Rで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→…

http://www.iu.a.u-tokyo.ac.jp/~kadota/r_seq.html

参考URL

(Rで)塩基配列解析(主に次世代シーケンサーのデータ) by 門田幸二 (last modified 2013/01/30, since 2010)

What's new?
・(replicatesのある)RNA-seqデータ用二群間発現変動解析を頑健に行うためのデータ正規化法TbT正規化法の数百倍高速な方法をRパッケージTCCに実装したver. 1.0.0を公開しました。no replicatesの二群間比較にも対応しており、以下の3/7のチュートリアルに間に合わせるべく、1/28までに一通り解析できるようにしました。英語のUser's Guideもありますので、早く利用したい方はここから辿ってご利用ください。(2013/01/30)NEW
・廃止予定の関数名(read.DNAStringSet -> readDNAStringSetなど)や「前処理 | 正規化...」周辺の項目名の変更をしました。(2013/01/16)NEW
・昨年度もありましたが、2013年3月6日(水)～8日(金)にCBRCでやっているHPCIのD-1 次世代シーケンサー 解析入門が開催されます。今年は枠が拡大されて私は3/7の終日担当します。Rパッケージ TCCのチュートリアルも行う予定です。興味ある方はどうぞ。(2013/01/16)NEW

・htmlのタグに問題があるらしくfirefoxでエラーという指摘をTbT論文共著者の西山さんから受けましたのでその周辺を修正しました。(2012/11/15)
・R2.15.2がリリースされていたのでこれに変更しました。(2012/11/15)
・若干項目名を(あまりにも場違いだったものを)変更しました、直接リンクを張ってたかた、すいませんm(_ _)m。(2012/07/12)
・ここで書いてはいないものの、あちこち追加などしてます。(2012/07/06)

- はじめに (last modified 2012/03/29)
- Rのインストールと起動(last modified 2012/11/15)
- サンプルデータ(last modified 2013/01/23) NEW
- イントロダクション | NGS | 各種覚書(last modified 201
- イントロダクション | NGS | 様々なプラットフォーム(last modified 2011/07/15)
- イントロダクション | NGS | リファレンス配列取得(マップされる側)(last modified 2011/02/03)
- イントロダクション | NGS | リファレンス配列取得後の各種情報抽出(特にRefSeq)(last modified 2011/03/20)
- イントロダクション | NGS | リファレンス配列取得後の各種情報抽出2(readFASTA関数の利用)(last modified 2011/04/07)
- イントロダクション | NGS | アノテーション情報取得(refFlatファイル)(last modified 2010/12/07)

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)

トランスクリプトームとは

- ある特定の状態の組織や細胞中に存在する全RNA（転写物、transcripts）の総体
- 様々なトランスクリプトーム解析技術
  - マイクロアレイ
    - cDNAマイクロアレイ、Affymetrix GeneChip、タイリングアレイなど
  - 配列決定に基づく方法
    - EST、SAGEなど、次世代シーケンサー (NGS)
  - 電気泳動に基づく方法
    - Differential Display、AFLPなど

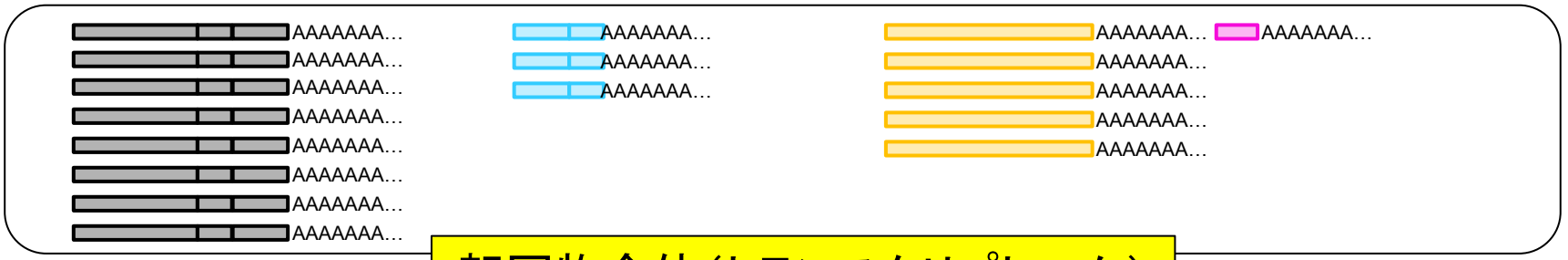
調べたい組織でどの遺伝子がどの程度発現しているのかを一度に観察

トランスクリプトームとは

- ある状態のあるサンプル（例：目）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（例：ヒト）が同じなら不変（目だろうが心臓だろうが…）



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（例：目）のあるゲノムの領域

遺伝子全体（ゲノム）

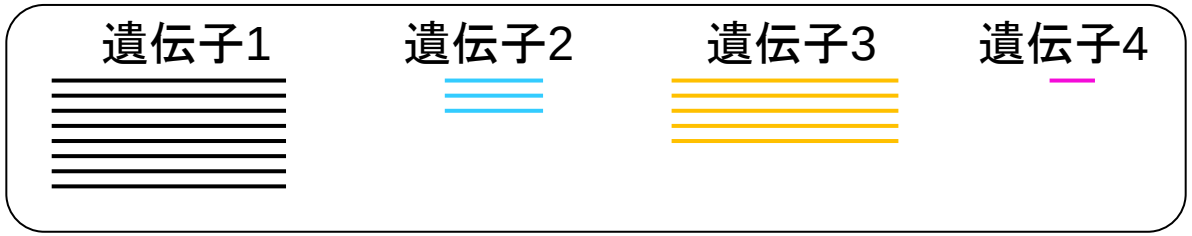
・どの染色体上のどの領域にどの遺伝子があるかは調べる個体（例：ヒト）が同じなら不変（目だろうが心臓だろうが…）

転写物全体（トランスクリプトーム）

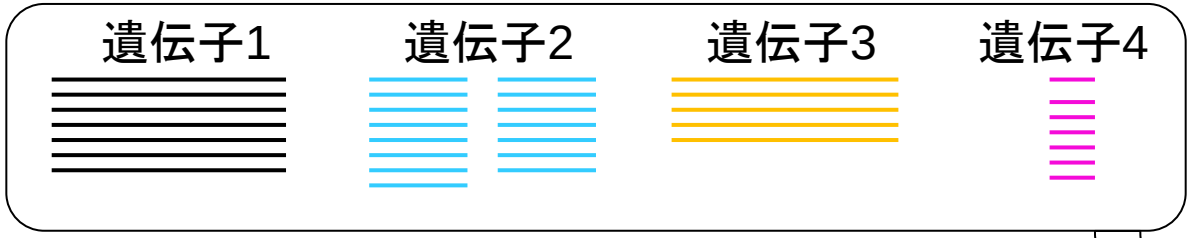
・遺伝子2は光刺激に応答して発現亢進
・遺伝子4も光刺激に応答して発現亢進

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



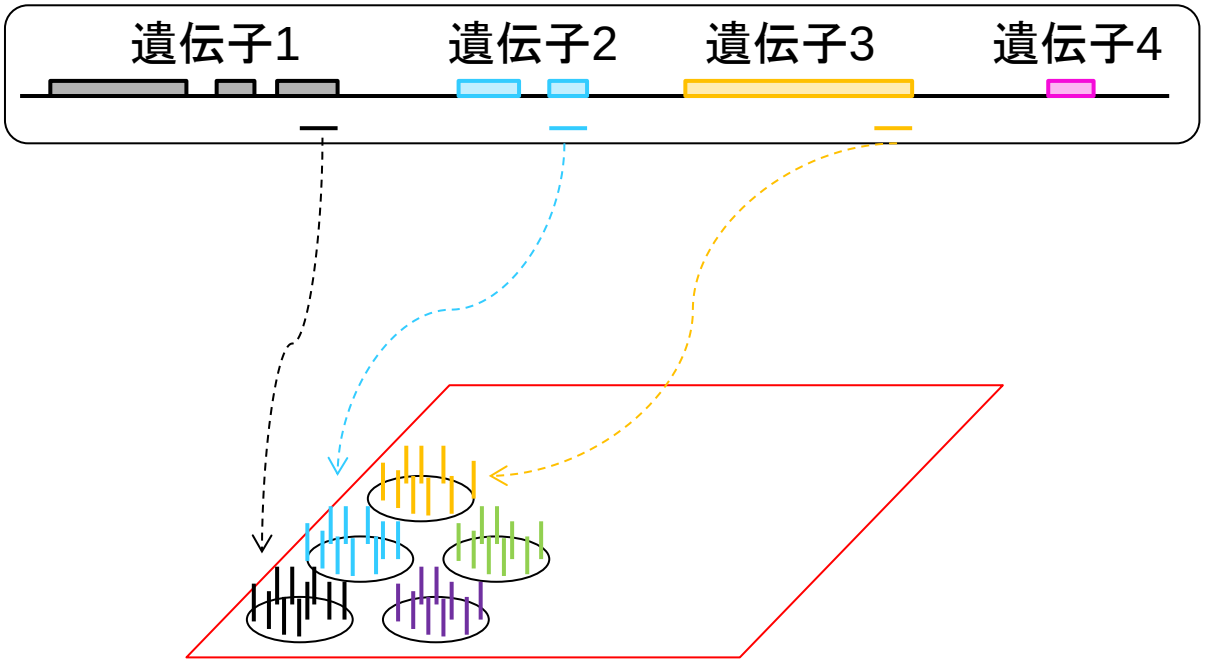
これがいわゆる
「遺伝子発現行列」

	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
- 電気泳動に基づく方法
- 配列決定に基づく方法

トランスクリプトーム取得（マイクロアレイ）

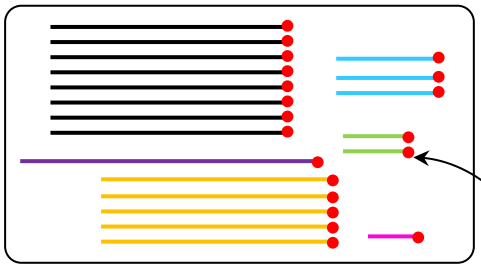
- よく研究されている生き物は多数の遺伝子（の配列情報）がわかっている



わかっている遺伝子（の配列の相補鎖）を搭載した”チップ”

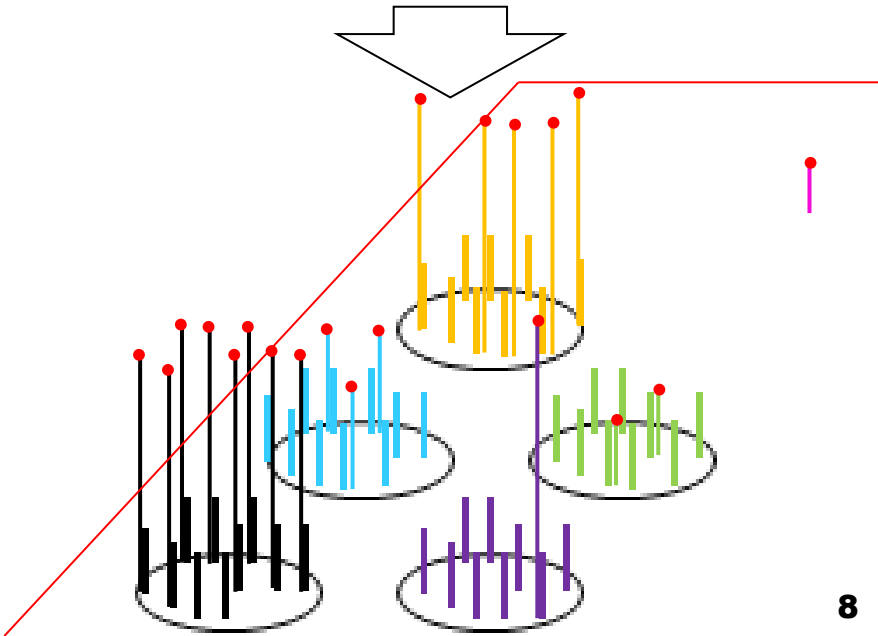
・メーカーによって搭載されている遺伝子の種類が異なる
→搭載されていない遺伝子（未知遺伝子含む、例：遺伝子4）の発現情報は測定不可…

光刺激前（T1）の目のトランスクリプトーム



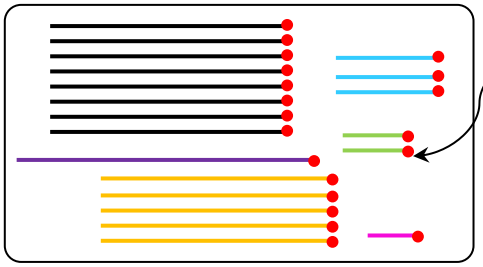
蛍光標識

ハイブリダイゼーション（二本鎖形成）

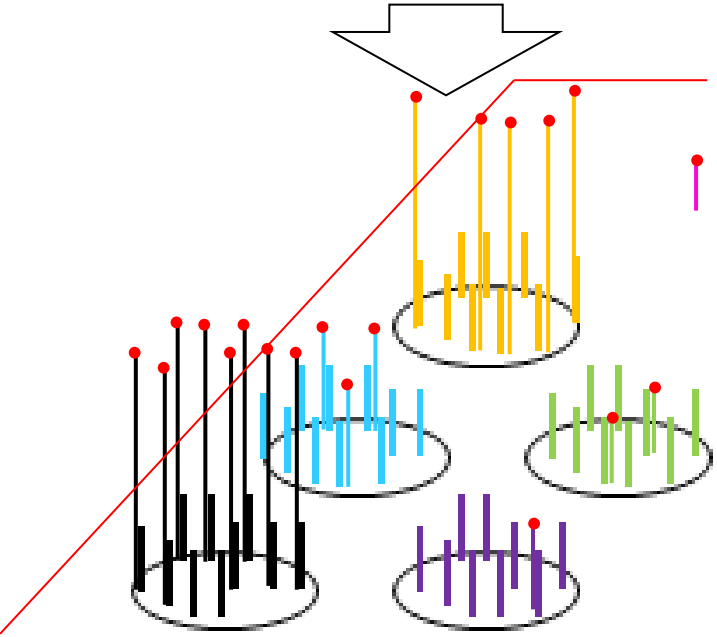


トランスクリプトーム取得（マイクロアレイ）

- 光刺激前（T1）の目のトランスクリプトーム

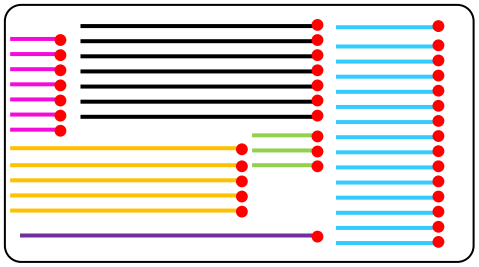


ハイブリダイゼーション
（二本鎖形成）



専用の検出器で各遺伝子に対応する領域の蛍光シグナル強度を測定

光刺激後（T2）の目の
トランスクリプトーム



ハイブリダイゼーション
と
シグナル検出

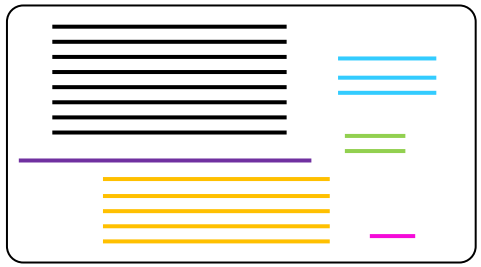
	T1
遺伝子1	8
遺伝子2	3
遺伝子3	5
遺伝子4	?
遺伝子5	…
…	…

T2
7
15
5
?
…
…

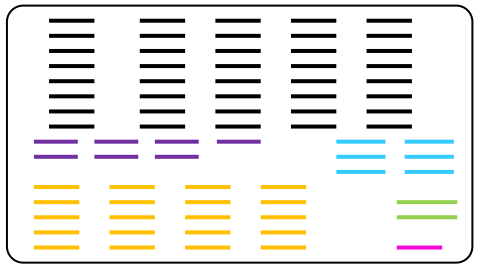
トランスクリプトーム取得（NGS）

- 次世代シーケンサー（Illumina社の場合）

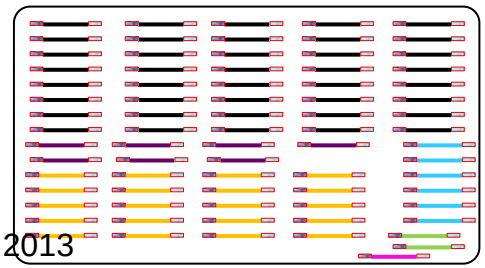
光刺激前（T1）の目のトランスクリプトーム



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

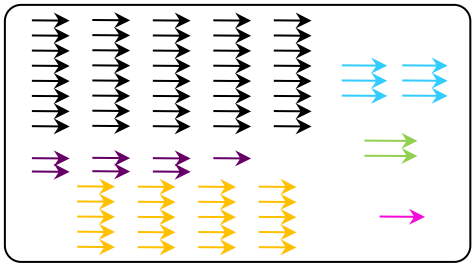


配列決定

・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



FASTQ形式（とFASTA形式）

- FASTA形式
  - 「“>”ではじまる一行のdescription行」と「配列情報」からなる形式
  - NGSのread長は短いので、実質的に一つのリードを二行で表現

```
>SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
```

