M-A plot; FDR)
png(out_f2, pointsize=13, width=param_fig[1], height=param_fig[2])#出力ファイルの各種パラメータ
plot(tcc, FDR=param_FDR, xlim=c(-3, 13), ylim=c(-10, 10))#param_FDRで指定した閾値を満たすDEGを
legend("bottomright", c(paste("DEG(FDR<", param_FDR, ")", sep=""), "non-DEG")
       col=c("magenta", "black"), pch=20)#凡例を作成している
dev.off()                                  #おまじない
sum(tcc$stat$q.value < 0.05)               #FDR < 0.05を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.10)               #FDR < 0.10を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.20)               #FDR < 0.20を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.30)               #FDR < 0.30を満たす遺伝子数を表示

#ファイルに保存(M-A plot; fold-change)
param_FC <- 2                              #fold-change閾値(param_FC倍)を指定
M <- getResult(tcc)$m.value                #M-A plotのM値を抽出
hoge <- rep(1, length(M))                  #初期値を1にしたベクトルhogeを作成
hoge[abs(M) > log2(param_FC)] <- 2          #条件を満たす位置に2を代入
cols <- c("black", "magenta")              #色情報を指定してcolsに格納

png(o
plot(
legen
```

R Console

```
> sum(tcc$stat$q.value < 0.05)          #FDR < 0.05を満たす遺伝子数を表示
[1] 0
> sum(tcc$stat$q.value < 0.10)          #FDR < 0.10を満たす遺伝子数を表示
[1] 0
> sum(tcc$stat$q.value < 0.20)          #FDR < 0.20を満たす遺伝子数を表示
[1] 0
> sum(tcc$stat$q.value < 0.30)          #FDR < 0.30を満たす遺伝子数を表示
[1] 0
>
```

hoge3_FDR.png



統計的手法のFDR < 0.05を満たすDEG検出結果は全26,221個中0個

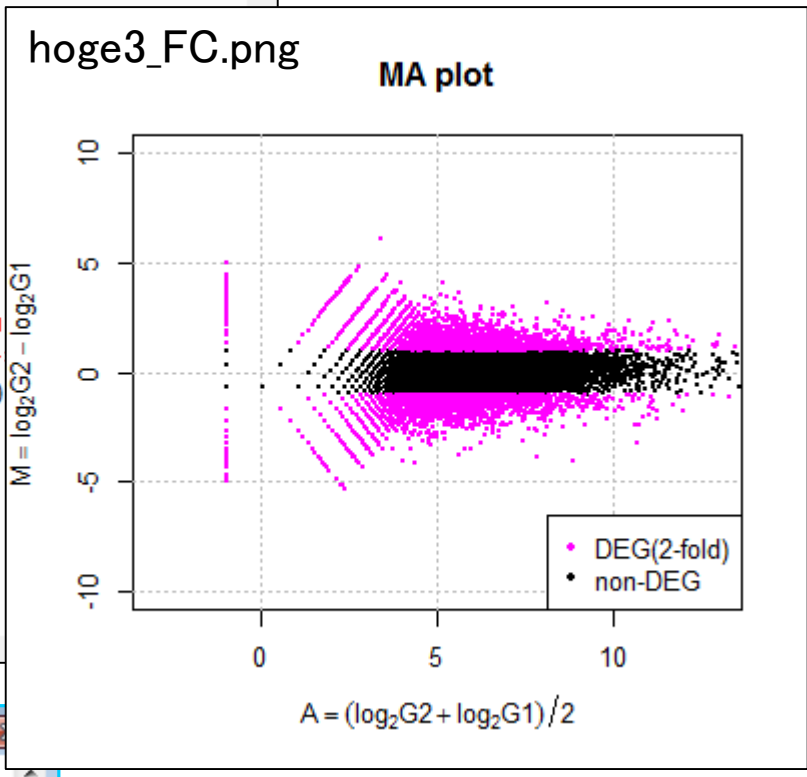
3. サンプルデータ8の 26,221 genes×6 samplesのリアルデータ(data_arab.txt; mock 3サンプル vs. hrcc 3サンプル)の場合:

• 解析 | 発現変動 | 2群間 | 対応なし | 複製なし | TCC (Sun 2013)

mock群の1番目(mock1)と3番目(mock3)のサブセットを抽出して複製なしデータとして取り扱っています。
「FDR閾値を満たすもの」と「fold-change閾値を満たすもの」それぞれのM-A plotを作成しています。

```
sum(tcc$stat$q.value < 0.05)          #FDR < 0.05を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.10)          #FDR < 0.10を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.20)          #FDR < 0.20を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.30)          #FDR < 0.30を満たす遺伝子数を表示

#ファイルに保存(M-A plot; fold-change)
param_FC <- 2                         #fold-change閾値(param_FC倍)を指定
M <- getResult(tcc)$m.value           #M-A plotのM値を抽出
hoge <- rep(1, length(M))             #初期値を1にしたベクトルhogeを作成
hoge[abs(M) > log2(param_FC)] <- 2     #条件を満たす位置に2を代入
cols <- c("black", "magenta")         #色情報を指定してcolsに格納

png(out_f3, pointsize=13, width=param_fig[1], height=param_fig[2])#出力ファイル
plot(tcc, col=cols, col.tag=hoge, xlim=c(-3, 13), ylim=c(-10, 10))#M-A plotを
legend("bottomright", c(paste("DEG(", param_FC, "-fold)", sep=""), "non-DEG")
       col=c("magenta", "black"), pch=20)#凡例を作成している
dev.off()                             #おまじない
sum(abs(M) > log2(16))                #16倍以上発現変動する遺伝子数を表示
sum(abs(M) > log2(8))                 #8倍以上発現変動する遺伝子数を表示
sum(abs(M) > log2(4))                 #4倍以上発現変動する遺伝子数を表示
sum(abs(M) > log2(2))                 #2倍以上発現変動する遺伝子数を表示
```

hoge3_FC.png



R Console