- FASTQ形式
  - 一行目：「“@”ではじまる一行のdescription行」
  - 二行目：「配列情報」
  - 三行目：「”+”からはじまる一行（のdescription行）」
  - 四行目：「クオリティ情報」

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

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

データ解析のスタート地点

- NGSから得られたFASTQ形式ファイル

様々なMotivation

- ～の原因遺伝子（ガン関連遺伝子とか）を同定したい
- FASTQ以降の一通りの解析ができるようになりたい
- （Windowsの）Rでできることとできないこと
- モデル生物と非モデル生物の解析戦略の違い
- 倍率変化で解析 vs. 分布を使って解析
- いろんなRパッケージがあるけれど…

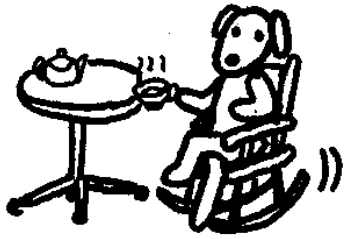
RNA-seqで二つのサンプルを比較し、発現変動遺伝子同定までを行うまでの流れを一通り紹介

A群
腎臓

正常組織
wildtype

B群
肝臓

腫瘍組織
mutant



比較トランスクリプトーム解析の流れ

複数のFASTQファイル

A1.fq, A2.fq, B1.fq, B2.fq

Linuxマシン

リファレンス配列の作成

複数サンプルの混合アセンブルにより、RefSeqのような転写物配列集合（multi-fastaファイル）を得るイメージ

マッピング

どの転写物にどれだけの数のリードがマップされたかという、いわゆる「遺伝子発現行列」を得るイメージ

データ解析

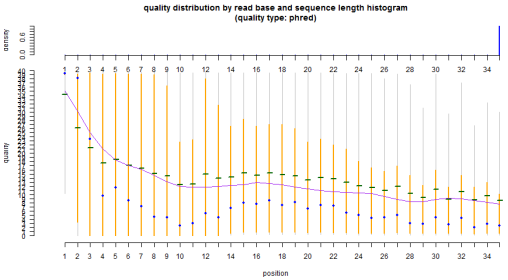
発現変動遺伝子のリストアップや、作図など

すべてが（Windowsの）Rで完結するというわけではありません…

比較トランスクリプトーム解析の流れ

複数のFASTQファイル

クオリティチェック



Linuxマシン

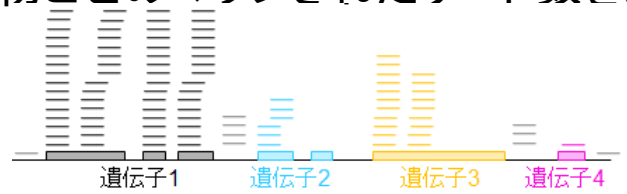
リファレンス配列の作成

アセンブル結果（multi-fasta）ファイルから平均長やトータルの長さなどの基本情報を抽出

Total length (bp)	2993
Number of contigs	4
Average length	748.3
Median length	784
Max length	888
Min length	537
N50	886
GC content	0.524

マッピング

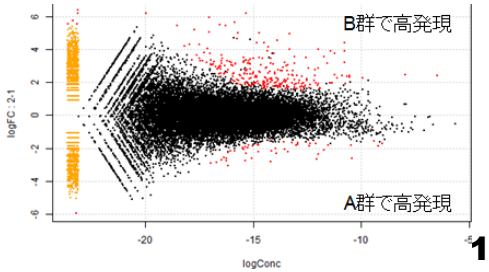
マッピング結果（BED形式）ファイルを入力として、転写物ごとのマップされたリード数をカウント



遺伝子1	40
遺伝子2	6
遺伝子3	20
遺伝子4	1

データ解析

発現変動遺伝子のリストアップや、作図など



大規模計算部分以外は一通りできます

Linuxマシン使用部分の解決策

- 自前で大容量メモリ計算サーバ(Linux)を購入し、必要なソフトのインストールからスタート

  特徴:難易度は高いが思い通りの解析が可能

- Linuxサーバをもつバイオインフォ系の人にお願いする

  特徴:気軽に頼める知り合いがいればいいが、その人次第

- DDBJ Read Annotation Pipelineを利用

  特徴:一番お手軽な選択肢だが、サポートしているプログラムのみ

  データ登録が前提?!だが、手取り足取り丁寧に教えてくれるので個人的にはこちらを推奨

比較トランスクリプトーム解析の流れ

複数のFASTQファイル

↓

リファレンス配列の作成

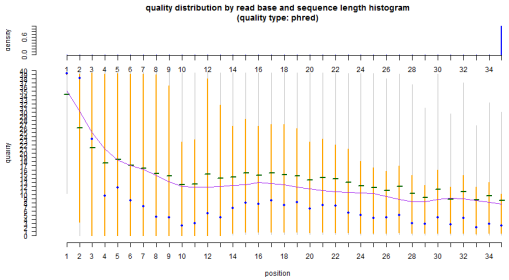
↓

マッピング

↓

データ解析

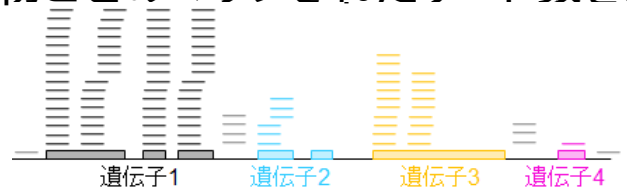
クオリティチェック



アセンブル結果（multi-fasta）ファイルから平均長やトータルの長さなどの基本情報を抽出

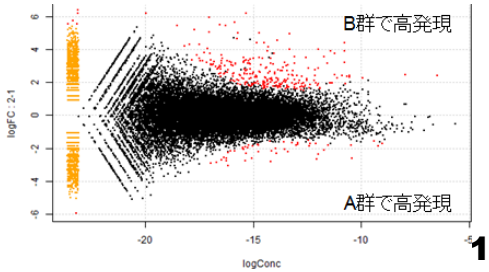
Total length (bp)	2993
Number of contigs	4
Average length	748.3
Median length	784
Max length	888
Min length	537
N50	886
GC content	0.524

マッピング結果（BED形式）ファイルを入力として、転写物ごとのマップされたリード数をカウント



遺伝子1	40
遺伝子2	6
遺伝子3	20
遺伝子4	1

発現変動遺伝子のリストアップや、作図など



様々なde novoアセンブリプログラム

- *de novo* **genome** assembly用プログラム
  - Velvet (Zerbino and Birney, *Genome Res*., **18**: 821-829, 2008)
  - ABySS (Simpson *et al*., *Genome Res*., **19**: 1117-1123, 2009)
  - EULER-SR (Chaisson *et al*., *Genome Res*., **19**: 336-46, 2009)
  - Platanus (unpublished)
- *de novo* **transcriptome** assembly用プログラム（特にIllumina）
  - Multiple-k (Surget-Groba and Montoya-Burgos, *Genome Res*., **20**: 1432-1440, 2010)
  - Trans-ABySS (Robertson *et al*., *Nat. Methods*, **7**: 909-912, 2010)
  - Rnnotator (Martin *et al*., *BMC Genomics*, **11**: 663, 2010)
  - Trinity (Grabherr *et al*., *Nat. Biotechnol*., **29**: 644-652, 2011)
  - Oases (Schulz *et al*., *Bioinformatics*, **28**: 1086-1092, 2012)

ゲノム用とトランスクリプトーム用の違い1

- Sequencing depth情報の利用法
  - ゲノムの場合
    - （例えば）配列長の10倍読んだデータなら、平均的にゲノムのどの領域も10回程度読まれていると仮定される（10X coverage）
    - *k*-mer出現頻度分布に基づくエラー補正が原理的に可能（実際によく用いられる）
    - 多くのアセンブラはsequencing depth情報をリピート配列の認識に利用
  - トランスクリプトーム（RNA-seq）の場合
    - 転写物ごとに大きく異なる（低発現転写物はlow coverage, 高発現転写物はhigh coverage）
    - アセンブル前の段階でどの*k*-merがどの転写物由来かはわからないので、*k*-mer出現頻度の外れ値としてartifactsを除去する戦略は（低発現転写物がターゲットの場合には）不可能。ただし、low coverageなものはたとえ除去していなくてもアセンブルされにくい。。。

ゲノム用とトランスクリプトーム用の違い2

- ストランド情報
  - ゲノムの場合
    - +鎖と-鎖の両方がsequenceされるため、heterozygosity（対立遺伝子の塩基配列が異なる）がある場合にアセンブルがうまくいかない
  - トランスクリプトーム（RNA-seq）の場合
    - 昔は（Illuminaの場合）strand specificityはなかったが、最近のIlluminaは「ストランド（方向性）」の情報を利用可能
- アイソフォーム（isoforms or transcript variants）の存在
  - ゲノムの場合は気にする必要はない。
  - ある遺伝子領域（ORF）から全ての可能な転写物（transcripts）をRNA-seqデータのみから再構築するのは困難

比較トランスクリプトーム解析の流れ

複数のFASTQファイル

↓

リファレンス配列の作成

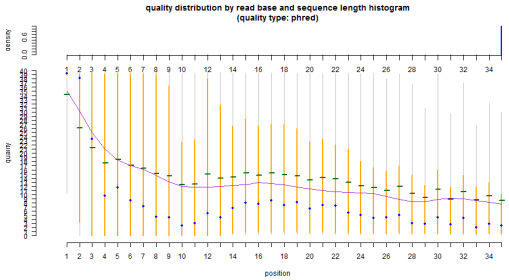
↓

マッピング

↓

データ解析

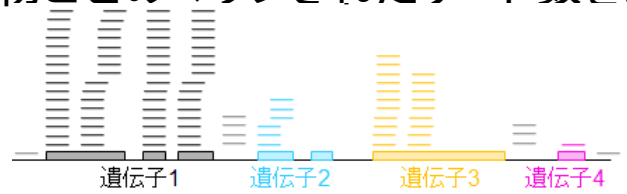
クオリティチェック



アセンブル結果（multi-fasta）ファイルから平均長やトータルの長さなどの基本情報を抽出

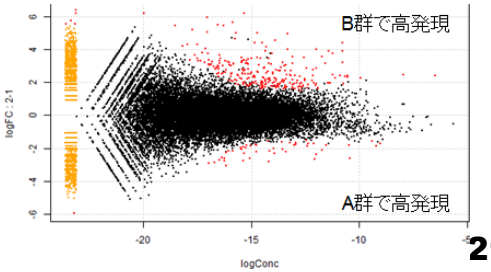
Total length (bp)	2993
Number of contigs	4
Average length	748.3
Median length	784
Max length	888
Min length	537
N50	886
GC content	0.524