```
> sum(abs(M) > log2(16))          #16倍以上発現変動する遺伝子数を表示
[1] 66
> sum(abs(M) > log2(8))           #8倍以上発現変動する遺伝子数を表示
[1] 475
> sum(abs(M) > log2(4))           #4倍以上発現変動する遺伝子数を表示
[1] 1889
> sum(abs(M) > log2(2))           #2倍以上発現変動する遺伝子数を表示
[1] 6641
>
```

2倍以上発現変動しているものをDEGとみなすと26,221遺伝子中6,641個も!!

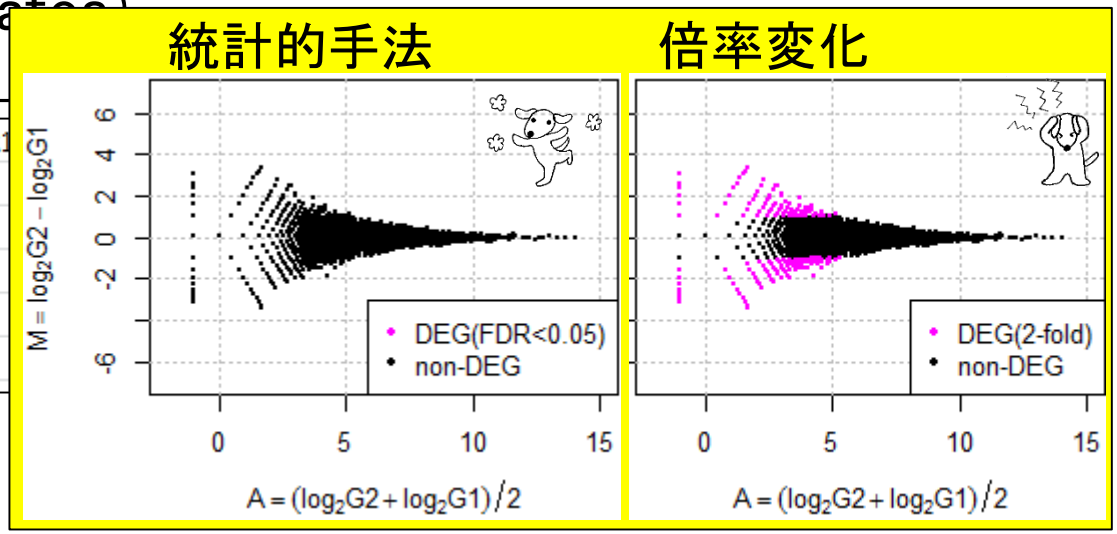
性能評価：統計的手法 vs. 倍率変化

- data_marioni.txt (technical replicates)

腎臓(Kidney)群

rownames(data)	R1L1Kidney	R1L3Kidney	R1L7Kidney	R2L2Kidney	R2L6Kidney	R1
ENSG00000000003	178	167	179	172	151	
ENSG00000000005	0	0	0	0	1	
ENSG00000000419	53	78	64	72	71	
ENSG00000000457	22	33	30	27	30	
ENSG00000000460	9	7	18	14	9	

G1群　　G2群

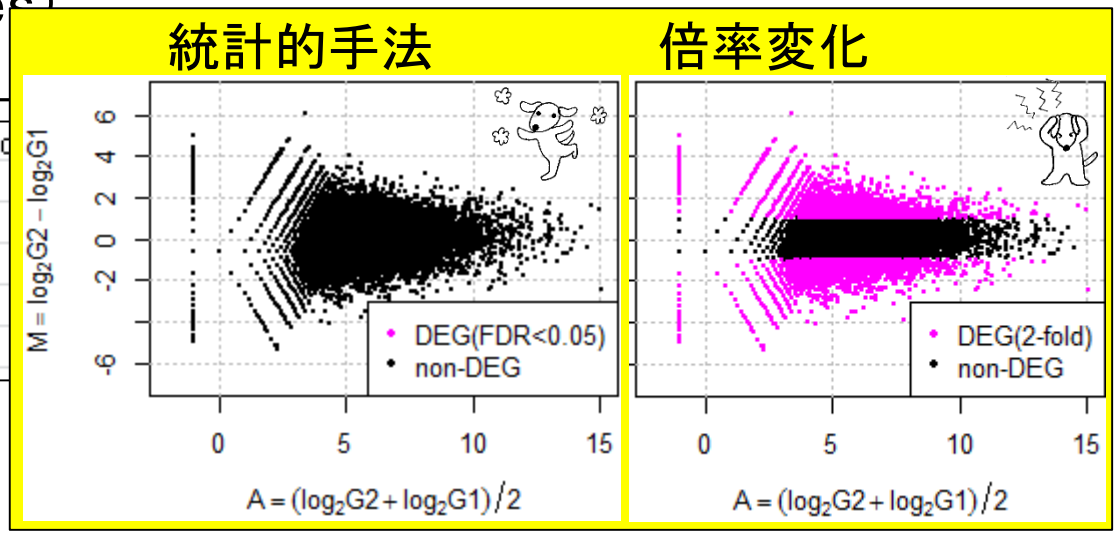


- data_arab.txt (biological replicates)

mock群　　hrcc群

identifier	mock1	mock2	mock3	hrcc1	hrcc2	hrcc
AT1G01010	35	77	40	46	64	
AT1G01020	43	45	32	43	39	
AT1G01030	16	24	26	27	35	
AT1G01040	72	43	64	66	25	
AT1G01050	49	78	90	67	45	

G1群　　G2群



統計的手法のほうがnon-DEGをDEGと判定するミス（false positives）が圧倒的に少ない

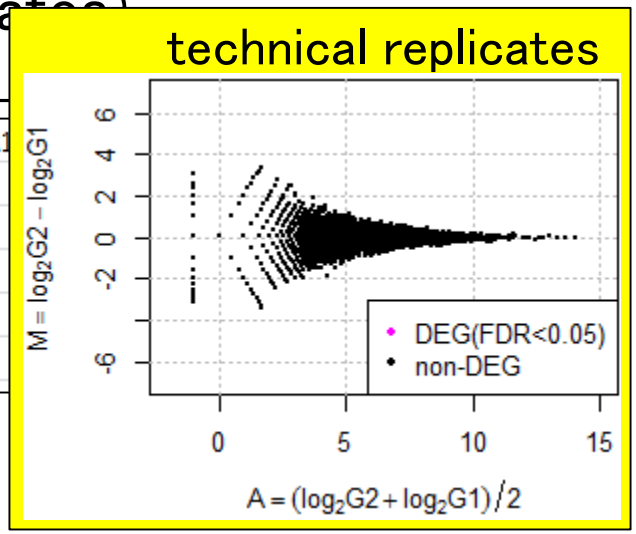
ばらつき度：technical vs. biological

- data_marioni.txt (technical replicates)

腎臓(Kidney)群

rownames(data)	R1L1Kidney	R1L3Kidney	R1L7Kidney	R2L2Kidney	R2L6Kidney	R1
ENSG00000000003	178	167	179	172	151	
ENSG00000000005	0	0	0	0	1	
ENSG00000000419	53	78	64	72	71	
ENSG00000000457	22	33	30	27	30	
ENSG00000000460	9	7	18	14	9	

G1群　　G2群

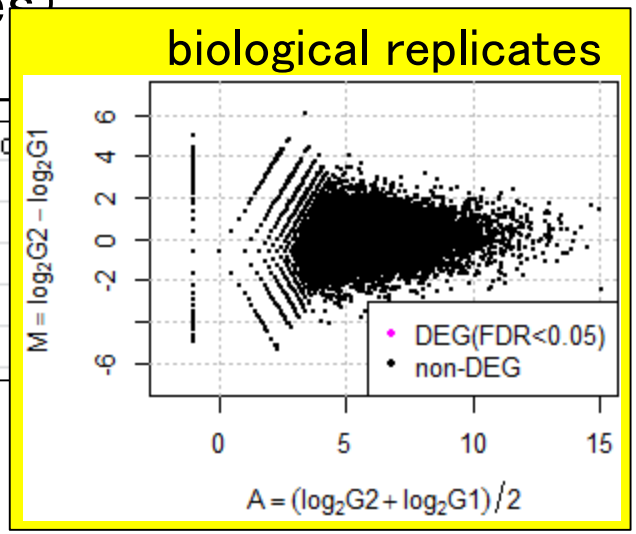


- data_arab.txt (biological replicates)

mock群　　hrcc群

identifier	mock1	mock2	mock3	hrcc1	hrcc2	hrcc
AT1G01010	35	77	40	46	64	
AT1G01020	43	45	32	43	39	
AT1G01030	16	24	26	27	35	
AT1G01040	72	43	64	66	25	
AT1G01050	49	78	90	67	45	

G1群　　G2群



Biological replicatesデータのほうが同一群のばらつきが大きい

平均-分散プロット

- 解析 | 一般 | 上流配列解析 | LDSS(Yamamoto_2007)(last modified 2012/07/17)
- 解析 | 一般 | 上流配列解析 | Relative Appearance Ratio(Yamamoto_2011)(last modified
- 解析 | 基礎 | 平均-分散プロット(Technical replicates)(la modified 2014/02/18) NEW
- 解析 | 基礎 | 平均-分散プロット(Biological replicates)(modified 2014/02/20) NEW
- 解析 | クラスタリング | について(last modified 2014/02/) NEW
- 解析 | クラスタリング | サンプル間 | helust(last modified 2014/02/12) NEW
- 解析 | ク
- 解析 | 発
- 解析 | 発

解析 | 基礎 | 平均-分散プロット(Biological replicates) NEW

Biological replicatesのデータは負の二項分布(negative binomial distribution; 分散 > 平均)に従うことを検証します。つまり、ポアソン分布(分散 = 平均)よりももっとばらつきが大きいということを言っています。ここでは、横軸: 平均、縦軸: 分散としたプロットを描画してその傾向を眺めます。

「ファイ

1. サンプル

Biological
までがDE
DEGであ

```
in_f <-
out_f1
out_f2
out_f3
out_f4
out_f5
param_G
param_G
param_f

#入力フ
data <-
colSums

#前処理(
data.cl
```

4. サンプルデータ8の 26,221 genes×6 samplesのリアルデータ(data_arab.txt; mock 3サンプル vs. hrcc 3サンプル)の場合:

3.と基本的に同じデータですが、ファイルの読み込みから行う一般的なやり方です。

```
in_f <- "data_arab.txt"                 #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge4_G1.txt"                #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge4_G1.png"                #出力ファイル名を指定してout_f2に格納
out_f3 <- "hoge4_G2.txt"                #出力ファイル名を指定してout_f3に格納
out_f4 <- "hoge4_G2.png"                #出力ファイル名を指定してout_f4に格納
out_f5 <- "hoge4_all.png"               #出力ファイル名を指定してout_f5に格納
param_G1 <- 3                           #G1群のサンプル数を指定
param_G2 <- 3                           #G2群のサンプル数を指定
param_fig <- c(380, 420)                #ファイル出力時の横幅と縦幅を指定(単位はピクセル)

#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で指定したファイル
colSums(data)                           #総リード数を表示

#前処理(データ正規化; 群ごとに総リード数の平均値を揃えている)
data.cl <- c(rep(1, param_G1), rep(2, param_G2))#G1群を1、G2群を2としたベクトルdata.clを作成
hoge <- data[,data.cl==1]               #G1群のデータのみ抽出している
nf <- mean(colSums(hoge))/colSums(hoge)#G1群の正規化係数を計算した結果をnfに格納
G1 <- sweep(hoge, 2, nf, "*")           #正規化係数を各列に掛けた結果をG1に格納
colSums(G1)                             #総リード数を表示
hoge <- data[,data.cl==2]               #G2群のデータのみ抽出している
nf <- mean(colSums(hoge))/colSums(hoge)#G2群の正規化係数を計算した結果をnfに格納
```

実行してみましょう

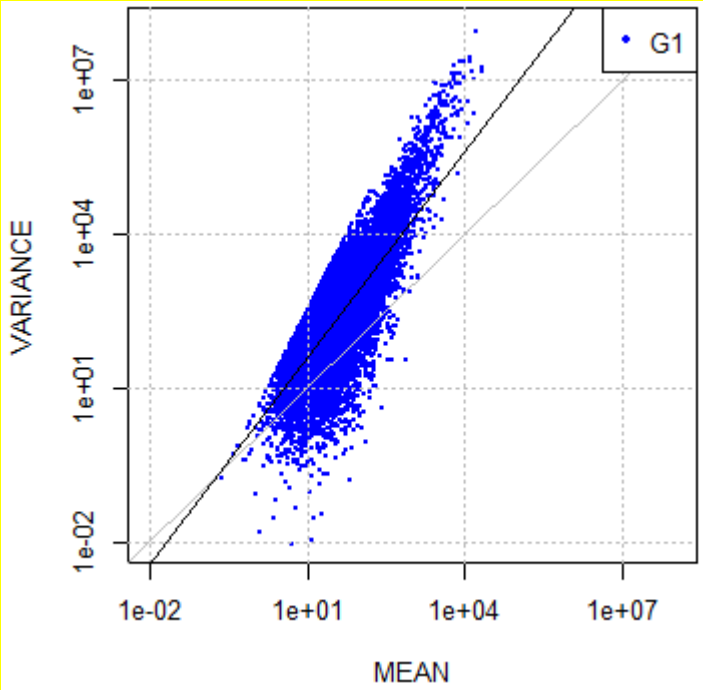
4. サンプルデータ8の 26,221 genes×6 samplesのリアルデータ(data_arab.txt; mock 3サンプル vs. hrcc 3サンプル)の場合:

3.と基本的に同じデータですが、ファイルの読み込みから行う一般的なやり方です。

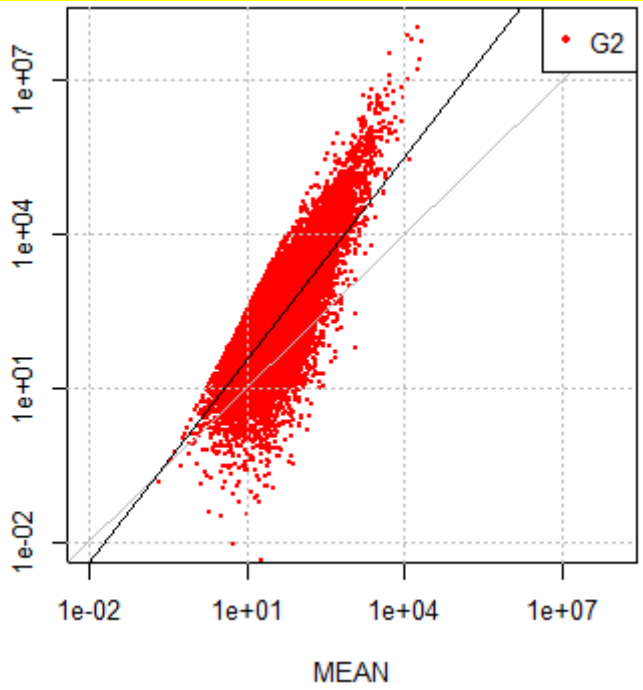
```
in_f <- "data_arab.txt"                  #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge4_G1.txt"                 #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge4_G1.png"                 #出力ファイル名を指定してout_f2に格納
out_f3 <- "hoge4_G2.txt"                 #出力ファイル名を指定してout_f3に格納
out_f4 <- "hoge4_G2.png"                 #出力ファイル名を指定してout_f4に格納
out_f5 <- "hoge4_all.png"                #出力ファイル名を指定してout_f5に格納
param_G1 <- 3                            #G1群のサンプル数を指定
param_G2 <- 3                            #G2群のサンプル数を指定
param_fig <- c(380, 420)                 #ファイル出力時の横幅と縦幅を指定(単位はピ

#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で指定したファイル
colSums(data)                            #総リード数を表示