マッピング結果（BED形式）ファイルを入力として、転写物ごとのマップされたリード数をカウント



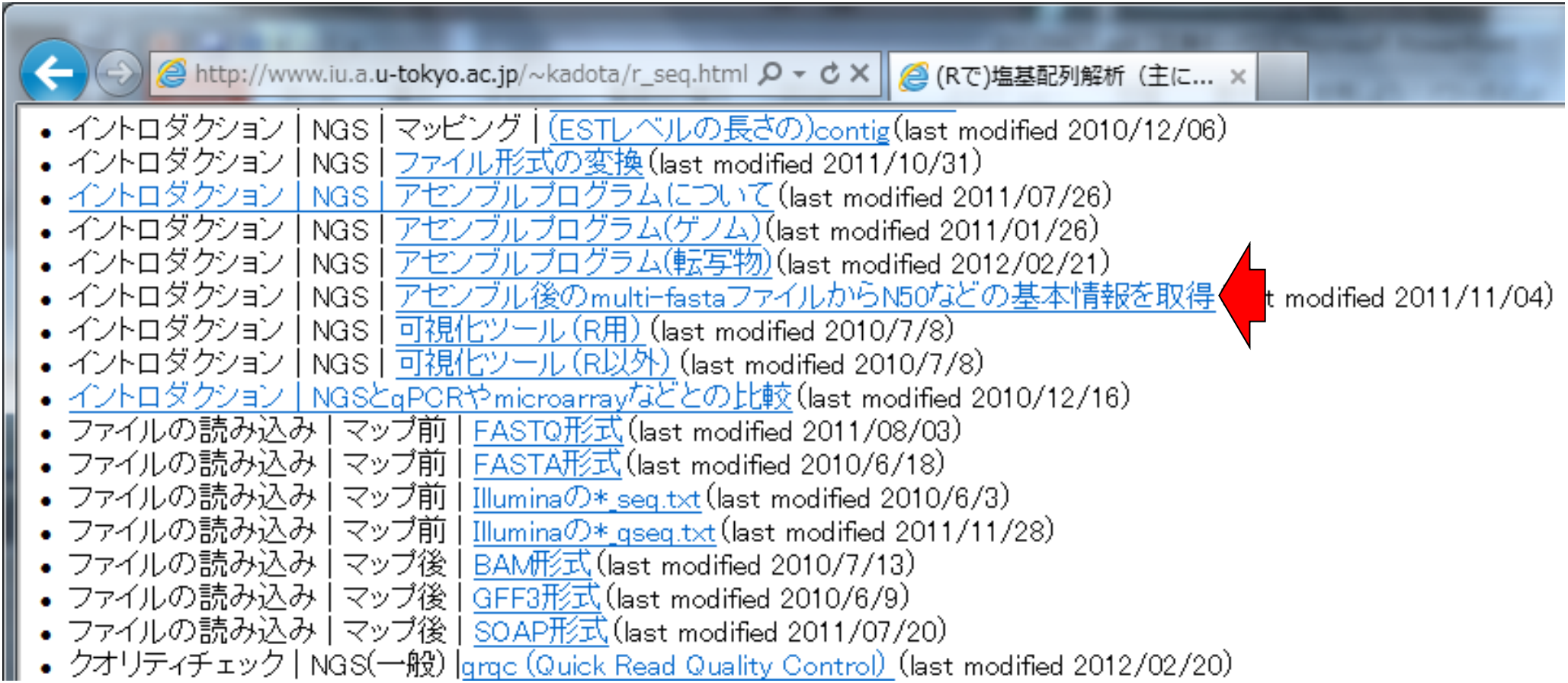
遺伝子1	40
遺伝子2	6
遺伝子3	20
遺伝子4	1

発現変動遺伝子のリストアップや、作図など



大規模計算部分以外は一通りできます

multi-fastaファイルからの各種情報抽出



multi-fastaって何？

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
```

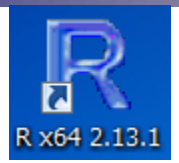
hoge4.fa - メモ帳

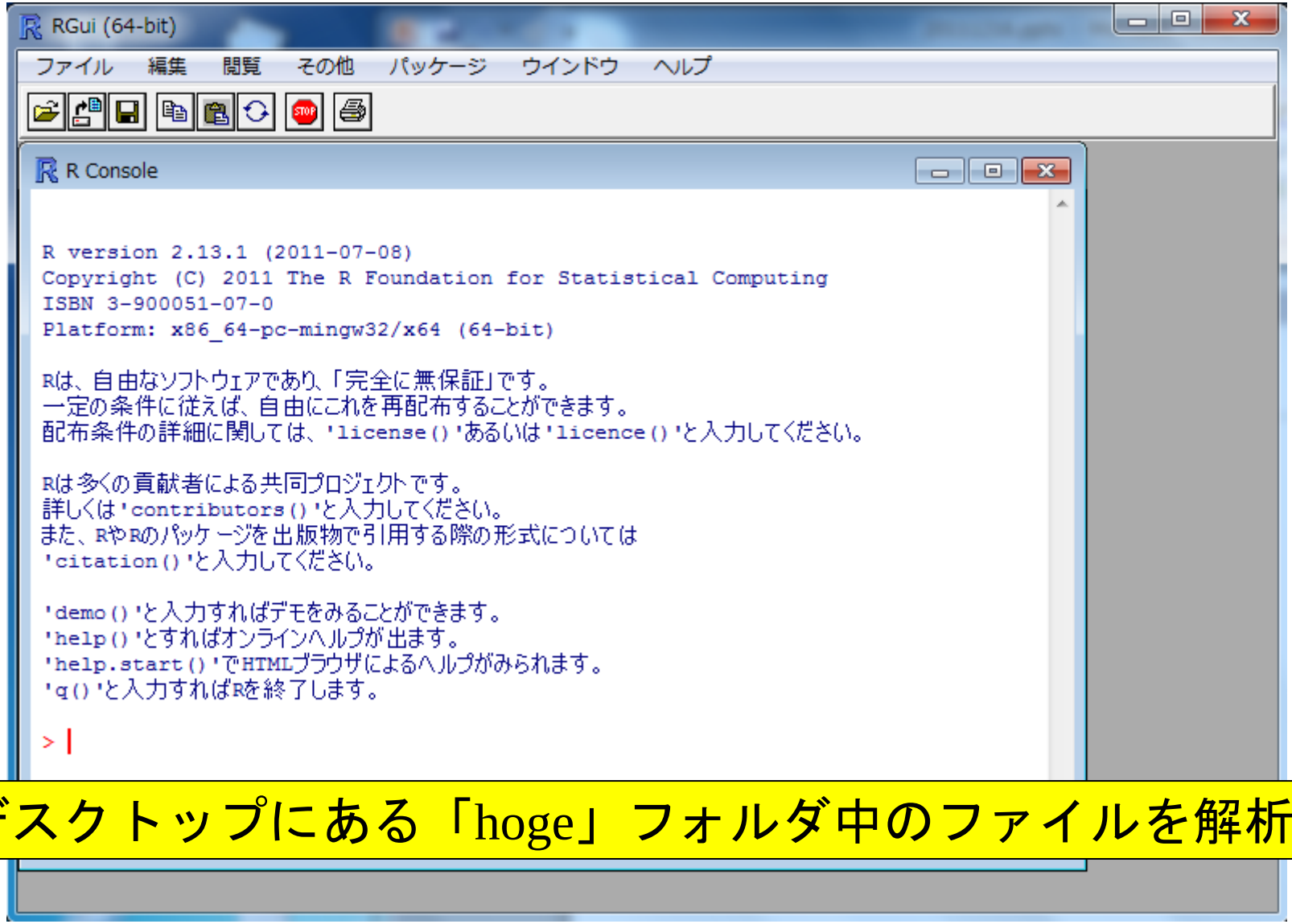
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
>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

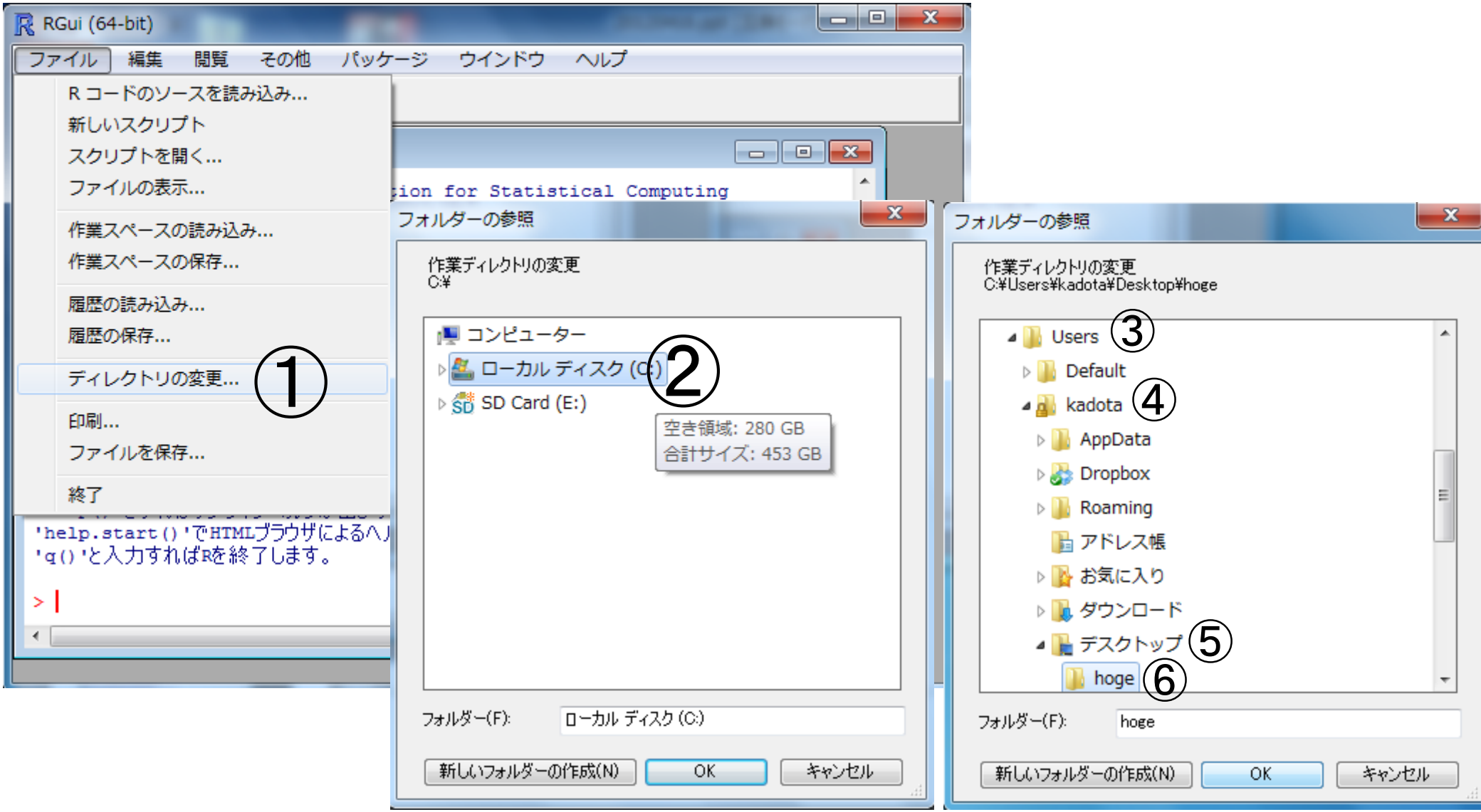
Rの起動



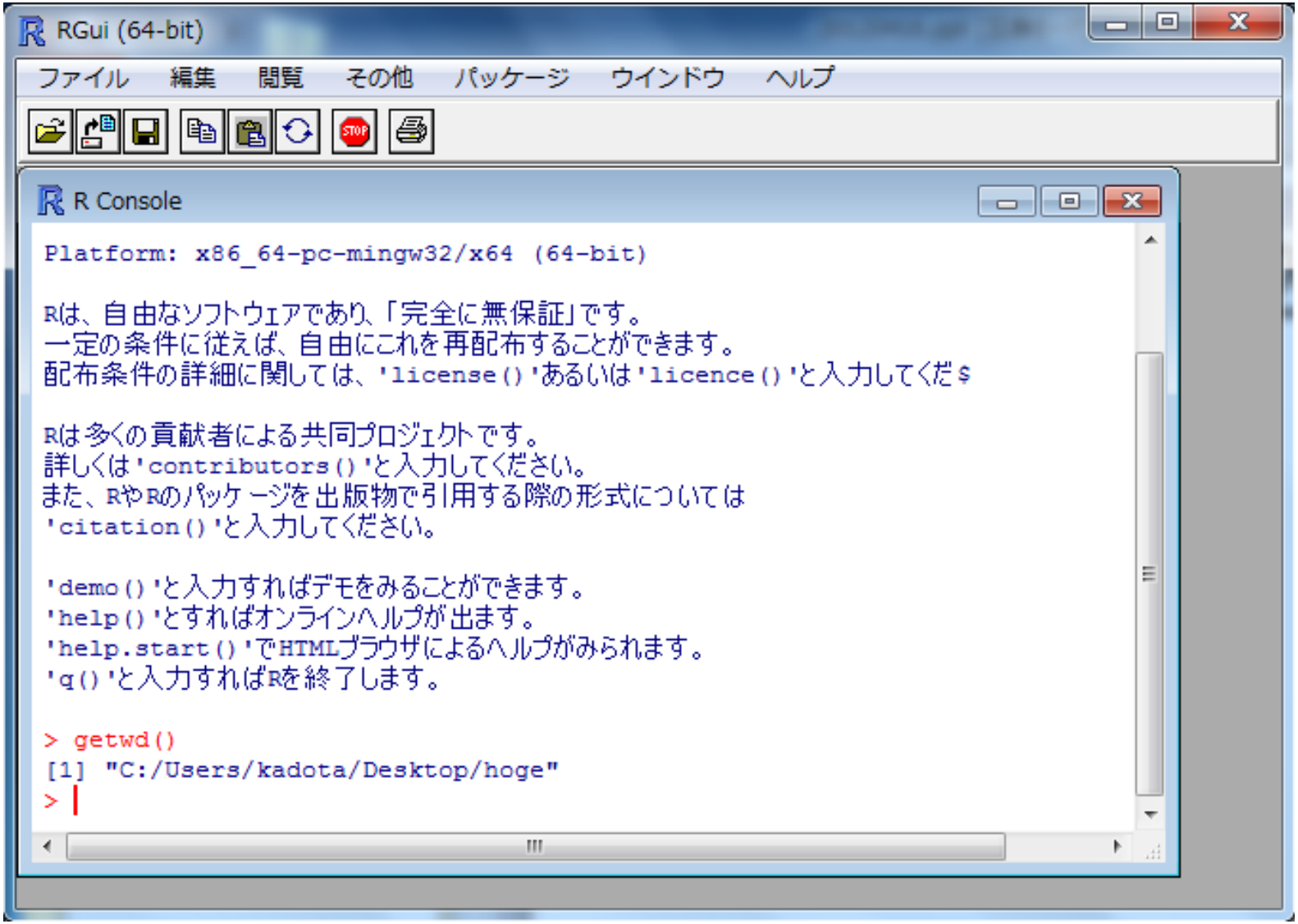


デスクトップにある「hoge」フォルダ中のファイルを解析

作業ディレクトリの変更



「getwd()」と打ち込んで確認



コピー＆ペースト

• イントロダクション | NGS | アセンブル後のmulti-fastaファイルからN50などの基本情報を取得

アセンブルプログラムを実行して得られるmulti-fastaファイルを読み込んで、Total lengthやaverage leng
行うためのやり方を示します。ここでは以下の二つを例題として行います:
1.「イントロダクション | 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(
2. 130MB程度のmulti-fastaファイル(h_rna.fasta)
「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

1. hoge4.faファイルの場合:

```
------    ここから    ------
in_f <- "hoge4.fa"
out_f <- "hoge1.txt"

library(Biostrings)
reads <- read.DNAStringSet(in_f, format="fasta")

Total_length <- sum(width(reads))
Number_of_contigs <- length(reads)
Average_length <- mean(width(reads))
Median_length <- median(width(reads))
Max_length <- max(width(reads))
Min_length <- min(width(reads))

#N50計算のところ
sorted_length <- rev(sort(width(reads)))
N50 <- sorted_length[cumsum(sorted_length) >= Total_length/2

#GC含量(GC content)計算のところ
count <- alphabetFrequency(reads)
CG <- rowSums(count[,2:3])
ACGT <- rowSums(count[,1:4])
GC_content <- sum(CG)/sum(ACGT)

#出力用に結果をまとめている
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))
```

①

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)計算のところ
> count <- alphabetFrequency(reads)
> CG <- rowSums(count[,2:3])
> ACGT <- rowSums(count[,1:4])
> GC_content <- sum(CG)/sum(ACGT)
>
> #出力用に結果をまとめている
> 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 <- rbind(
> tmp <- rbind(
> write.table(t
>
>
```

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

• イントロダクション | NGS | アセンブル後のmulti-fastaファイルからN50などの基本情報を取得

アセンブルプログラムを実行して得られる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。ここでは以下の二つを例題として行います:
1.「イントロダクション | 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)
2. 130MB程度のmulti-fastaファイル(h_rna.fasta)
「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. hoge4.faファイルの場合:

```
------    ここから    ------
in_f <- "hoge4.fa"                                  #multi-fasta形式の入力ファイルを指定
out_f <- "hoge1.txt"                                #出力ファイル名を指定
```

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
```

```
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))
```

hoge1.txt

ファイル ホー 挿入 ペー 数式

F14

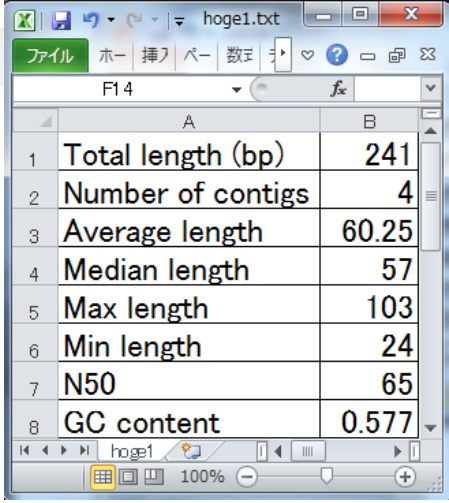
	A	B
1	Total length (bp)	241
2	Number of contigs	4
3	Average length	60.25
4	Median length	57
5	Max length	103
6	Min length	24
7	N50	65
8	GC content	0.577

hoge1

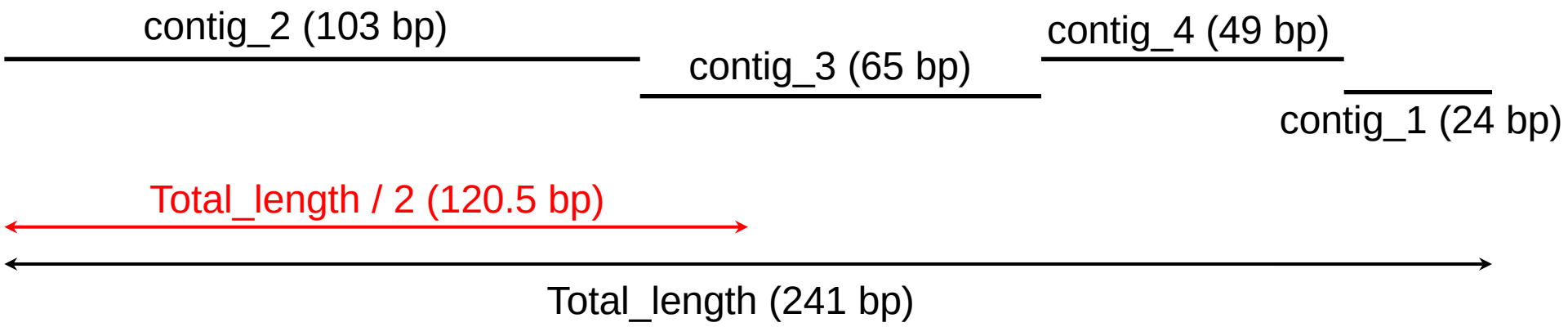
「N50」って何？

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



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。

multi-fastaファイルからの各種情報抽出

```
1. hoge4.faファイルの場合:
------    ここから    ------
in_f <- "hoge4.fa"                                    #multi-fasta形式の入力ファイルを指定
out_f <- "hoge1.txt"                                  #出力ファイル名を指定

library(Biostrings)                                   #パッケージの読み込み
reads <- read.DNAStringSet(in_f, format="fasta")      #in_fで指定したファイルをFASTA形式で読み込み

Total_length <- sum(width(reads))                     #コンティグの「トータルの長さ」の情報を取得
Number_of_contigs <- length(rea
Average_length <- mean(width(re
Median_length <- median(width(r
Max_length <- max(width(reads))
Min_length <- min(width(reads))
```

R Console

```
> reads
  A DNAStringSet instance of length 4
    width seq                                               names     $
[1]    24 CGGACAGCTCCTCGGCATCCGGAT                          contig_1
[2]   103 GTCTGCCTCAAGCGCCCCAAG...GAGACCCAGCACACCCTGTC     contig_2
[3]    65 TGTAGGAGAAGGGCGGTATCA...TAATTGTATGAGGTCGGGCA     contig_3
[4]    49 CGTGCTGATTCCACACAGCAG...GACCTCTACCTATGAACATG     contig_4
>
> width(reads)
[1]  24 103  65  49
>
> sum(width(reads))
[1] 241
>
```

width関数を使えば配列長情報を取り出せるようだ

multi-fastaファイルからの各種情報抽出

R Console

```
>
> reads[1]
  A DNAStringSet instance of length 1
    width seq                                                names
[1]    24 CGGACAGCTCCTCGGCATCCGGAT                          contig_1
>
> reads[2]
  A DNAStringSet instance of length 1
    width seq                                                names
[1]   103 GTCTGCCTCAAGCGCCCCAAGTGGG...GGCAGAGACCCAGCACACCCTGTC contig_2
>
> width(reads) >= 50
[1] FALSE  TRUE  TRUE FALSE
>
> reads[width(reads) >= 50]
  A DNAStringSet instance of length 2
    width seq                                                names
[1]   103 GTCTGCCTCAAGCGCCCCAAGTGGG...GGCAGAGACCCAGCACACCCTGTC contig_2
[2]    65 TGTAGGAGAAGGGCGGTATCAGCGT...TTACTAATTGTATGAGGTCGGGCA contig_3
>
```

ID	Length
contig_1	24
contig_2	103
contig_3	65
contig_4	49

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!

- ファイルの読み込み | マップ後 | SOAP形式 (last modified 2011/07/20)
- クオリティチェック | NGS(一般)について (last modified 2012/08/02)
- クオリティチェック | NGS(一般) | qrqc (Quick Read Quality Control) (last modified 2012/02/20)
- フィルタリング | 一般 | fastqファイルからACGTのみからなる配列(重複あり)の抽出 (last modified 2012/07/13)
- フィルタリング | 一般 | fastqファイルからACGTのみからなる配列(重複なし)の抽出 (last modified 2012/07/13)
- フィルタリング | 一般 | multi-fastaファイルのdescription行の記述を整形 (last modified 2012/07/13)
- フィルタリング | 一般 | multi-fastaファイルからACGT以外の文字の出現数でフィルタリング (last modified 2012/07/13)
- フィルタリング | 一般 | multi-fastaファイルから指定した配列長のもののみ抽出 (last modified 2012/04/03)
- フィルタリング | 一般 | multi-fastaファイルから任意のサブセットを抽出 (last modified 2012/07/17)
- フィルタリング | NGS(miRNA) | アダプター配列除去0(基本的なところ) (last modified 2010/5/27)
- フィルタ

• フィルタリング | 一般 | multi-fastaファイルから指定した配列長のもののみ抽出

multi-fasta形式のファイルが手元にあったとして、任意の配列長のもののみ抽出するやり方を示します。
ここでは以下の二つを例題として行います:
1.「イントロダクション | 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)
2. 130MB程度のRefSeqのhuman mRNAのmulti-fastaファイル(h_rna.fasta)

「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. hoge4.faファイルの場合:

```
------    ここから    ------
in_f <- "hoge4.fa"                                  #multi-fasta形式の入力ファイルを指定
out_f <- "hoge1.fasta"                              #出力ファイル名を指定
param <- 50                                         #配列長の閾値を指定

#必要なパッケージなどをロード
library(Biostrings)

reads <- readDNAStringSet(in_f, format="fasta")     #in_fで指定したファイルをFASTA形式で読み込
reads                                               #今現在のreadsオブジェクトを表示
reads <- reads[width(reads) >= param]               #paramで指定した配列長以上のもののみ抽出し
reads                                               #今現在のreadsオブジェクトを表示
writeXStringSet(reads, file=out_f, format="fasta")  #out_fで指定したファイル名でreadsというオ

------    ここまで    ------
```

指定した配列長以上のものを抽出できます

- イントロダクション | NGS | アノテーション情報取得(BioMart and biomaRt)(last modified 2011/08/26)
- イントロダクション | 一般 | 任意のキーワードを含む行を抽出(last modified 2012/03/29) NEW
- イントロダクション | 一般 | ランダムな塩基配列を作成(last modified 2012/04/02) NEW
- イントロダクション | 一般 | 配列取得(last modified 2010/7/7)
- イントロダクション | 一般 | 指定した範囲の配列を取得(last modified 2012/04/03) NEW
- イントロダクション | 一般 | 翻訳配列(translate)を取得(last modified 2011/07/27)

• イントロダクション | 一般 | 指定した範囲の配列を取得

1.では、12塩基(AGTGACGGTCTT)からなる一つの塩基配列(description行が">kadota")からなるFASTA形式ファイル(sample1.fasta)を入力として、この塩基配列の任意の範囲 (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(tmp1.fasta)に出力するやり方を示します。
2.では、RefSeqのhuman mRNAのmulti-fasta形式のファイル (h_rna.fasta)が手元にあ
の任意の範囲 (例:始点が2, 終点が5)の配列の抽出を行います。
3.では、2.の延長線上で、目的のaccession番号が複数ある場合に対応したものです。
start位置, 3列目:end位置」からなるリストファイル (list_sub1.txt)を読み込ませて、目的
を示します。
ここでは、得られた部分配列をFASTA形式ファイル(ファイル名:tmp3.fasta)で保存す
4.では、3.と同じことを異なるファイル(multi-fastaファイル:hoge4.fa、リストファイル;list

「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

入力ファイル:sample1.fasta

```
>kadota
AGTGACGGTCTT
```

出力ファイル:tmp1.fasta

```
>kadota
TGACGGT
```

1. sample1.fastaファイルの場合:

```
------    ここから    ------
in_f <- "sample1.fasta"                          #multi-fasta形式
out_f <- "tmp1.fasta"                            #出力ファイル名を
param <- c(3, 9)                                 #抽出したい範囲の始点と終点を指定

library(Biostrings)                              #パッケージの読み込み
reads <- read.DNAStringSet(in_f, format="fasta") #in_fで指定したファイルをFASTA形式で読み込み
out <- subseq(reads, 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てoutに格納
write.XStringSet(out, file=out_f, format="fasta", width=80)  #outの中身をout_fで指定したファイル名で保存

------    ここまで    ------
```

1. sample1.fastaファイルの場合:

```
------    ここから    ------
in_f <- "sample1.fasta"
out_f <- "tmp1.fasta"
param <- c(3, 9)

library(Biostrings)
reads <- read.DNAStringSet(in_f, format="fasta")
out <- subseq(reads, param[1], param[2])
write.XStringSet(out, file=out_f, format="fasta", width=80)

------    ここま
```

入力ファイル:sample1.fasta

```
>kadota
AGTGACGGTCTT
```

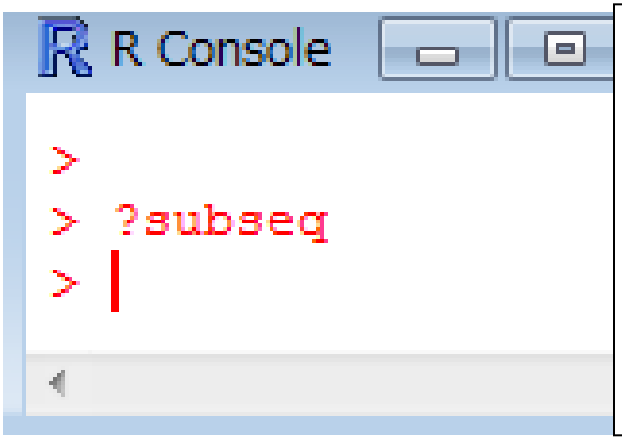
出力ファイル:tmp1.fasta

```
>kadota
TGACGGT
```

```
> reads
  A DNAStringSet instance of length 1
    width seq                                             names
[1]    12 AGTGACGGTCTT                                    kadota
> out
  A DNAStringSet instance of length 1
    width seq                                             names
[1]     7 TGACGGT                                         kadota
> param
[1] 3 9
> param[1]
[1] 3
> param[2]
[1] 9
> subseq(reads, 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」という形式で使うようだ

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

R Console

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

入力ファイル：sample1.fasta

```
>kadota
AGTGACGGTCTT
```

出力ファイル：tmp1.fasta

```
>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. hoge4.faファイルで抽出したい情報をリストファイル（list_sub2.txt）で与える場合:

```
------    ここから    ------
in_f1 <- "hoge4.fa"                    #multi-fasta形式のファイルを指定
in_f2 <- "list_sub2.txt"               #リストファイルを指定
out_f <- "tmp4.fasta"                  #出力ファイル名を指定

library(Bio
reads <- re                                              で読み込み
list_obj <-

out <- NULL                                              レースホル
for(i in 1:
    obj <-                                               致する位置を
    out <-                                               てobjがTRUE
}
write.XStri                                              名で保存

------
```

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
```

list_sub2.txt

contig_4	1	10
contig_2	3	8

出力ファイル：tmp4.fasta

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

- 前処理 | 正規化(混合) | RPKM正規化(Mortazavi_2008) | ゲノムマップ後の
- 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 2012/04/13)
- 解析 | 一般 | GC含量 (GC contents) (last modified 2010/7/1)
- 解析 | 一般 | Sequence logos (Schneider_1990) (last modified 2012/06/27
- 解析 | 一般 | 上流配列解析 | Local Distribution of Short Sequences (LDS
- 解析 | 一般 | 上流配列解析 | Relative Appearance Ratio (RAR)解析 (Yan
- 解析 | NGS(RNA-seq) | その他 | Technical replicatesのデータがボアソン
- 解析 | NGS(RNA
- 解析 | NGS(RNA

- 解析 | 一般 | GC含量 (GC contents) (last modified 2010/7/1)

• 解析 | 一般 | GC含量 (GC contents)

GC含量 (GC contents)の計算の仕方を書きます。ここでは、ファイルの読み込み(FASTA形式)で読み込んだ250 readsからなるtest1.fastaファイルを入力として、250 readsの各配列に対して「description」「C,Gの総数」「A,C,G,Tの総数」「配列長」「%GC含量」をファイルに出力するやり方を例示します。
尚、ここでは%GC含量の計算を「CGの総数/ACGTの総数」で計算していますので、もしGC含量を計算したい配列中にNなどが含まれる場合でNなどを含めた「配列長」を分母にしたい場合にはGC含量を計算をする数式中の「CG/ACGT*100」を「CG/width(reads)*100」に変更してください。
「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。

```
------    ここから    ------
in_f <- "test1.fasta"                                   #読み込みたいFASTA形式のファイル名を指定してin_fに格納
out_f <- "hoge.txt"                                     #出力ファイル名を指定
library(Biostrings)                                     #パッケージの読み込み
reads <- read.DNAStringSet(in_f, format="fasta")        #in_fで指定したファイルの読み込み
count <- alphabetFrequency(reads)                       #A,C,G,T,..の数を各配列ごとにカウントした結果をcountに格納
CG <- rowSums(count[,2:3])                              #C,Gの総数を計算してCGに格納
ACGT <- rowSums(count[,1:4])                            #A,C,G,Tの総数を計算してACGTに格納
out <- CG/ACGT*100                                      #%GC含量を計算してoutに格納

tmp <- cbind(names(reads), CG, ACGT, width(reads), out)         #ファイルに出力したい情報を連結してtmpに格納
colnames(tmp) <- c("description", "CG", "ACGT", "Length", "%GC_contents")#列名情報を与えている
write.table(tmp, out_f, sep="¥t", append=F, quote=F, row.names=F, col.names=T)#tmpの中身をout_fで指定したファイル名で保存。

------    ここまで    ------
```

BioconductorのBiostringsのwebページ

配列ごとのGC含量を計算したいとき

比較トランスクリプトーム解析の流れ

複数のFASTQファイル

↓

リファレンス配列の作成

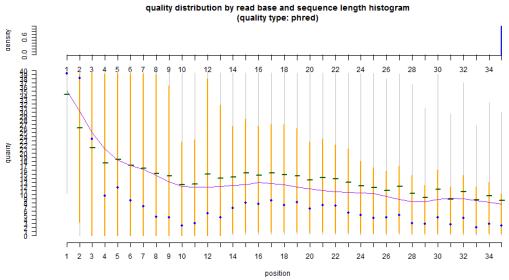
↓

マッピング

↓

データ解析

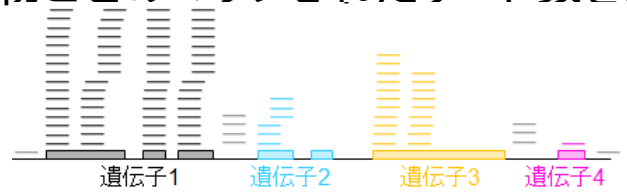
クオリティチェック



アセンブル結果（multi-fasta）ファイルから平均長やトータルの長さなどの基本情報を抽出

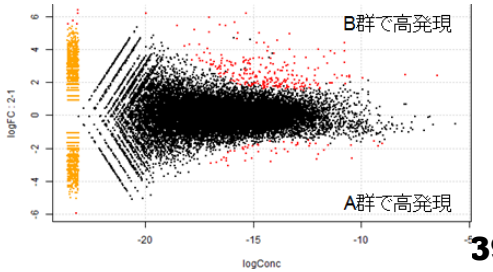
Total length (bp)	2993
Number of contigs	4
Average length	748.3
Median length	784
Max length	888
Min length	537
N50	886
GC content	0.524

マッピング結果（BED形式）ファイルを入力として、転写物ごとのマップされたリード数をカウント



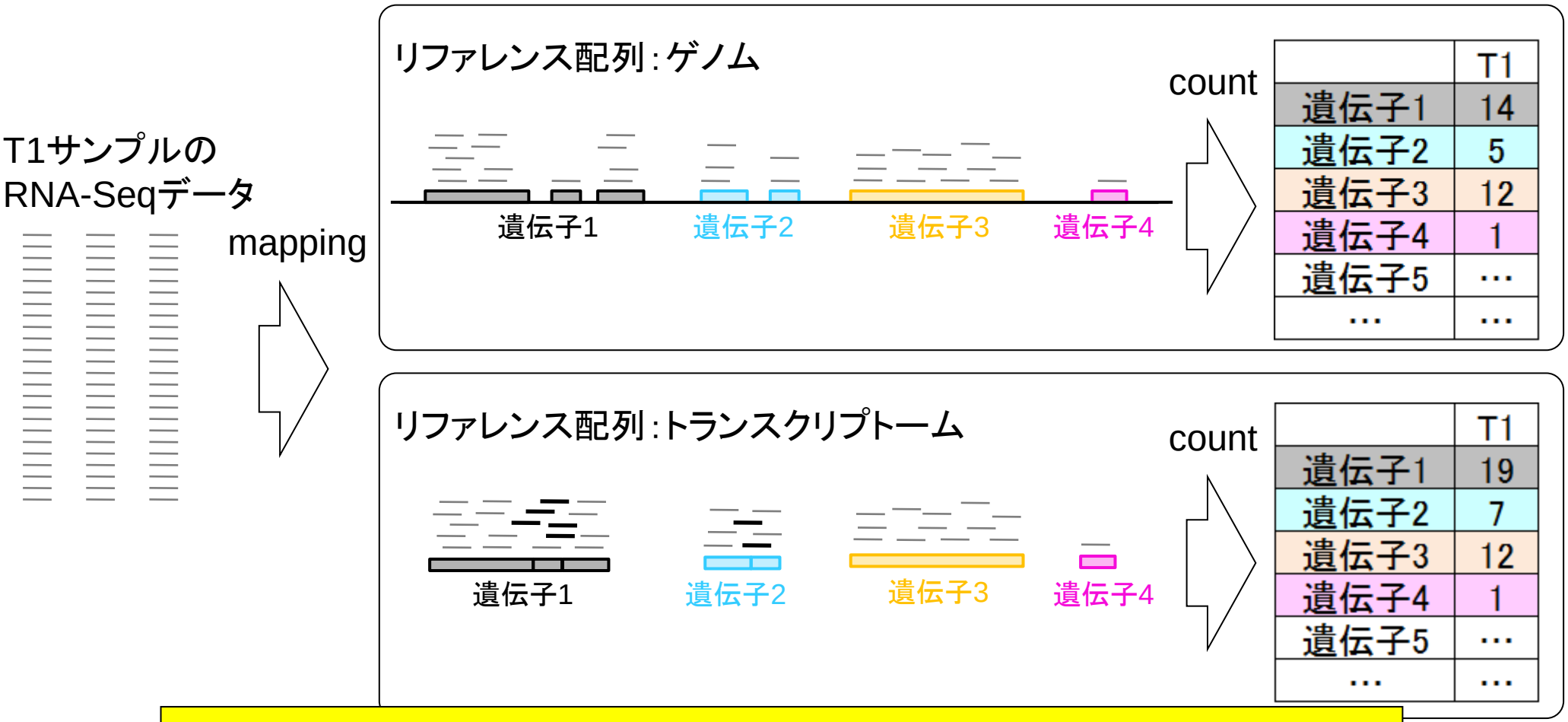
遺伝子1	40
遺伝子2	6
遺伝子3	20
遺伝子4	1

発現変動遺伝子のリストアップや、作図など



マッピングの基本的なイメージ

- 基本的なマッピングプログラム(basic aligner)を用いた場合

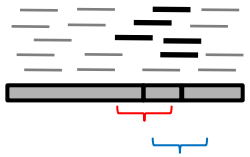


ゲノム配列へのマッピングの場合、複数のエクソンにまたがるリード(**spliced reads**)はマップされないので…

対策（リードが短かったころ；<50bp）

- 既知のsplice junction周辺配列を予め組み込んだリファレンスゲノム配列側にマッピング

遺伝子1



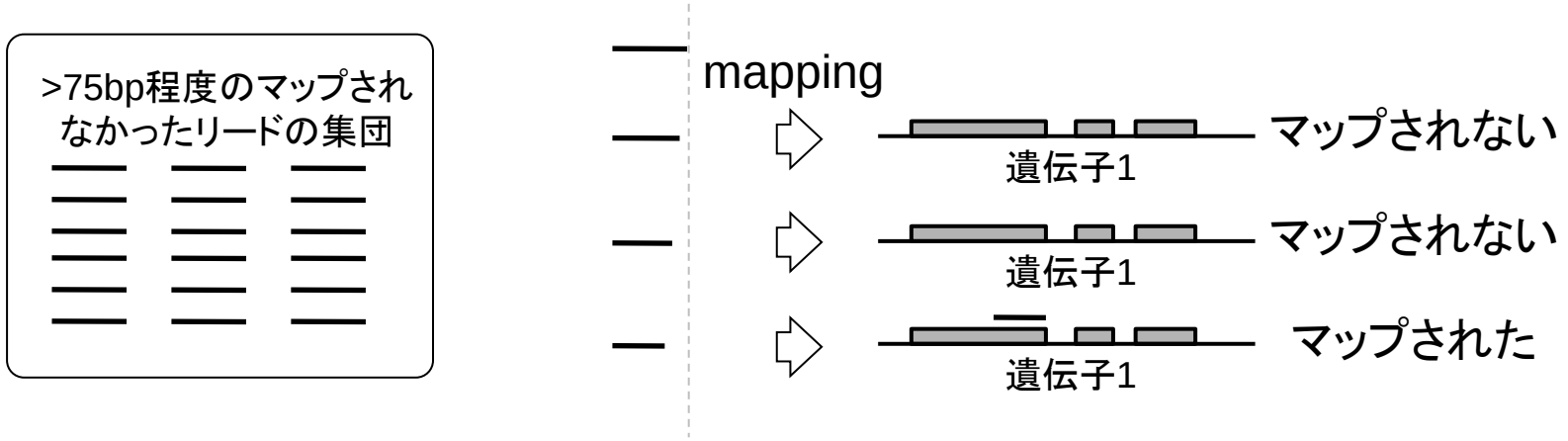
リファレンスゲノム配列への組み込み後のイメージ

```
>chr1
GGGGTTCAAAGCAGTATCGATCAAATAGTA
>chr2
GTTCAAAGCAGTATCGATCAAATAGTAAAT
…
>遺伝子1の「Exon1のend-20bp」から「exon2のstart+20bp」
ACGATGCAGCCTTAACGATGGTCCACAATT…
>遺伝子1の「Exon2のend-20bp」から「exon3のstart+20bp
```