#前処理(データ正規化; 群ごとに総リード数の平均値を揃えている)
```

Biological replicatesのデータは:
- **VARIANCE > MEAN**
- 負の二項(NB)分布に従う
- NBモデルが適用可能

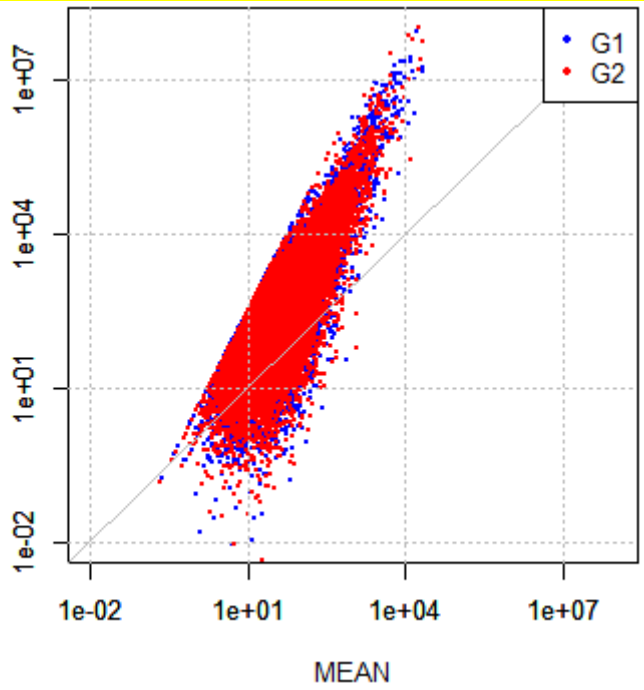
hoge4_G1.png



hoge4_G2.png



hoge4_all.png



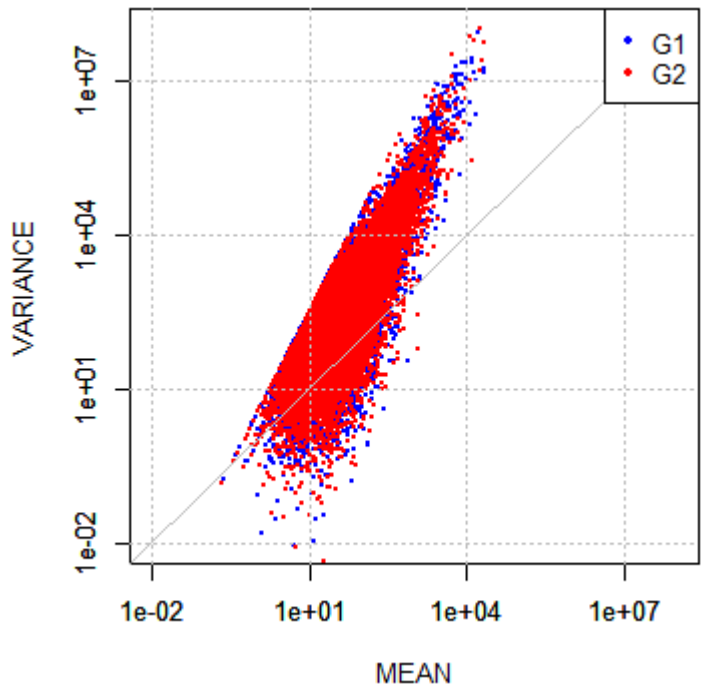
平均-分散プロットの結果の解釈

- 分散(VARIANCE)は平均(MEAN)よりも大きい傾向にある
  - VARIANCE > MEAN ($y = x$の灰色直線がVARIANCE = MEANに相当)
  - VARIANCE = MEAN + φ ×MEAN^2 (φ ＞0；φ : dispersion parameter)で表現される

G1群　G2群

26,222 genes

identifier	mock1	mock2	mock3	hrcc1	hrcc2	hrcc3
AT1G01010	35	77	40	46	64	60
AT1G01020	43	45	32	43	39	49
AT1G01030	16	24	26	27	35	20
AT1G01040	72	43	64	66	25	90
AT1G01050	49	78	90	67	45	60
AT1G01060	0	15	2	0	21	8
AT1G01070	16	34	6	9	20	1
AT1G01080	170	191	382	127	98	184
AT1G01090	291	346	563	171	116	453
AT1G01100	113	125	246	78	27	361
AT1G01110	0	1	1	0	0	0

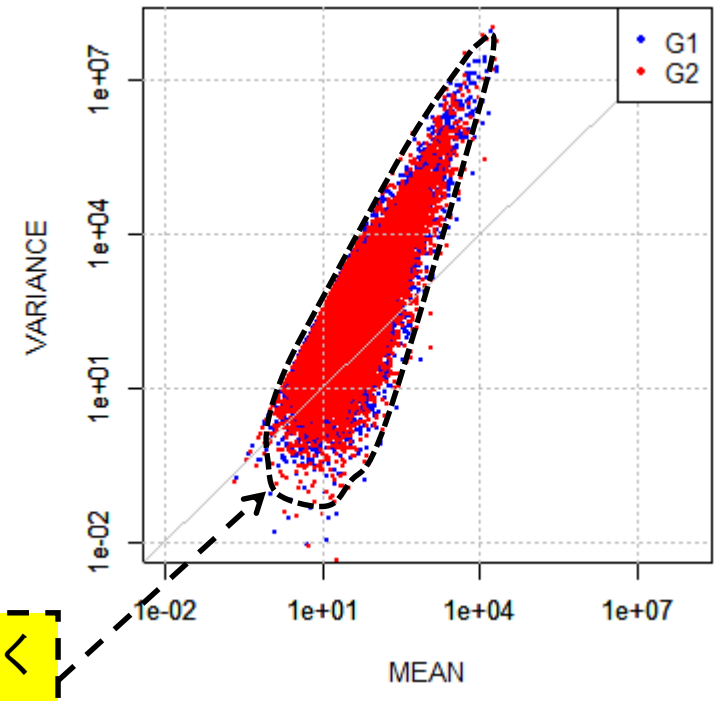


負の二項分布（negative binomial distribution; NB分布）はbiological replicatesデータのばらつきの程度を表現する基本モデル

統計的手法とは

- 同一群に属する反復実験データを用いてばらつきの程度を把握
  - モデル構築に相当
  - 負の二項分布(NB)モデル:VARIANCE = MEAN + φ ×MEAN^2 (φ ＞0)
    - Biological replicatesデータ表現用
  - ポアソン分布モデル:VARIANCE = MEAN
    - Technical replicatesデータ表現用
    - NBモデルのφ = 0の場合に相当

      →NBモデルの数式でポアソンモデルに対応可能



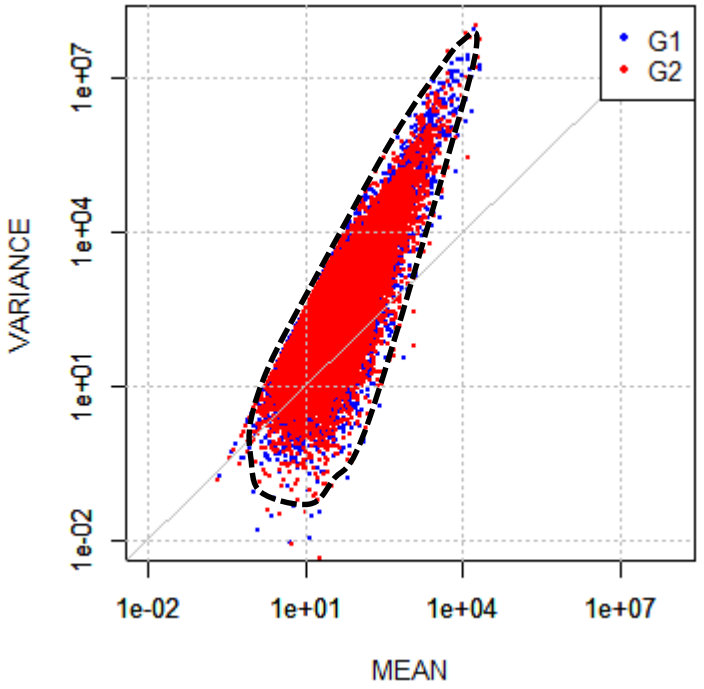
発現変動していない遺伝子でもこれだけばらつく
というデータの素性を知ることが重要なんです

統計的手法とは

- 同一群に属する反復実験データを用いてばらつきの程度を把握
  - モデル構築に相当
  - 負の二項分布(NB)モデル: $VARIANCE = MEAN + \varphi \times MEAN^2$ ($\varphi > 0$)
    - Biological replicatesデータ表現用
  - ポアソン分布モデル: VARIANCE = MEAN
    - Technical replicatesデータ表現用
    - NBモデルの$\varphi = 0$の場合に相当

      →NBモデルの数式でポアソンモデルに対応可能

数式で表現するのは、検定結果として*p*値を出力したいからです。どの数式を使ってどれだけうまくnon-DEG分布を把握しきれるかがポイント



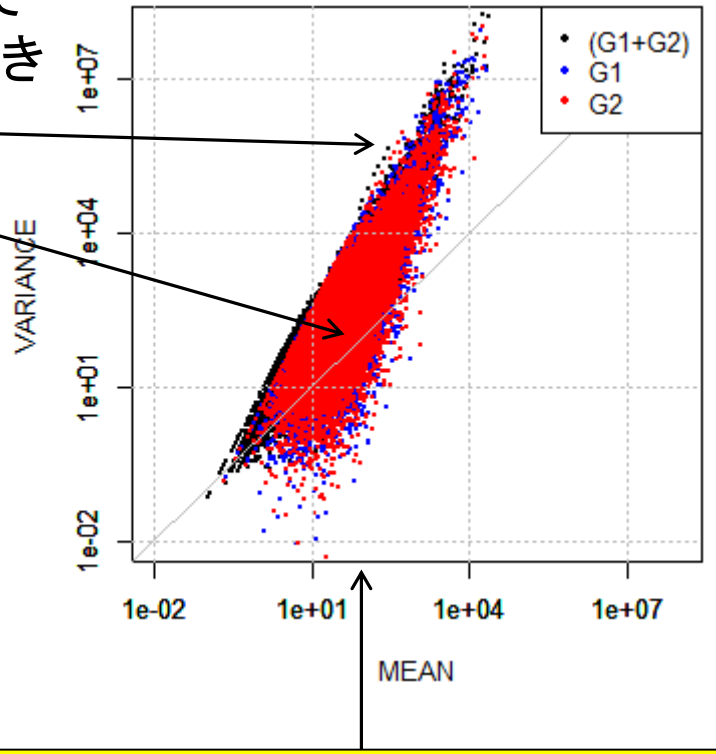
似非(エセ)検定

- 基本的な考え方
  - 評価軸:平均(MEAN)と分散(VAR)
  - non-DEG分布:同一群のばらつき(モデルに相当)
  - 検定:(G1+G2)群のMEANとVARをプロット(●)しておき、そのあとに各群のプロット(●と●)を重ね書き
    - non-DEG分布のど真ん中に位置した●:p値 = 1
    - non-DEG分布から遠く離れた位置の●:p値 = 0

(G1+G2)群

G1群　G2群

identifier	mock1	mock2	mock3	hrcc1	hrcc2	hrcc3
AT1G01010	35	77	40	46	64	60
AT1G01020	43	45	32	43	39	49
AT1G01030	16	24	26	27	35	20
AT1G01040	72	43	64	66	25	90
AT1G01050	49	78	90	67	45	60
AT1G01060	0	15	2	0	21	8



MEAN = 100となるDEGをプロットしてみる

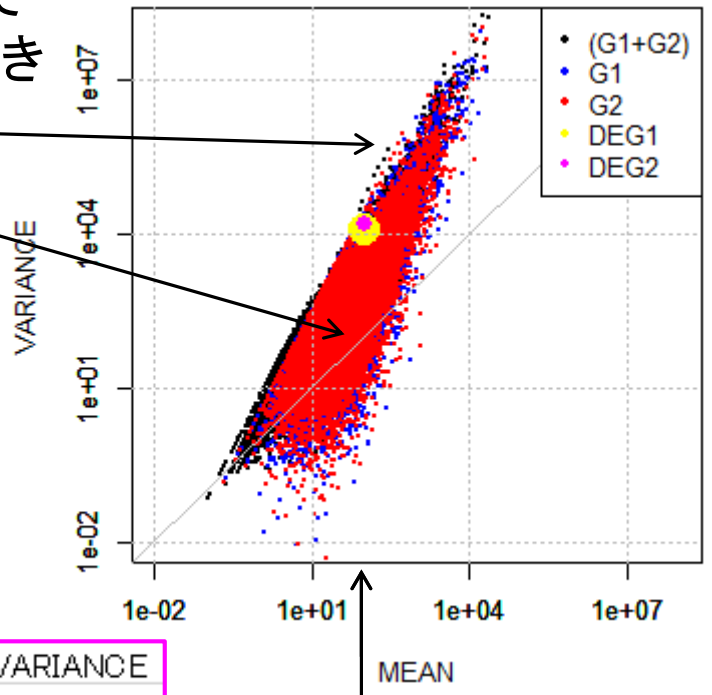
似非(エセ)検定

- 基本的な考え方
  - 評価軸:平均(MEAN)と分散(VAR)
  - non-DEG分布:同一群のばらつき(モデルに相当)
  - 検定:(G1+G2)群のMEANとVARをプロット(●)しておき、そのあとに各群のプロット(●と●)を重ね書き
    - non-DEG分布のど真ん中に位置した●:p値 = 1
    - non-DEG分布から遠く離れた位置の●:p値 = 0

(G1+G2)群

G1群　G2群

identifier	mock1	mock2	mock3	hrcc1	hrcc2	hrcc3
AT1G01010	35	77	40	46	64	60
AT1G01020	43	45	32	43	39	49
AT1G01030	16	24	26	27	35	20



	mock1	mock2	mock3	hrcc1	hrcc2	hrcc3	MEAN	VARIANCE
DEG1	0	0	0	200	200	200	100	12000
DEG2	0	0	0	200	300	100	100	16000

DEG1(黄色)、DEG2(マゼンタ)ともにnon-DEG分布の端の方に位置することがわかる

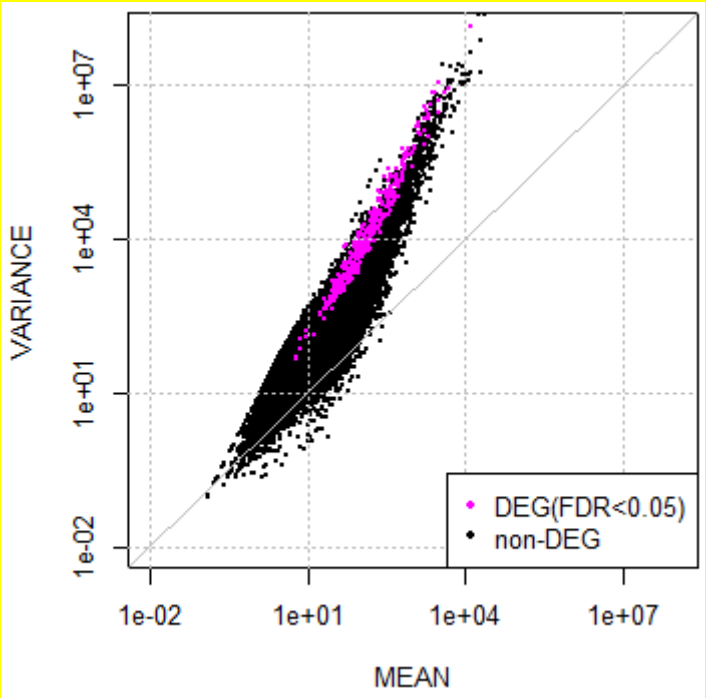
平均-分散プロット:実際のDEG同定

6. サンプルデータ8の26,221 genes×6 samplesのリアルデータ(data_arab.txt; mock 3サンプル vs. hrcc 3サンプル)の場合:

TCCパッケージ中のiDEGES/edgeR正規化後のデータを用いて、2つの群をまとめてプロットしています。
iDEGES/edgeR-edgeR解析パイプライン適用後のFDR < 0.05を満たす遺伝子をマゼンタで色づけしています。