（少なくとも）既知のexon間をまたぐリードのマッピングは可能

対策（ーリード>75bp程度の現在）

- 再帰的にマッピングする戦略（recursive mapping strategy）
  - （通常のマッピングプログラムでマップされなかったものに対して）リードを短くしてマップされるかどうか、を繰り返す



splice-aware alignerを用いることで（既知遺伝子構造情報を参照することもないので）新規アイソフォームの同定も可能

Splice-aware alignerの様々な戦略

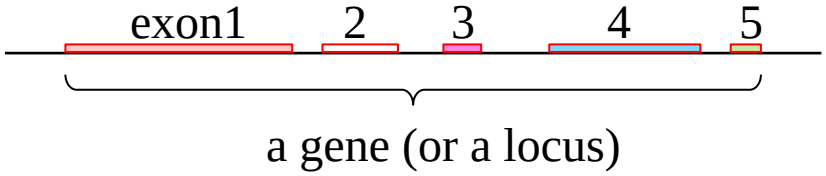
詳細を知りたいヒトは右上論文のReviewを参照

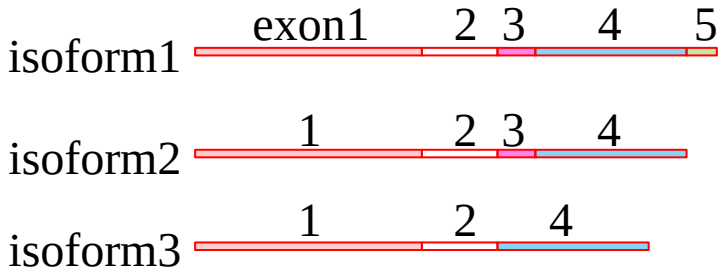
雑感

- 一口にトランスクリプトーム解析といっても目的や手段は多様。
  - 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（or isoform）ごとの発現量推定
    - 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含量biasなどの各種補正がポイント
    - 遺伝子レベルの発現量　⇔　isoformレベルの発現量の正確な推定
  - 比較するサンプル間で発現変動している遺伝子（or 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

定量化：遺伝子レベル ⇒ isoformレベル

- 復習
  - RNA-seqデータは転写物（transcripts）を断片化してsequencingしたもの
  - 一つの遺伝子領域（locus）から複数のsplice variants（or isoforms or transcripts）が生成されうる
    - 特定のisoformのみで使われているexonもあれば（例：isoform1だけがexon5を使っている）…
    - 転写されるすべてのisoformsで共通して使われているexonもある（例：exons 1, 2, and 4）し…
    - （以下の図にはないが…）特定のisoformのみで使われているexonがない場合もある…

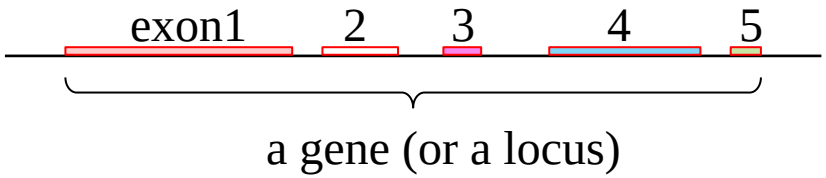


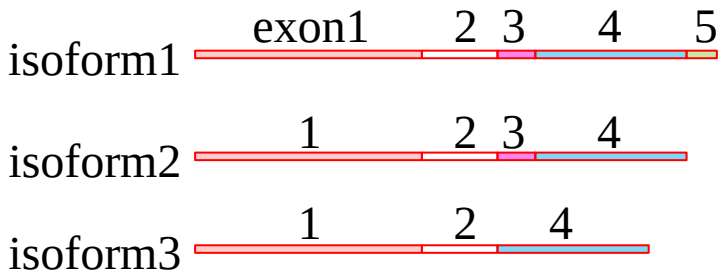


isoformレベルの定量化も様々な戦略があります

isoformレベルの定量化

- ALEXA-seqの場合
  - 戦略：そのisoformだけにマップされたリード数（unique reads）をカウント
  - 短所：unique exonを持たないisoformの定量化はできない

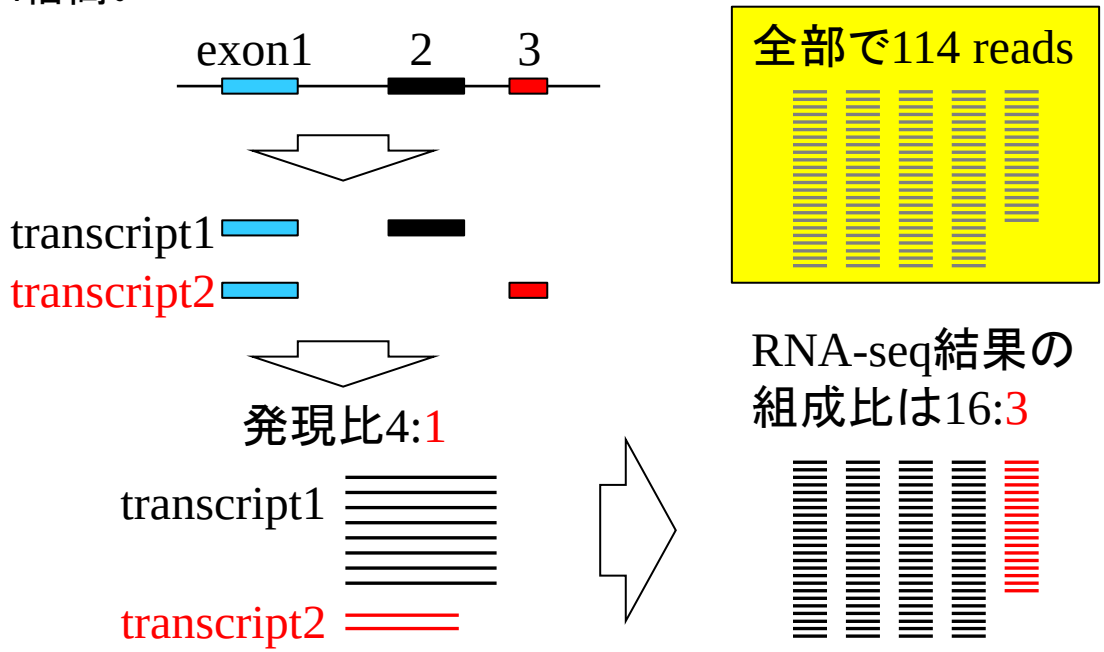




isoform2と3の定量化ができない

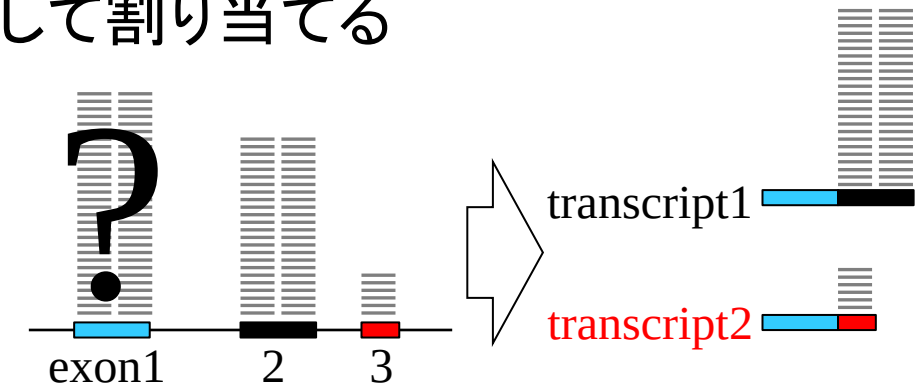
isoformレベルの定量化

- Cufflinks（Trapnell, 2010）やMISO（Katz, 2010）の場合
  - 戦略：複数のisoformsにマップされるリード（multi reads）の割り当てについて、それ以外のマップされたリードの（長さなどを考慮した）割合などを考慮して割り当てる
  - 説明のための仮定
    - ある遺伝子領域から二つの転写物ができている（transcript1とtranscript2）
    - 真実：発現レベルはtranscript1のほうが4倍高い
    - exon1,2,3の長さ比は2:2:1

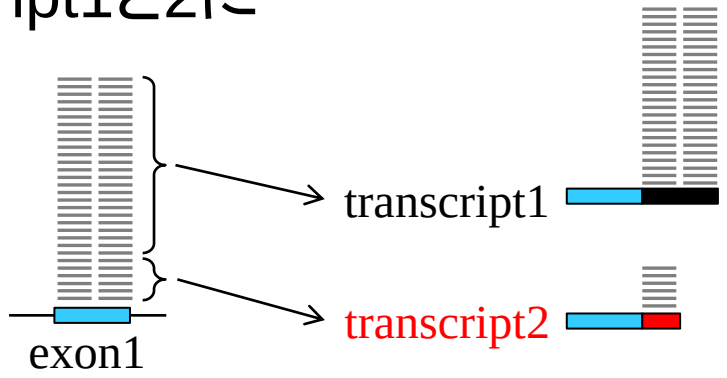


isoformレベルの定量化

- 戦略：複数のisoformsにマップされるリード（multi reads）の割り当てについて、**それ以外のマップされたリードの（長さなどを考慮した）割合**などを考慮して割り当てる
- マッピング結果
  - exon1にマップされたリード数：60 reads
  - exon2にマップされたリード数：48 reads
  - exon3にマップされたリード数：6 reads



- 問題：exon1にマップされた60 readsをtranscript1と2にどのように分配するか?
  - 「マップされたリード数の比（48 : 6 = 8 : 1）」は×
  - 「長さも考慮した比（48/2 : 6 = 4 : 1）」は○



exon1にマップされた60 readsの80%をisoform1へ、残りの20%をisoform2へ分配

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）
- 「今ここにあるデータ」を様々な視点で解析可能な解像度は…遺伝子レベルのデータ
  - 例：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

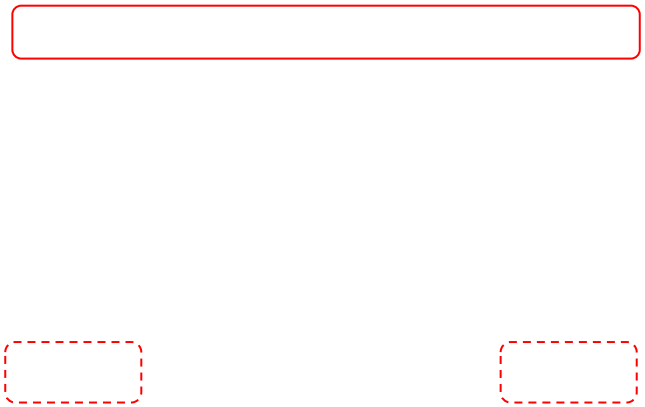
…

- 「isoformレベルの情報　→　遺伝子レベルの情報」への要約戦略
  - 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

論点：count情報を得る際に、どのexonの情報を用いるか？

どのexonのカウント情報を用いるか？

- 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



様々な思想があります…。当然、その後の解析結果に影響を及ぼします

発現レベルの定量化を行うプログラムたち

- （おそらく）ゲノムマップデータを入力とするもの
  - Scripture (Guttman *et al*., *Nat. Biotechnol*., **28**: 503-510, 2010)
  - Cufflinks (Trapnell et al., *Nat. Biotechnol.*, **28**: 511-515, 2010)
  - rQuant (Bohnert and Ratsch, *Nucleic Acids Res.*, **38**: W358-W351, 2010)
  - ALEXA-seq (Griffith *et al*., *Nat. Methods*, **7**: 843-847, 2010)
  - MISO (Katz *et al*., *Nat. Methods*, **7**: 1009-1015, 2010)
  - IsoformEx (Kim *et al*., *BMC Bioinformatics*, **12**: 305, 2011)
  - RSEM (Li and Dewey, *BMC Bioinformatics*, **12**: 323, 2011)
  - SLIDE (Li *et al*., *PNAS*, **108**: 19867-19872, 2011)
- （おそらく）トランスクリプトームマップデータを入力とするもの
  - NEUMA (Lee et al., *Nucleic Acids Res.*, **39**: e9, 2011)
  - IsoEM (Nicolae *et al*., *Algorithms Mol. Biol*., **6**: 9 2011)
  - RSEM (Li and Dewey, *BMC Bioinformatics*, **12**: 323, 2011)

Reference-based strategy

1. Splice-aware aligner（or spliced aligner）を用いてゲノムマッピング
   - BLAT (Kent WJ, *Genome Res*., **12**: 656-664, 2002)
   - QPALMA (De Bona *et al*., *Bioinformatics*, **24**: i174-i180, 2008)
   - TopHat (Trapnell *et al*., *Bioinformatics*, **25**: 1105-1111, 2009)
   - GSNAP (Wu *et al*., *Bioinformatics*, **26**: 873-881, 2010)
   - SpliceMap (Au *et al*., *Nucleic Acids Res*., **38**: 4570-4578, 2010)
   - MapSplice (Wang *et al*., *Nucleic Acids Res*., **38**: e178, 2010)
   - HMMSplicer (Dimon *et al*., *PLoS One*, **5**: e13875, 2010)
   - X-MATE (Wood *et al*., *Bioinformatics*, **27**: 580-581, 2011)
   - RNASEQR (Chen *et al*., *Nucleic Acids Res*., **40**: e42, 2012)
   - PASSion (Zhang *et al*., *Bioinformatics*, **28**: 479-486, 2012)
   - ContextMap (Bonfert *et al*., *BMC Bioinformatics*, **13 Suppl 6**: S9, 2012)

     これらのプログラム出力結果を利用して最終的な遺伝子構造を構築するのがCufflinksやScriptureなどのプログラム

   - TopHatとCufflinksを使って実際の作業を行うプロトコルもあります
     - Trapnell *et al*., *Nat. Protocols*, **7**: 562-578, 2012

Schbath *et al*., *J. Comput. Biol.*, **19**: 796-813, 2012

Basic alignerについて

- splice-aware aligner (spliced aligner)の多く?!は内部的にbasic aligner (unspliced aligner)を利用している。
- アルゴリズム的な観点から大きく二つに大別可能
  - Seed-and-extend methods
    - MAQ (Li *et al*., *Genome Res.*, **18**: 1851-1858, 2008)
    - SHRiMP2 (David *et al*., *Bioinformatics*, **27**: 1011-1012, 2011)
    - …
  - Burrows-Wheeler transformation (BWT) methods
    - Bowtie (Langmead *et al*., *Genome Biol*., **10**: R25, 2009)
    - BWA-SW (Li and Durbin, *Bioinformatics*, **26**: 589-595, 2010)
    - …

BWT系はmismatchやindelに弱いが速い、などの特徴があったが、両者ともに改良されている模様。昔ながらのプログラムの結果が不満なら最新のプログラムを試してみるのもありだろう。

Splice-aware alignerの様々な戦略

TopHat
SpliceMap
MapSplice
…

BLAT
QPALMA
GSNAP
…

比較トランスクリプトーム解析の流れ

複数のFASTQファイル

↓

リファレンス配列の作成

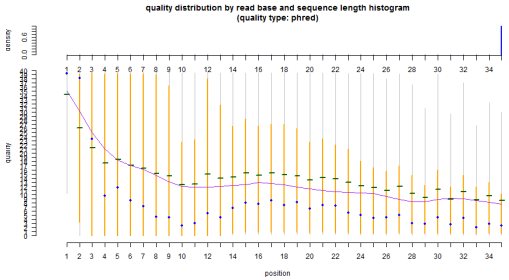
↓

マッピング

↓

データ解析

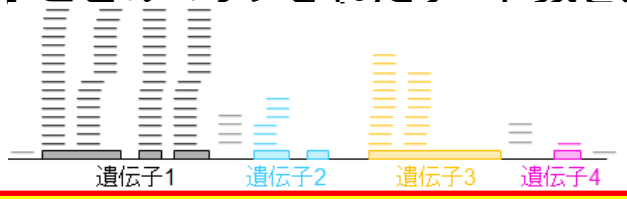
クオリティチェック



アセンブル結果（multi-fasta）ファイルから平均長やトータルの長さなどの基本情報を抽出

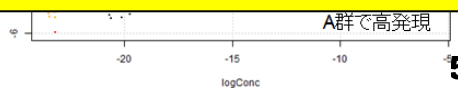
Total length (bp)	2993
Number of contigs	4
Average length	748.3
Median length	784
Max length	888
Min length	537
N50	886
GC content	0.524

マッピング結果（BED形式）ファイルを入力として、遺伝子ごとのマップされたリード数をカウント



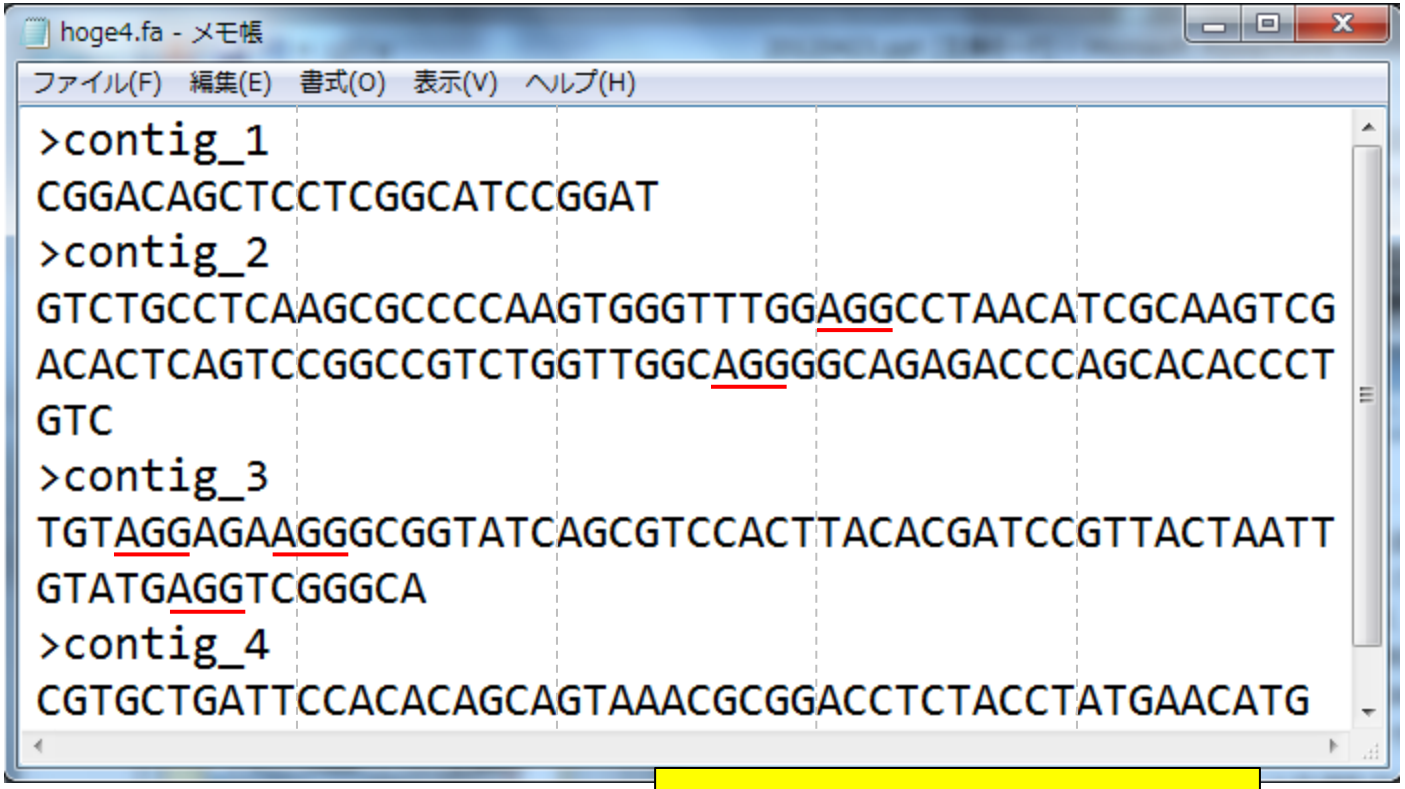
遺伝子1	40
遺伝子2	6
遺伝子3	20
遺伝子4	1

発現変動解析の入力データとして用いる「遺伝子発現行列」中の数値は一意に決まるわけではない（様々なバリエーションがあります）



マッピング　＝　（大量高速）文字列検索

- マップされる側の配列：4コンティグ（or 4遺伝子 or 4染色体）
- マップする側のNGS由来塩基配列データ：”AGG”



出力ファイル：hoge2.txt

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

Rでやってみよう

パターンマッチング

- 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 2012/04/13) NEW
- 解析 | 一般 | GC含量 (GC content) (last modified 2010/7/1)
- 解析 | 一般 | Sequence logos (Schneider_1990) (last modified 2012/04/04) NEW
- 解

• 解析 | 一般 | パターンマッチング

読み込んだリファレンス配列(reference sequence or subject sequence)から(短い)配列パターンを探す場合に利用します。
ここでは、以下の5つの例題を行います:

1. D
Finge
2. 1. と同
3. 2. と同
4. multi-
5. multi-
6. multi-
7. multi-
(記述の

4. multi-fastaファイルhoge4.faを入力として、"AGG"でキーワード探索を行う場合：

```
------    ここから    ------
in_f <- "hoge4.fa"                                          #読み込みたいFASTA形式のファイル名を指
out_f <- "hoge2.txt"                                        #出力ファイル名を指定してout_fに格納
param <- "AGG"                                              #調べたい配列パターンを指定してparam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
seq <- read.DNAStringSet(in_f, format="fasta")              #in_fで指定したファイルの読み込み

#本番
out <- vmatchPattern(pattern=param, subject=seq)            #paramで指定した配列と100%マッチの領域
hoge <- cbind(start(unlist(out)), end(unlist(out)))         #一致領域の(start, end)の位置情報をhog
colnames(hoge) <- c("start", "end")                         #行列hogeに列名を付加
rownames(hoge) <- names(unlist(out))                        #行列hogeに行名を付加
tmp <- cbind(rownames(hoge), hoge)                          #ファイルに出力したい情報を連結してtmp
write.table(tmp, out_f, sep="¥t", append=F, quote=F, row.names=F)#tmpの中身をout_fで指定したファイル名

------    ここまで    ------
```

基本はコピペ

```
4. multi-fastaファイルhoge4.faを入力として、"AGG"でキーワード探索を行う場合：
------    ここから    ------
in_f <- "hoge4.fa"                          #読み込みたいFASTA形式の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
param <- "AGG"                              #調べたい配列パターンを指定してparamに格納

#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
seq <- read.DNAStringSet

#本番
out <- vmatchPattern(pat
hoge <- cbind(start(unli
colnames(hoge) <- c("sta
rownames(hoge) <- names(
tmp <- cbind(rownames(ho
write.table(tmp, out_f,

------    ここまで    --
```

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
すべてのアクセラレー

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

```
> getwd()
[1] "C:/Users/kadota_DELL/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
ウィンドウを常にトップに置く

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

```
> library(Biostrings)                                          #パッケ
>
> #入力ファイルの読み込み
> seq <- read.DNAStringSet(in_f, format="fasta")               #in_f
>
> #本番
> out <- vmatchPattern(pattern=param, subject=seq)             #para
> hoge <- cbind(start(unlist(out)), end(unlist(out)))          #一致
> colnames(hoge) <- c("start", "end")                          #行列
> rownames(hoge) <- names(unlist(out))                         #行列
> tmp <- cbind(rownames(hoge), hoge)                           #ファイ
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)#tmpの
>
>
```

hoge2.txt - M.

ファイル　ホー　挿入　ペー　数式　デー

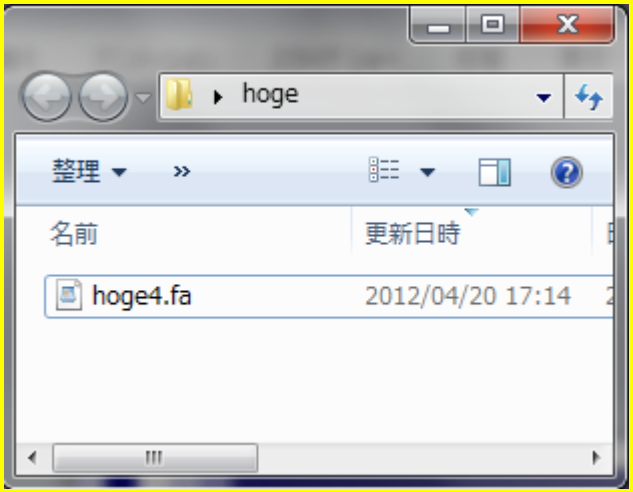
E12

	A	B	C	D
1		start	end	
2	contig_2	31	33	
3	contig_2	77	79	
4	contig_3	4	6	
5	contig_3	10	12	
6	contig_3	56	58	
7				

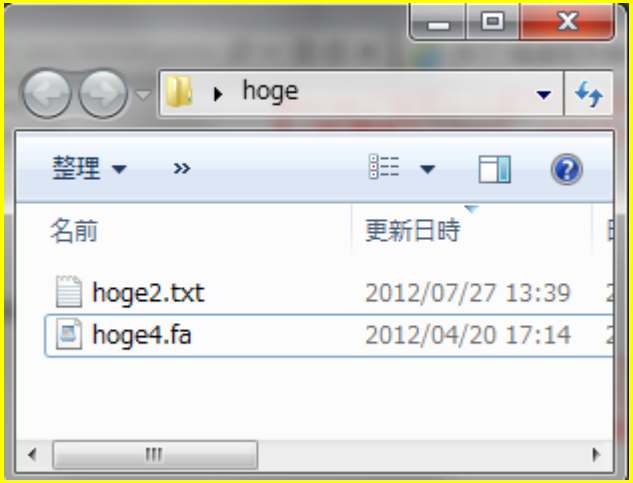
hoge2

コマンド　100%

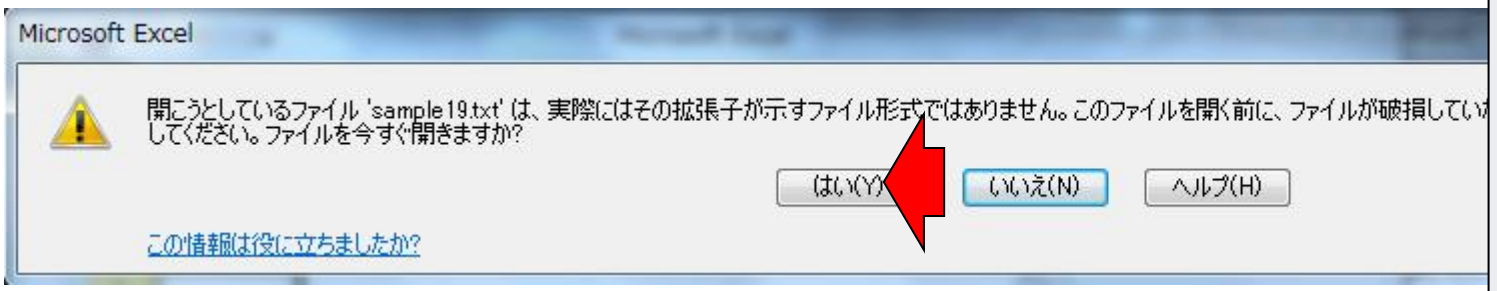
実行前のhogeフォルダ

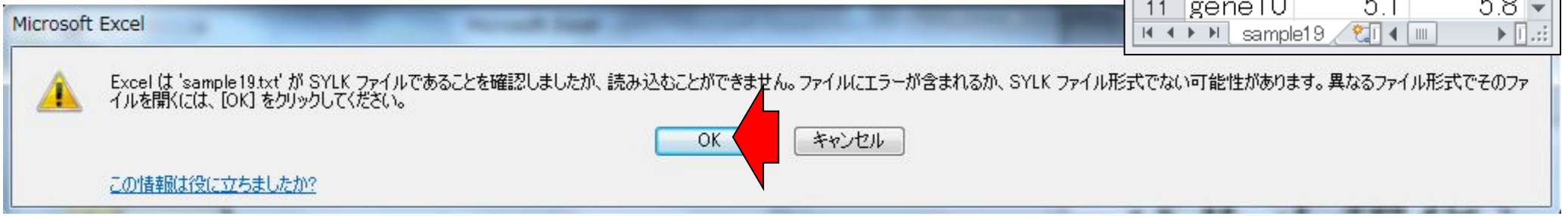


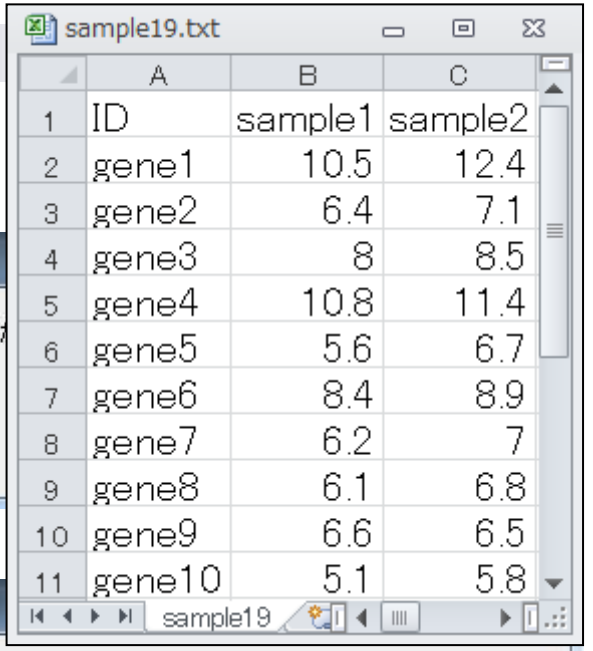
実行後のhogeフォルダ



エクセルで開くとき…

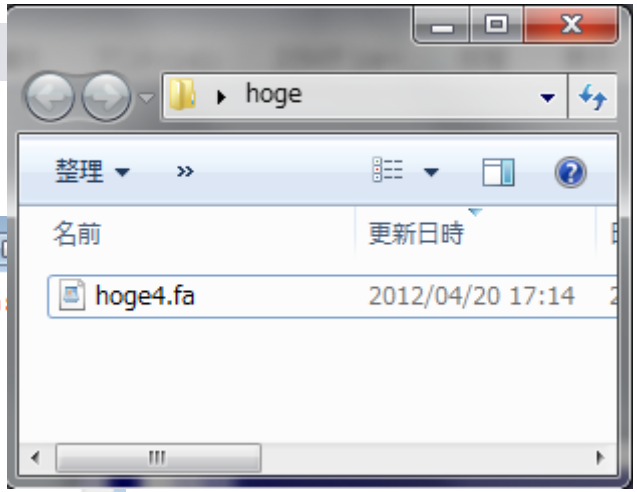






（ドラッグ＆ドロップで開こうとすると）エラーが出て一回目は開けないことがあるが、その後もう一度同じ作業を繰り返すと開けます…

ありがちなミス1

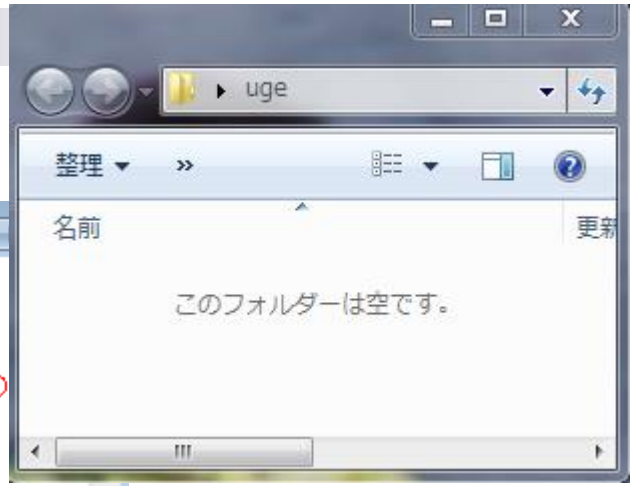


R Console

```
> in_f <- "hoge4.fa"                                    #読み込みたいFASTA形式の
> out_f <- "hoge2.txt"                                  #出力ファイル名を指定し$
> param <- "AGG"                                        #調べたい配列パターンを$
>
> #必要なパッケージをロード
> library(Biostrings)                                   #パッケージの読み込み
 要求されたパッケージ BiocGenerics をロード中です

 次のパッケージを付け加えます: ''BiocGenerics''

The following object(s) are masked from 'package:stats':

    xtabs

The following object(s) are masked from 'package:base':

    anyDuplicated, cbind, colnames, duplicated, eval, Filter, Find, get,
    intersect, lapply, Map, mapply, mget, order, paste, pmax, pmax.int, pmin,
    pmin.int, Position, rbind, Reduce, rep.int, r
    tapply, union, unique