```
in_f <- "data_arab.txt"                    #入力ファイル名を指定してin_fに格納
out_f <- "hoge6.png"                       #出力ファイル名を指定してout_fに格納
param_G1 <- 3                              #G1群(kidney)のサンプル数を指定
param_G2 <- 3                              #G2群(liver)のサンプル数を指定
param_FDR <- 0.05                          #DEG検出時のfalse discovery rate (FDR
param_fig <- c(380, 420)                   #ファイル出力時の横幅と縦幅を指定(単位は

#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

#前処理(TCCクラスオブジェクトの作成)
data.cl <- c(rep(1, param_G1), rep(2, param_G2))#G1群を1、G2群を2としたベクト
tcc <- new("TCC", data, data.cl)           #TCCクラスオブジェクトtccを作成

#本番(iDEGES/edgeR正規化)
tcc <- calcNormFactors(tcc, norm.method="tmm", test.method="edger",#正規化を
                       iteration=3, FDR=0.1, floorPDEG=0.05)#正規化を実行した
normalized.count <- getNormalizedData(tcc)#正規化後のデータを取り出してnormali
```

hoge6.png



FDR < 0.05を満たす366個のDEGの分散は、それ以外のnon-DEGよりも全体的に大きいので妥当

- 解析 | 発現変動 | について(last modified 2013/08/29)
- 解析 | 発現変動 | 2群間 | 対応なし | について(last modified 2013/0[illegible]/12)
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC (Sun_2013)([illegible]modified 2014/02/04)推奨 NEW
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | edgeR (Robinson [illegible]r0)(last modified 2014/01/30) NEW
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | SAMseq (Li_2013)(last modified 2014/01/30) NEW
- 解析 | 発現変動 | 2群間 | 対応なし | 複製なし | TCC (Sun_2013)(last modified 2014/02/04)推奨 NEW
- 解析 | 発現[illegible]
- 解析 | 発現[illegible]
- 解析 | 発現[illegible]
- 解析 | 発現[illegible]
- 解析 | 発現[illegible]
- 解析 | 発現[illegible]
- 解析 | 発現[illegible]
- 解析 | 発現[illegible]
- 解析 | 発現[illegible]
- 解析 | 発現[illegible]

- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC (Sun_2013)

解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC (Sun_2013) NEW

TCCを用いたやり方を示します。
内部的にiDEGES/edgeR(Sun_2013)正規化を実行したのち、edgeRパッケージ中のexact testで発現変動遺伝子(Differentially expressed Genes; DEGs)検出を行っています。TCC原著論文中のiDEGES/edgeR-edgeRという解析パイプラインに相当します。全てTCCパッケージ(Sun et al., BMC Bioinformatics, 2013)内で完結します。
「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1[illegible]の10,000 genes×6 samplesのカウントデータ(data_hypodata_3vs3.txt)の場合:

Biological replic[illegible]
gene_2000までか[illegible]
gene_10000までか[illegible]

```
in_f <- "data_h
out_f1 <- "hoge
out_f2 <- "hoge
param_G1 <- 3
param_G2 <- 3
param_FDR <- 0.
param_fig <- c(

#必要なパッケージ
library(TCC)

#入力ファイルの読
data <- read.tal

#前処理(TCCクラス
data.cl <- c(re
tcc <- new("TCC
```

6. サンプルデータ8の26,221 genes×6 samplesのリアルデータ(data_arab.txt; mock 3サンプル vs. hrcc 3サンプル)の場合:

「FDR閾値を満たすもの」と「fold-change閾値を満たすもの」それぞれのM-A plotを作成しています。

```
in_f <- "data_arab.txt"                 #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge6.txt"                   #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge6_FDR.png"               #出力ファイル名を指定してout_f2に格納
out_f3 <- "hoge6_FC.png"                #出力ファイル名を指定してout_f3に格納
param_G1 <- 3                           #G1群のサンプル数を指定
param_G2 <- 3                           #G2群のサンプル数を指定
param_FDR <- 0.05                       #DEG検出時のfalse discovery rate (FDR)閾値を指定
param_FC <- 2                           #fold-change閾値(param_FC倍)を指定
param_fig <- c(400, 380)                #MA-plot描画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(TCC)                            #パッケー[illegible]

#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で指定したファイル0

#前処理(サブセットの抽出)
data <- data[,1:(param_G1+param_G2)]    #最初の(param_G1+param_G2)列分のデータを抽出している

#前処理(TCCクラスオブジェクトの作成)
data.cl <- c(rep(1, param_G1), rep(2, param_G2))#G1群を1、G2群を2としたベクトルdata.clを作成
tcc <- new("TCC", data, data.cl)        #TCCクラスオブジェクトtccを作成
```

通常は、平均-分散プロットではなく
M-A plotでDEG検出結果を示します

6. サンプルデータ8の26,221 genes×6 samplesのリアルデータ(data_arab.txt; mock 3サンプル vs. hrcc 3サンプル)の場合:

• 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC (Sun 2013)

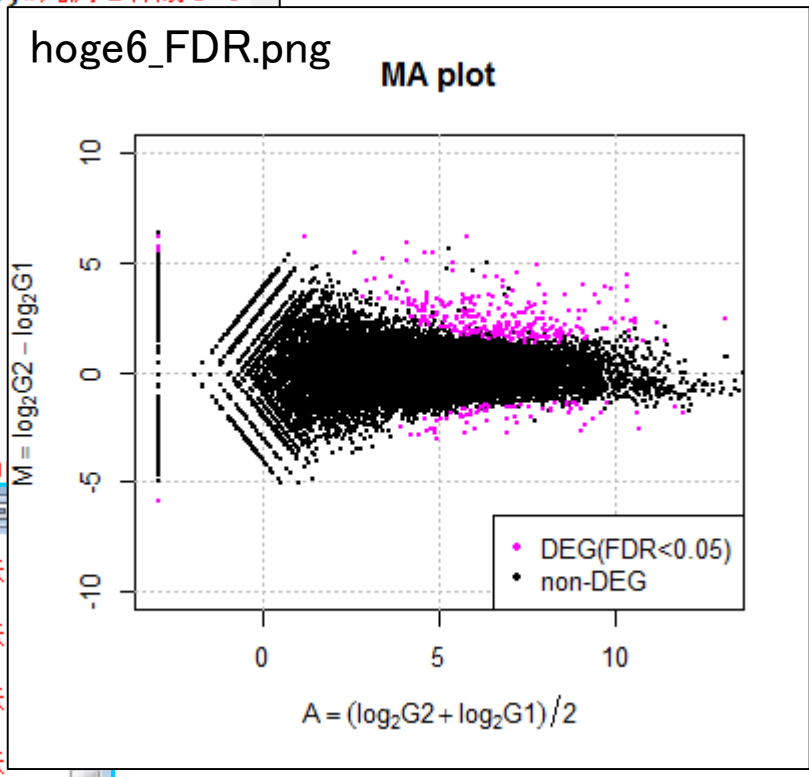
「FDR閾値を満たすもの」と「fold-change閾値を満たすもの」それぞれのM-A plotを作成しています。

```
#ファイルに保存(M-A plot; FDR)
png(out_f2, pointsize=13, width=param_fig[1], height=param_fig[2])#出力ファイルの各種パラメータ
plot(tcc, FDR=param_FDR, xlim=c(-3, 13), ylim=c(-10, 10))#param_FDRで指定した閾値を満たすDEGを
legend("bottomright", c(paste("DEG(FDR<", param_FDR, ")", sep=""), "non-DEG"),#凡例を作成してい
       col=c("magenta", "black"), pch=20)#凡例を作成している
dev.off()                                  #おまじない
sum(tcc$stat$q.value < 0.05)               #FDR < 0.05を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.10)               #FDR < 0.10を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.20)               #FDR < 0.20を満たす遺伝子数を表示
sum(tcc$stat$q.value < 0.30)               #FDR < 0.30を満たす遺伝子数を表示

#ファイルに保存(M-A plot; fold-change)
param_FC <- 2                              #fold-change閾値(param_FC倍)を指定
M <- getResult(tcc)$m.value                #M-A plotのM値を抽出
hoge <- rep(1, length(M))                  #初期値を1にしたベクトルhogeを作成
hoge[abs(M) > log2(param_FC)] <- 2          #条件を満たす位置に2を代入
cols <- c("black", "magenta")              #色情報を指定してcolsに格納

png(out_f3, pointsize=13, width=param_fig[1], height=param_fig[2])#出力ファイル
plot(
legen
```

R Console