 要求されたパッケージ IRanges をロード中です
>
> #入力ファイルの読み込み
> seq <- read.DNAStringSet(in_f, format="fasta")        #in_fで指定したファイル$
 以下にエラー .Call2("new_input_ExternalFilePtr", fp, PACKAGE = "Biostrings") :
  cannot open file 'hoge4.fa'
>
> #本番
> out <- vmatchPattern(pattern=param, subject=seq)      #paramで指定した配列と10$
 以下にエラー function (classes, fdef, mtable)  :
  unable to find an inherited method for function "vmatchPattern", for signature "function"
```

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

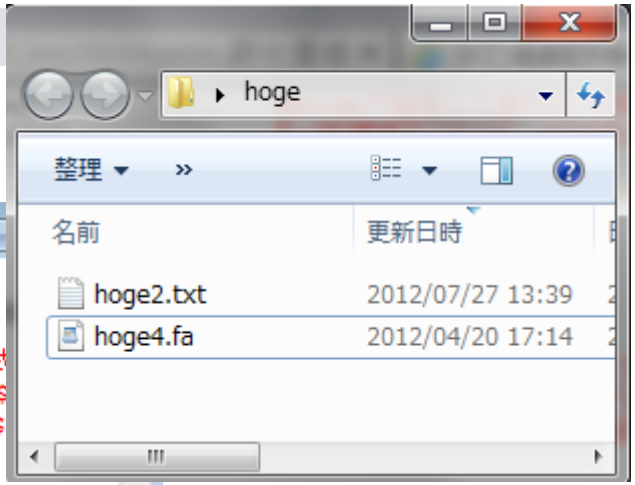
```
>
> getwd()
[1] "C:/Users/kadota_DELL/Desktop/uge"
> in_f <- "hoge4.fa"                                    #読み込みたいFASTA形式の
> out_f <- "hoge2.txt"                                  #出力ファイル名を指定し$
> param <- "AGG"                                        #調べたい配列パターンを$
>
> #必要なパッケージをロード
> library(Biostrings)                                   #パッケージの読み込み
>
> #入力ファイルの読み込み
> seq <- read.DNAStringSet(in_f, format="fasta")        #in_fで指定したファイル$
 以下にエラー .Call2("new_input_ExternalFilePtr", fp, PACKAGE = "Biostrings") :
  cannot open file 'hoge4.fa'
>
> #本番
> out <- vmatchPattern(pattern=param, subject=seq)      #paramで指定した配列と10$
 以下にエラー function (classes, fdef, mtable)  :
  unable to find an inherited method for function "vmatchPattern", for signature "function"
> hoge <- cbind(start(unlist(out)), end(unlist(out)))   #一致領域の(start, end)$
 以下にエラー start(unlist(out)) :
   引数 'x' の評価中にエラーが起きました (関数 'start' に対するメソッドの選択時):  以下にエ$
   引数 'x' の評価中にエラーが起きました (関数 'unlist' に対するメソッドの選択時):  エラー$


> colnames(hoge) <- c("start", "end")                   #行列hogeに列名を付加
 以下にエラー colnames(hoge) <- c("start", "end") :
   オブジェクト 'hoge' がありません
> rowname                                               #行列hoge
 以下にエラ
   引数 '
```

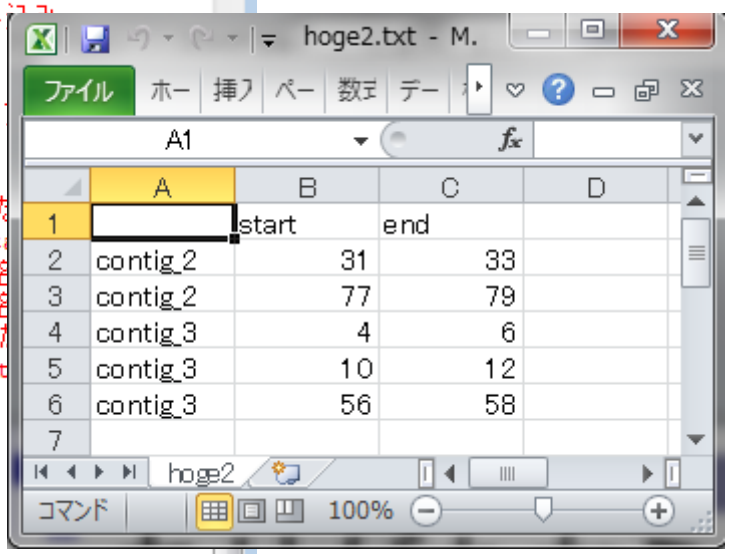
必要な入力ファイルが作業ディレクトリ中に存在しない…

ありがちなミス3

R Console

```
> getwd()
[1] "C:/Users/kadota_DELL/Desktop/hoge"
> in_f <- "hoge4.fa"                                    #読み込みたいFASTA形式
> out_f <- "hoge2.txt"                                  #出力ファイル名を指定し$
> param <- "AGG"                                        #調べたい配列パターンを$
>
> #必要なパッケージをロード
> library(Biostrings)                                   #パッケージの読み込
>
> #入力ファイルの読み込み
> seq <- read.DNAStringSet(in_f, format="fasta")        #in_fで指定した
>
> #本番
> out <- vmatchPattern(pattern=param, subject=seq)      #paramで指定した
> hoge <- cbind(start(unlist(out)), end(unlist(out)))   #一致領域の(st
> colnames(hoge) <- c("start", "end")                   #行列hogeに列名
> rownames(hoge) <- names(unlist(out))                  #行列hogeに行名
> tmp <- cbind(rownames(hoge), hoge)                    #ファイルに出力した
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)#tmpの中身をout
 以下にエラー file(file, ifelse(append, "a", "w")) :
   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, ifelse(append, "a", "w")) :
   ファイル 'hoge2.txt' を開くことができません: Permission denied
>
>
```



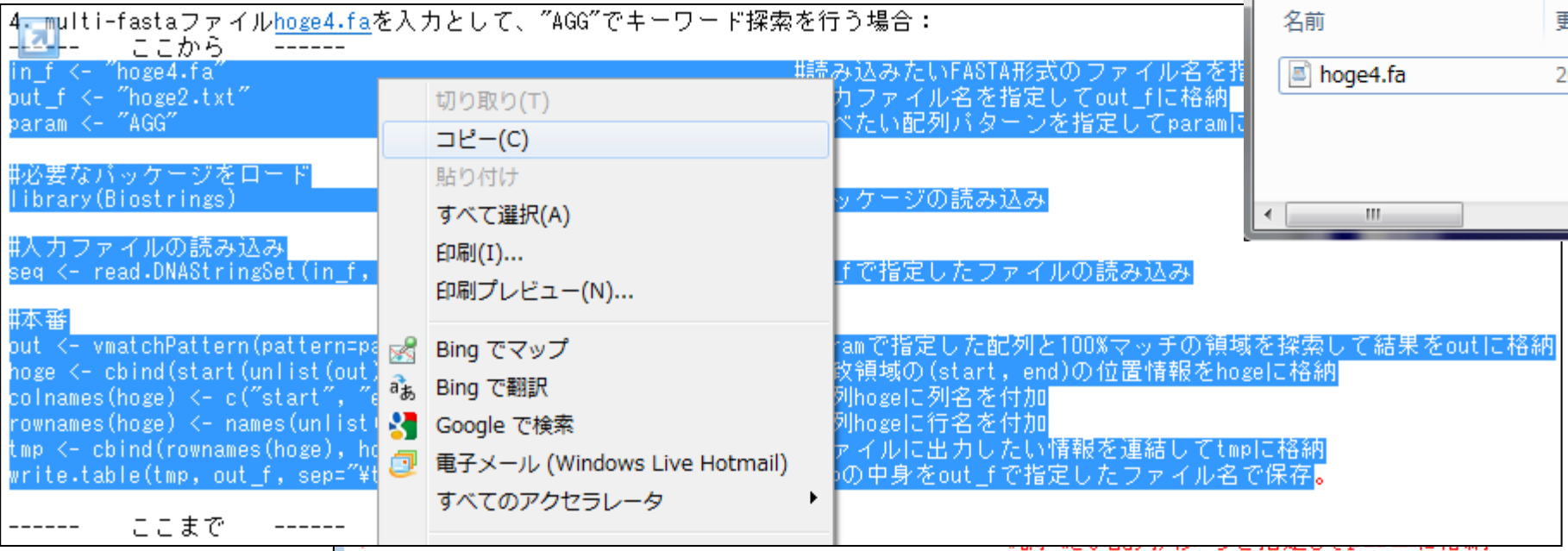


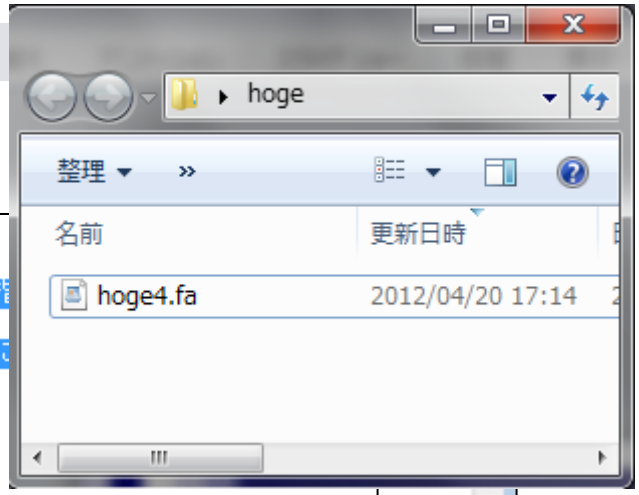
出力予定のファイル名と同じものを別のプログラムで開いているため最後のwrite.table関数のところでエラーが出る

ありがちなミス4

4. multi-fastaファイルhoge4.faを入力として、"AGG"でキーワード探索を行う場合：

```
------    ここから    ------
in_f <- "hoge4.fa"                                    #読み込みたいFASTA形式のファイル名を指
out_f <- "hoge2.txt"                                  #カファイル名を指定してout_fに格納
param <- "AGG"                                        #べたい配列パターンを指定してparamに

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

#入力ファイルの読み込み
seq <- read.DNAStringSet(in_f,                        #fで指定したファイルの読み込み

#本番
out <- vmatchPattern(pattern=pa                       #amで指定した配列と100%マッチの領域を探索して結果をoutに格納
hoge <- cbind(start(unlist(out)                       #致領域の(start, end)の位置情報をhogeに格納
colnames(hoge) <- c("start", "e                       #列hogeに列名を付加
rownames(hoge) <- names(unlist(                       #列hogeに行名を付加
tmp <- cbind(rownames(hoge), ho                       #ァイルに出力したい情報を連結してtmpに格納
write.table(tmp, out_f, sep="¥t                       #の中身をout_fで指定したファイル名で保存。

------    ここまで    ------
```





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）

```
$ロード
$("start", "end")                                     #行列hogeに列名を付加
$ames(unlist(out))                                    #行列hogeに行名を付加
$es(hoge), hoge)                                      #ファイルに出力したい情報を連結してtmpに格納
$t_f, sep="\t", append=F, quote=F, row.names=F)       #tmpの中身をout_fで指定したファイル名で保存
```

- 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 2012/04/13) NEW
- 解析 | 一般 | GC含量 (GC contents) (last modified 2010/7/1)
- 解析 | 一般 | Sequence logos (Schneider_1990) (last modified 2012/04/04) NEW

- 解析 | 一般 | パターンマッチング

解析 | 一般 | パターンマッチング

読み込んだリファレンス配列(reference sequence or subject sequence)から(短い)配列パターンを探す場合に利用します。
ここでは、以下の6つの例題を行います:

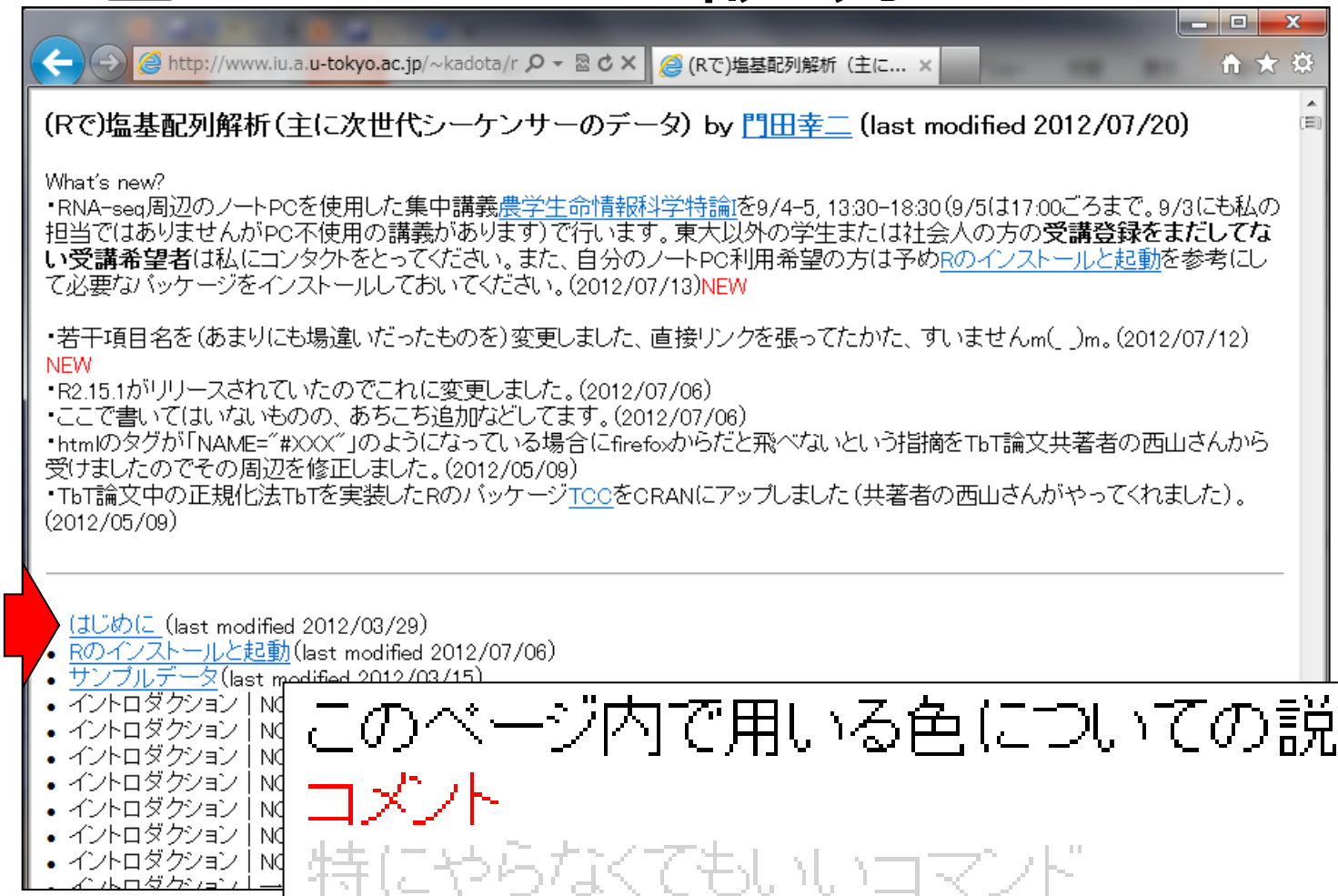
1. Dihyd
Finger N
2. 1. と同
3. 2. と同
4. multi-
5. multi-
6. multi-
7. multi-
(記述の

4. multi-fastaファイルhoge4.faを入力として、"AGG"でキーワード探索を行う場合：

```
------    ここから    ------
in_f <- "hoge4.fa"                                  #読み込みたいFASTA形式のファイル名を指
out_f <- "hoge2.txt"                                #出力ファイル名を指定してout_fに格納
param <- "AGG"                                      #調べたい配列パターンを指定してparam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
seq <- read.DNAStringSet(in_f, format="fasta")      #in_fで指定したファイルの読み込み

#本番
out <- vmatchPattern(pattern=param, subject=seq)    #paramで指定した配列と100%マッチの領域
hoge <- cbind(start(unlist(out)), end(unlist(out)))  #一致領域の(start, end)の位置情報をhog
colnames(hoge) <- c("start", "end")                 #行列hogeに列名を付加
rownames(hoge) <- names(unlist(out))                #行列hogeに行名を付加
tmp <- cbind(rownames(hoge), hoge)                  #ファイルに出力したい情報を連結してtmp
write.table(tmp, out_f, sep="¥t", append=F, quote=F, row.names=F)#tmpの中身をout_fで指定したファイル名

------    ここまで    ------
```

「----- ここまで -----」の一つ上の空行には「スクリプト最終行のコマンドを確実に実行するため」という深い意味があります

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

- 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 2012/04/13) NEW
- 解析 | 一般 | GC含量 (GC contents) (last modified 2010/7/1)
- 解析 | 一般 | Sequence logos (Schneider_1990) (last modified 2012/04/04) NEW

- 解析 | 一般 | パターンマッチング

解析 | 一般 | パターンマッチング

読み込んだリファレンス配列(reference sequence or subject sequence)から(短い)配列パターンを探す場合に利用します。
ここでは、以下の6つの例題を行います:

1. Dihyd
Finger N
2. 1. と同
3. 2. と同
4. multi-
5. multi-
6. multi-
7. multi-
(記述の

4. multi-fastaファイルhoge4.faを入力として、"AGG"でキーワード探索を行う場合：

```
------    ここから    ------
in_f <- "hoge4.fa"                                  #読み込みたいFASTA形式のファイル名を指
out_f <- "hoge2.txt"                                #出力ファイル名を指定してout_fに格納
param <- "AGG"                                      #調べたい配列パターンを指定してparam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
seq <- read.DNAStringSet(in_f, format="fasta")      #in_fで指定したファイルの読み込み