```
> sum(tcc$stat$q.value < 0.05)          #FDR < 0.05を満たす遺伝子数を表示
[1] 366
> sum(tcc$stat$q.value < 0.10)          #FDR < 0.10を満たす遺伝子数を表示
[1] 499
> sum(tcc$stat$q.value < 0.20)          #FDR < 0.20を満たす遺伝子数を表示
[1] 711
> sum(tcc$stat$q.value < 0.30)          #FDR < 0.30を満たす遺伝子数を表示
[1] 899
>
```

hoge6_FDR.png



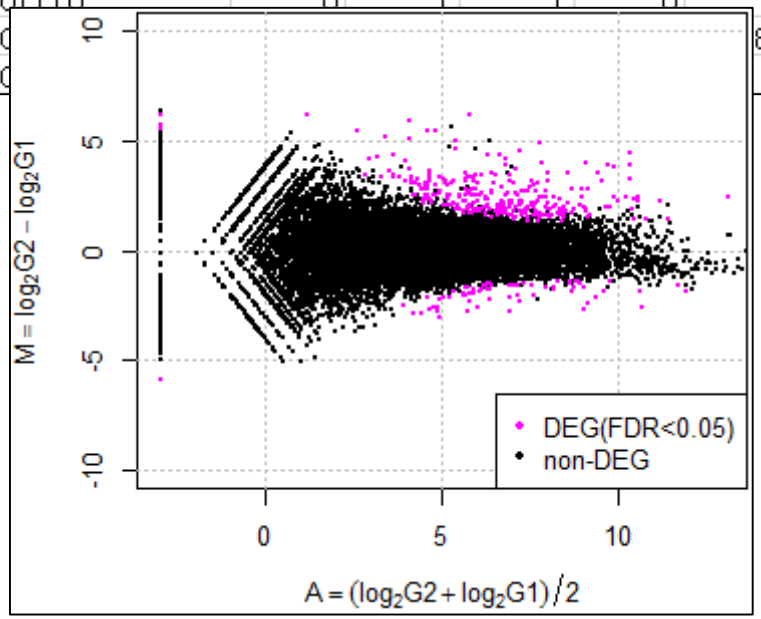
FDR < 0.05を満たすものは366個

出力ファイル（hoge6.txt）の説明

入力データ

p-valueとその順位

rownames(tcc$count)	mock1	mock2	mock3	hrcc1	hrcc2	hrcc3	gene_id	a.value	m.value	p.value	q.value	rank	estimatedDEG
AT1G01010	35	77	40	46	64	60	AT1G01010	5.850	0.293	0.609	1.000	13922	0
AT1G01020	43	45	32	43	39	49	AT1G01020	5.465	0.190	0.722	1.000	16786	0
AT1G01030	16	24	26	27	35	20	AT1G01030	4.730	0.574	0.401	1.000	8870	0
AT1G01040	72	43	64	66	25	90	AT1G01040	5.885	−0.050	0.943	1.000	22169	0
AT1G01050	49	78	90	67	45	60	AT1G01050	6.044	−0.211	0.633	1.000	14680	0
AT1G01060	0	15	2	0	21	8	AT1G01060	3.135	1.091	0.521	1.000	11669	0
AT1G01070	16	34	6	9	20	1	AT1G01070	4.054	−0.556	0.626	1.000	14500	0
AT1G01080	170	191	382	127	98	184	AT1G01080	7.506	−0.749	0.065	0.763	2219	0
AT1G01090	291	346	563	171	116	453	AT1G01090	8.224	−0.740	0.083	0.850	2553	0
AT1G01100	113	125	246	78	27	361	AT1G01100	7.145	−0.239	0.689	1.000	15936	0
AT1G01110	0	1	1	0	0	0	AT1G01110	−2.969	−1.348	0.639	1.000	14806	0
AT1G						83	AT1G01120	7.464	−0.733	0.064	0.760	2210	0
AT1G						2	AT1G01130	2.327	−1.247	0.487	1.000	10906	0



M-A plotのA値とM値

q-value

FDR閾値判定結果。q-value < 0.05 を満たすDEGが1、non-DEGが0。

基本的には、これらの結果を用います

Contents（Rで…）

- ゲノム解析
  - アノテーションファイルを読み込んで目的のキーワードを含む行のみ抽出
  - multi-FASTAファイルを自在に解析
    - 配列長分布、GC含量、フィルタリング、部分配列の切り出しなど
    - 連続塩基の出現頻度（CpG）解析、ゲノム配列取得など
- トランスクリプトーム解析
  - 研究目的別留意点：サンプル内とサンプル間の違い
  - マッピング → カウント情報取得
  - データを眺める：クラスタリングやM-A plot
  - 理想的な実験デザイン
  - なぜx倍発現変動という議論がだめなんですか？
  - モデルとか分布って、自分の解析結果にどういう影響を与えているの？
  - **多重比較問題：FDRって何？**

多重比較問題：FDRって何？

- DEGかnon-DEGかを判定する閾値を決める問題
  - 有意水準5%というのがp-value < 0.05に相当
  - False discovery rate (FDR) 5%というのがq-value < 0.05に相当
- 発現変動**ランキング結果は不変**なので上位x個という決め打ちの場合にはこの問題とは無関係

rownames(tcc$count)	mock1	mock2	mock3	hrcc1	hrcc2	hrcc3	gene_id	a.value	m.value	p.value	q.value	rank	estimatedDEG
AT1G01010	35	77	40	46	64	60	AT1G01010	5.850	0.293	0.609	1.000	13922	0
AT1G01020	43	45	32	43	39	49	AT1G01020	5.465	0.190	0.722	1.000	16786	0
AT1G01030	16	24	26	27	35	20	AT1G01030	4.730	0.574	0.401	1.000	8870	0
AT1G01040	72	43	64	66	25	90	AT1G01040	5.885	-0.050	0.943	1.000	22169	0
AT1G01050	49	78	90	67	45	60	AT1G01050	6.044	-0.211	0.633	1.000	14680	0
AT1G01060	0	15	2	0	21	8	AT1G01060	3.135	1.091	0.521	1.000	11669	0
AT1G01070	16	34	6	9	20	1	AT1G01070	4.054	-0.556	0.626	1.000	14500	0
AT1G01080	170	191	382	127	98	184	AT1G01080	7.506	-0.749	0.065	0.763	2219	0
AT1G01090	291	346	563	171	116	453	AT1G01090	8.224	-0.740	0.083	0.850	2553	0
AT1G01100	113	125	246	78	27	361	AT1G01100	7.145	-0.239	0.689	1.000	15936	0
AT1G01110	0	1	1	0	0	0	AT1G01110	-2.969	-1.348	0.639	1.000	14806	0
AT1G01120	228	189	270	147	83	174	AT1G01120	7.464	-0.733	0.064	0.760	2210	0
AT1G01130	9	11	1	0	2	9	AT1G01130	2.327	-1.247	0.487	1.000	10906	0

DEG数に関するよりよい結果を得たい場合には、
p-valueではなくq-value閾値を利用しましょう

多重比較問題：FDRって何？

- *p*−value (false positive rate; FPR)
  - 本当はDEGではないにもかかわらずDEGと判定してしまう確率
  - 全遺伝子に占めるnon−DEGの割合（分母は**遺伝子総数**）
  - 例：10,000個のnon−DEGからなる遺伝子を*p*−value < 0.05で検定すると、10,000 × 0.05 = 500個程度のnon−DEGを間違ってDEGと判定することに相当
    - 実際のDEG検出結果が900個だった場合：500個は偽物で400個は本物と判断
    - 実際のDEG検出結果が510個だった場合：500個は偽物で10個は本物と判断
    - 実際のDEG検出結果が500個以下の場合：全て偽物と判断
- *q*−value（false discovery rate: FDR）
  - DEGと判定した中に含まれるnon−DEGの割合
  - DEG中に占めるnon−DEGの割合（分母は**DEGと判定された数**）
  - non−DEGの期待値を計算できれば、*p*値でも上位*x*個でもDEGと判定する手段はなんでもよい。以下は10,000遺伝子の検定結果でのFDR計算例
    - $p < 0.001$を満たすDEG数が100個の場合：FDR = 10,000 × 0.001/100 = 0.1
    - $p < 0.01$を満たすDEG数が400個の場合：FDR = 10,000 × 0.01/400 = 0.25
    - $p < 0.05$を満たすDEG数が926個の場合：FDR = 10,000 × 0.05/926 = 0.54

2群間比較：technical replicatesデータ

- data_marioni.txt（ヒトのデータ）


kidney（腎臓）


liver（肝臓）

18,110 genes

rownames(data)	R1L1Kidney	R1L3Kidney	R1L7Kidney	R2L2Kidney	R2L6Kidney	R1L2Liver	R1L4Liver	R1L6Liver	R1L8Liver	R2L3Liver
ENSG00000000003	178	167	179	172	151	138	178	175	187	169
ENSG00000000005	0	0	0	0	1	0	0	0	0	0
ENSG00000000419	53	78	64	72	71	30	42	41	33	43
ENSG00000000457	22	33	30	27	30	47	60	37	42	62
ENSG00000000460	9	7	18	14	9	19	9	13	14	19
ENSG00000000938	14	18	7	27	15	42	34	39	37	45
ENSG00000000971	28	27	28	34	42	470	502	500	490	536
ENSG00000001036	154	195	140	168	187	56	67	55	64	67
ENSG00000001084	77	72	71	93	84	305	330	322	292	325
ENSG00000001167	41	37	35	44	40	40	26	30	33	32
ENSG00000001460	24	29	30	31	23	4	5	3	6	4
ENSG00000001461	23	26	32	35	32	3	4	4	2	6
ENSG00000001497	55	48	52	59	70	34	32	30	38	50
ENSG00000001561	139	166	136	154	152	30	33	44	39	38
ENSG00000001617	136	130	112	153	156	12	5	7	5	4

G1群　　G2群

発現変動遺伝子(DEG)がないデータで2群間比較をしてみると…

- 解析 | 発現変動 | について(last modified 2013/08/29)
- 解析 | 発現変動 | 2群間 | 対応なし | について(last modified 2013/0[illegible]12)
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC (Sun_2013)[illegible]modified 2014/02/04)推奨 NEW
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | edgeR (Robinson [illegible]r0)(last modified 2014/01/30) NEW
- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | SAMseq (Li 2013)(last modified 2014/01/30) NEW
- 解析
- 解析
- 解析
- 解析
- 解析
- 解析
- 解析
- 解析
- 解析
- 解析
- 解析
- 解析

- 解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC (Sun_2013)

解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC (Sun_2013) NEW

TCCを用いたやり方を示します。
内部的にiDEGES/edgeR(Sun 2013)正規化を実行したのち、edgeRパッケージ中のexact testで発現変動遺伝子
(Di[illegible]ntially expressed Genes: DEGs)検出を行っています。TCC原著論文中のiDEGES/edgeR-edgeRという解析パ
イプ
「ファ[illegible]

5. サンプルデータ4の18,110 genes×10 samplesのリアルデータ(data_marioni.txt; kidney 5サンプル vs. liver 5サンプル)の最初の4サンプルを比較する場合:

「FDR閾値を満たすもの」と「fold-change閾値を満たすもの」それぞれのM-A plotを作成しています。

1. サンプ

```
Biolog
gene_2
gene_1
```

```
in_f
out_
out_
para
para
para
para
```

```
#必要
libr
```

```
#入力
data
```

```
#前処
data
tcc
```

```
in_f <- "d
out_f1 <-
out_f2 <-
out_f3 <-
param_G1 <
param_G2 <
param_FDR
param_FC <
param_fig

#必要なパッ
library(TC

#入力ファイ
data <- re

#前処理(サ
data <- da

#前処理(TCC
data.cl <-
tcc <- new
```

R Console

```
> sum(tcc$stat$q.value < 0.05)
[1] 0
> sum(tcc$stat$q.value < 0.10)
[1] 0
> sum(tcc$stat$q.value < 0.20)
[1] 0
> sum(tcc$stat$q.value < 0.30)
[1] 0
> sum(tcc$stat$p.value < 0.05)
[1] 465
> sum(tcc$stat$p.value < 0.10)
[1] 1030
> sum(tcc$stat$p.value < 0.20)
[1] 2277
> sum(tcc$stat$p.value < 0.30)
[1] 3624
>
```

```
てin_fに格納
てout_f1に格納
てout_f2に格納
て
数を指定
covery rate (FDR)閾値を指
m_FC倍)を指定
縦幅を指定(単位はピクセル)

n_G2)列分のデータを抽出し

群を2としたベクトルdata.cl
tccを作成
```

q−value < 0.05を満たすDEG数は0個

p−value < 0.05を満たすDEG数は465個

他の公共カウントデータでも確認できます

- はじめに(last modified 2014/01/30) NEW
- Rのインストールと起動(last modified 2013/09/27)
- サンプルデータ(last modified 2014/02/09) NEW

サンプルデータ NEW

1. Marioni et al., Genome Res., 2008のSupplementary table 2のデータ。

8. NBPSeqパッケージ(Di et al., SAGMB, 10:art24, 2011)中の ArabidopsisのBiological replic vs. G2群3サンプル; Cumbie et al., PLoS One, 2011)です。
   26,221 genes×6 samplesの「複製あり」タグカウントデータ(data_arab.txt)
   オリジナルは"AT4G32850"というIDのものが重複して存在していたため、19520行目のデキストファイルにしています。
9. ReCountデータベース(Frazee et al., BMC Bioinformatics, 2011)
   マッピング済みの遺伝子発現行列形式のデータセットを多数提供しています。
10. Ye

使い慣れているので、私はReCountのデータをよく利用しています。自分でもいろいろと試してみましょう。

ReCount

A multi-experiment resource of analysis-ready RNA-seq gene count datasets

JOHNS HOPKINS BLOOMBERG SCHOOL of PUBLIC HEALTH

Study	PMID	Species	Number of biological replicates	Number of uniquely aligned reads	ExpressionSet	Count table	Phenotype table	Notes
bodymap	22496456	human	19	2,197,622,796	link	link	link	Illumina Human BodyMap 2.0 -- tissue comparison
cheung	20856902	human	41	834,584,950	link	link	link	HapMap - CEU
core	19056941	human	2	8,670,342	link	link	link	lung fibroblasts
gilad	20009012	human	6	41,356,738	link	link	link	liver; males

東大生以外の方も受講可能です（平成26年度もやります）

謝辞

共同研究者

清水 謙多郎　先生（東京大学・大学院農学生命科学研究科）

西山 智明　先生（金沢大学・学際科学実験センター）

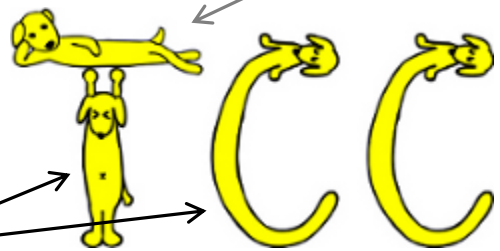
孫 建強　氏（東京大学・大学院農学生命科学研究科・大学院生）

グラント

- 基盤研究(C)（H24-26年度）:「シークエンスに基づく比較トランスクリプトーム解析のためのガイドライン構築」(代表)
- 新学術領域研究(研究領域提案型)(H22年度-):「非モデル生物におけるゲノム解析法の確立」(分担;研究代表者:西山智明)

挿絵やTCCのロゴなど

（妻の）門田 雅世さま作



（有能な秘書の）三浦 文さま作