#本番
out <- vmatchPattern(pattern=param, subject=seq)    #paramで指定した配列と100%マッチの領域
hoge <- cbind(start(unlist(out)), end(unlist(out)))  #一致領域の(start, end)の位置情報をhog
colnames(hoge) <- c("start", "end")                 #行列hogeに列名を付加
rownames(hoge) <- names(unlist(out))                #行列hogeに行名を付加
tmp <- cbind(rownames(hoge), hoge)                  #ファイルに出力したい情報を連結してtmp
```

hoge4.faファイルに対してNGS由来塩基配列データ(例:"CCT")のマッピング(or 文字列検索)を行い、一致領域情報を任意のファイル名(例:"hoge3.txt")で出力したいときは？

4. multi-fastaファイルhoge4.faを入力として、"AGG"でキーワード探索を行う場合：

------　ここから　------

```
in_f <- "hoge4.fa"                                  #読み込みたいFASTA形式の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
param <- "AGG"

#必要なパッケージをロード
library(Biostrings)
```

切り取り(T)
コピー(C)
貼り付け
すべて選択(A)

①テンプレートのスクリプトをコピーして

無題 - メモ帳

ファイル(F)　編集(E)　書式(O)　表示(V)　ヘルプ(H)

```
in_f <- "hoge4.fa"                                  #読み込みたいFASTA形式の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
param <- "AGG"                                      #調べたい配列パターンを指定してparamに格納

#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
seq <- read.DNAStringSet(in_f, format="fasta")      #in_fで指定したファイルの読み込み

#本番
out <- vmatchPattern(pattern=param, subject=seq)    #paramで指定した配列と100%マッチの領域を探索して結果をoutに
```

②メモ帳などのテキストエディタにペーストして

無題 - メモ帳

ファイル(F)　編集(E)　書式(O)　表示(V)　ヘルプ(H)

```
in_f <- "hoge4.fa"                                  #読み込みたいFASTA形式のファイル名を指定してin_fに格納
out_f <- "hoge3.txt"                                #出力ファイル名を指定してout_fに格納
param <- "CCT"                                      #調べたい配列パターンを指定してparam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
seq <- read.DNAStringSet(in_f, format="fasta")      #in_fで指定したファイルの読み込み

#本番
out <- vmatchPattern(pattern=param, s               配列と100%マッチの領域を探索して結果をoutに
hoge <- cbind(start(unlist(out)), enc               t, end)の位置情報をhogeに格納
colnames(hoge) <- c("start", "end")                 を付加
rownames(hoge) <- names(unlist(out))                #行列hogeに行名を付加
tmp <- cbind(rownames(hoge), hoge)                  #ファイルに出力したい情報を連結してtmpに格納
write.table(tmp, out_f, sep="¥t", append=F, quote=F, row.names=F)#tmpの中身をout_fで指定したファイル名で保存。
```

③必要な箇所を変更して

2 行、16 列

無題 - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
in_f <- "hoge4.fa"                                    #読み込みたいFASTA形式のファイル名を指定してin_fに格納
out_f <- "hoge3.txt"                                  #出力ファイル名を指定してout_fに格納
param <- "CCT"                                        #調べたい配列パターンを指定してparamに格納

#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
seq <- read.DNAStringSet(in                           #in_fで指定したファイルの読み込み

#本番
out <- vmatchPattern(patter                           #paramで指定した配列と100%マッチの領域を探索して結果をoutに
hoge <- cbind(start(unlist(                           #一致領域の(start, end)の位置情報をhogeに格納
colnames(hoge) <- c("start"                           #行列hogeに列名を付加
rownames(hoge) <- names(unl                           #行列hogeに行名を付加
tmp <- cbind(rownames(hoge)                           #ファイルに出力したい情報を連結してtmpに格納
write.table(tmp, out_f, sep                           names=F)#tmpの中身をout_fで指定したファイル名で保存。
```

元に戻す(U)
切り取り(T)
コピー(C)
貼り付け(P)
削除(D)
すべて選択(A)
右から左に読む(R)
Unicode 制御文字の表示(S)
Unicode 制御文字の挿入(I)
IME を開く(O)

④変更後のスクリプトをまたコピーして

R Console

```
> getwd()
[1] "C:/Users/kadota_DELL/Desktop/hoge"
> out_f <- "hoge3.txt"
> param <- "CCT"
>
> #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
> seq <- read.DNAStringSet(in_f, format="fasta")                    #in_fで指定したファイル$
>
> #本番
> out <- vmatchPattern(pattern=param, subject=seq)                  #paramで指定した配列と10$
> hoge <- cbind(start(unlist(out)), end(unlist(out)))               #一致領域の(start, end)$
> colnames(hoge) <- c("start", "end")                               #行列hogeに列名を付加
> rownames(hoge) <- names(unlist(out))                              #行列hogeに行名を付加
> tmp <- cbind(rownames(hoge), hoge)                                #ファイルに出力したい情$
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)#tmpの中身をout_fで指定$
>
```

⑤(入力ファイルがあるフォルダの場所になっているかどうかをちゃんと確認して)ペースト

実行結果

hoge3.txt M...

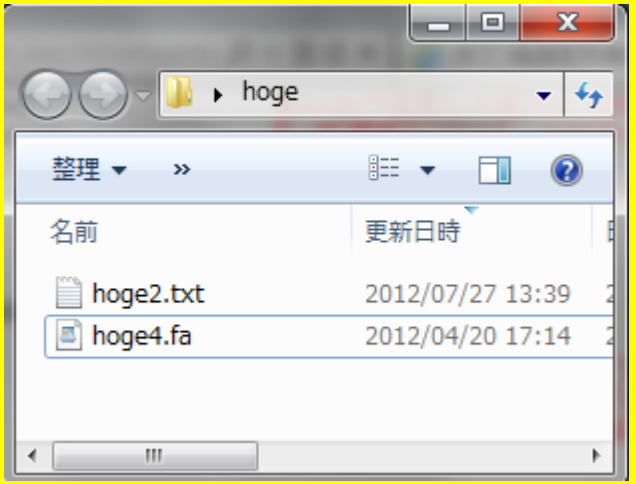
	A	B	C	D
1		start	end	
2	contig_1	10	12	
3	contig_2	6	8	
4	contig_2	34	36	
5	contig_2	98	100	
6	contig_4	32	34	
7	contig_4	38	40	
8				

hoge4.fa - メモ帳

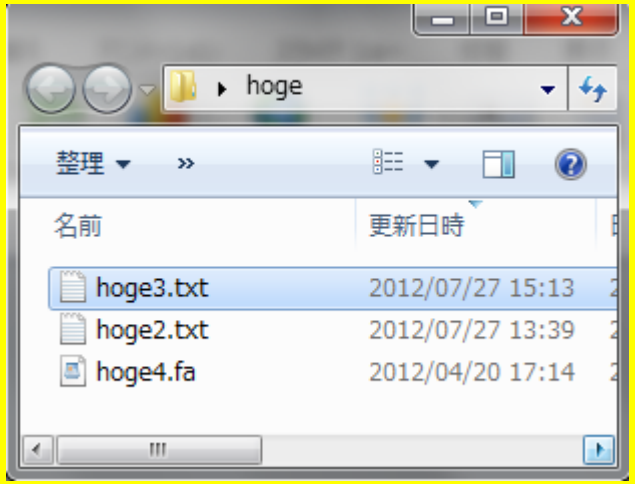
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

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

実行前のhogeフォルダ



実行後のhogeフォルダ



より現実に近い解析

data_reads.txt

```
>seq1
TTT
>seq2
GGG
>seq3
ACT
>seq4
ACA
```

- 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 2012/04/13) NEW
- 解析 | 一般 | GC含量 (GC content) (last modified 2010/7/1)
- 解析 | 一般 | Sequence logos (Schneider_1990) (last modified 2012/04/04) NEW

• 解析 | 一般 | パターンマッチング

6. multi-fastaファイルhoge4.faをリファレンス配列（マップされ ドからなるdata_reads.txtで

```
------    ここから    ------
in_f1 <- "hoge4.fa"                                  #読み込みたいFASTA形式ファイルを指定して
in_f2 <- "data_reads.txt"                            #読み込みたいFASTA形式ファイルを指定して
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)

#入力ファイルの読み込み
seq <- read.DNAStringSet(in_f1, format="fasta")      #in_f1で指定したファイルの読み込み
reads <- read.DNAStringSet(in_f2, format="fasta")    #in_f2で指定したファイルの読み込み

#本番
out <- c("in_f2", "in_f1", "start", "end")           #最終的に得る出力ファイルのヘッダー情報
for(i in 1:length(reads)){                           #リード数分だけループを回す
    tmp <- vmatchPattern(pattern=as.character(reads[i]), subject=seq)#オブジェクトreads中の各塩基配列と1
    hoge1 <- cbind(start(unlist(tmp)), end(unlist(tmp)))  #一致領域の(start, end)の位置情報をhoge
    hoge2 <- names(unlist(tmp))                      #ヒットしたリファレンス配列中のIDをhoge2
    hoge3 <- rep(as.character(reads[i]), length(hoge2))  #hoge2の要素数分だけ、マップする側の配列
    out <- rbind(out, cbind(hoge3, hoge2, hoge1))    #cbind(hoge3, hoge2, hoge1)で表される欲
}
write.table(out, out_f, sep="¥t", append=F, quote=F, row.names=F, col.names=F)#resultの中身をout_fで指定

------    ここまで    ------
```

複数個のリードからなるファイルを読み込んで
一度にマッピング結果を返すことも可能です

6. multi-fastaファイルhoge4.faをリファレンス配列（マップされる側）として、4リードからなるdata_reads.txtで

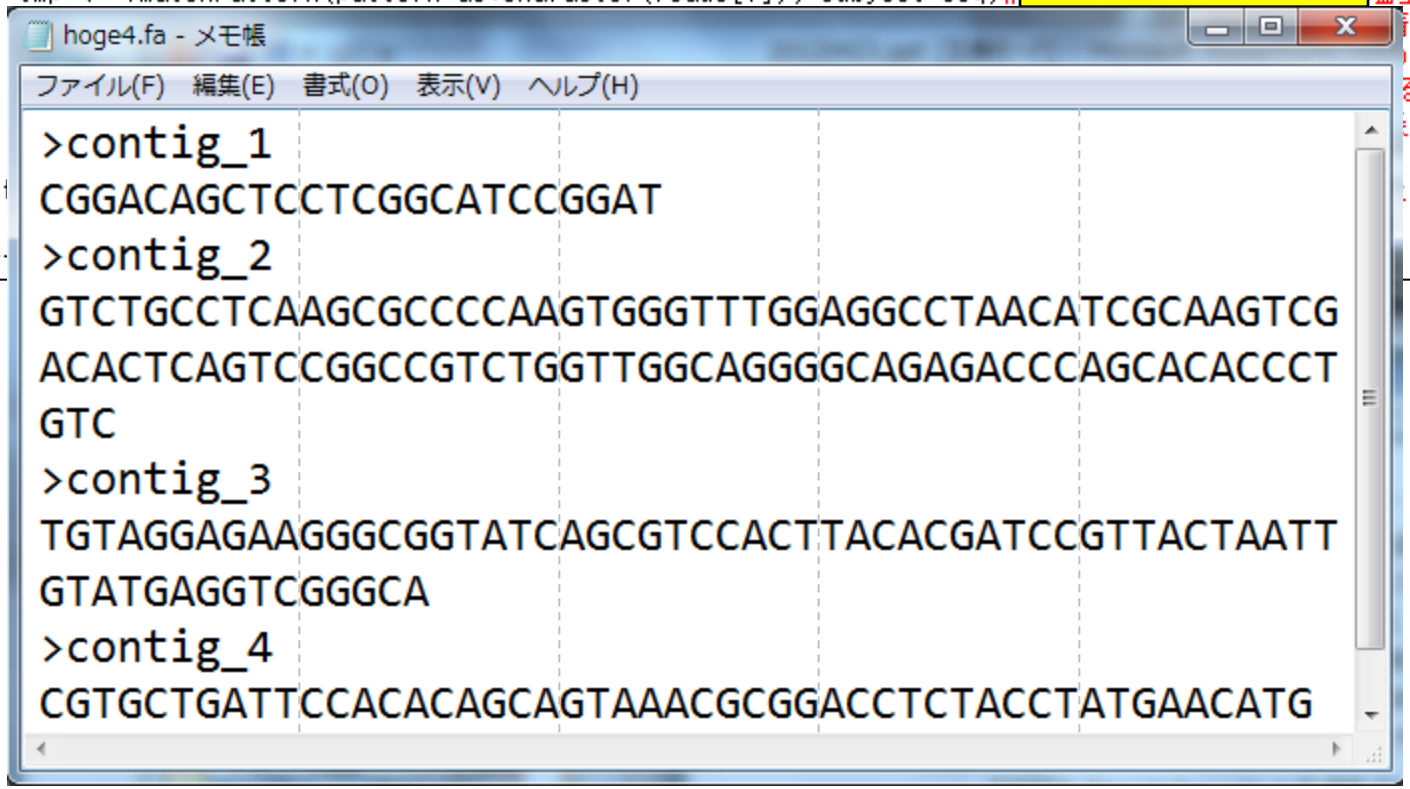
```
------    ここから    ------
in_f1 <- "hoge4.fa"                                    #読み
in_f2 <- "data_reads.txt"                              #読み
out_f <- "hoge4.txt"                                   #出力

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

#入力ファイルの読み込み
seq <- read.DNAStringSet(in_f1, format="fasta")        #in_f
reads <- read.DNAStringSet(in_f2, format="fasta")      #in_f

#本番
out <- c("in_f2", "in_f1", "start", "end")             #最終
for(i in 1:length(reads)){                             #リー
    tmp <- vmatchPattern(pattern=as.character(reads[i]), subject=seq)#
}
writ
```

data_reads.txt

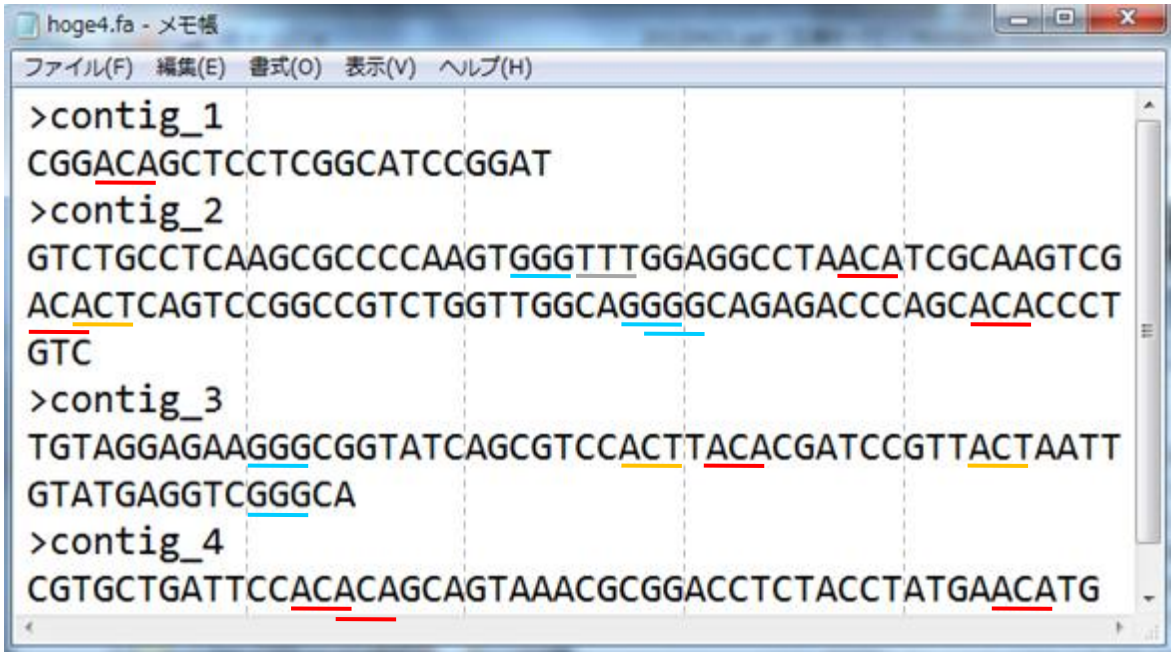
```
>seq1
TTT
>seq2
GGG
>seq3
ACT
>seq4
ACA
```

出力ファイル：hoge4.txt

in_f2	in_f1	start	end
TTT	contig_2	26	28
GGG	contig_2	23	25
GGG	contig_2	78	80
GGG	contig_2	79	81
GGG	contig_3	11	13
GGG	contig_3	61	63
ACT	contig_2	53	55
ACT	contig_3	28	30
ACT	contig_3	44	46
ACA	contig_1	4	6
ACA	contig_2	38	40
ACA	contig_2	51	53
ACA	contig_2	94	96
ACA	contig_3	32	34
ACA	contig_4	13	15
ACA	contig_4	15	17
ACA	contig_4	45	47



出力結果ファイルと発現量の関係



data_reads.txt

```
>seq1
TTT
>seq2
GGG
>seq3
ACT
>seq4
ACA
```

出力ファイル：hoge4.txt

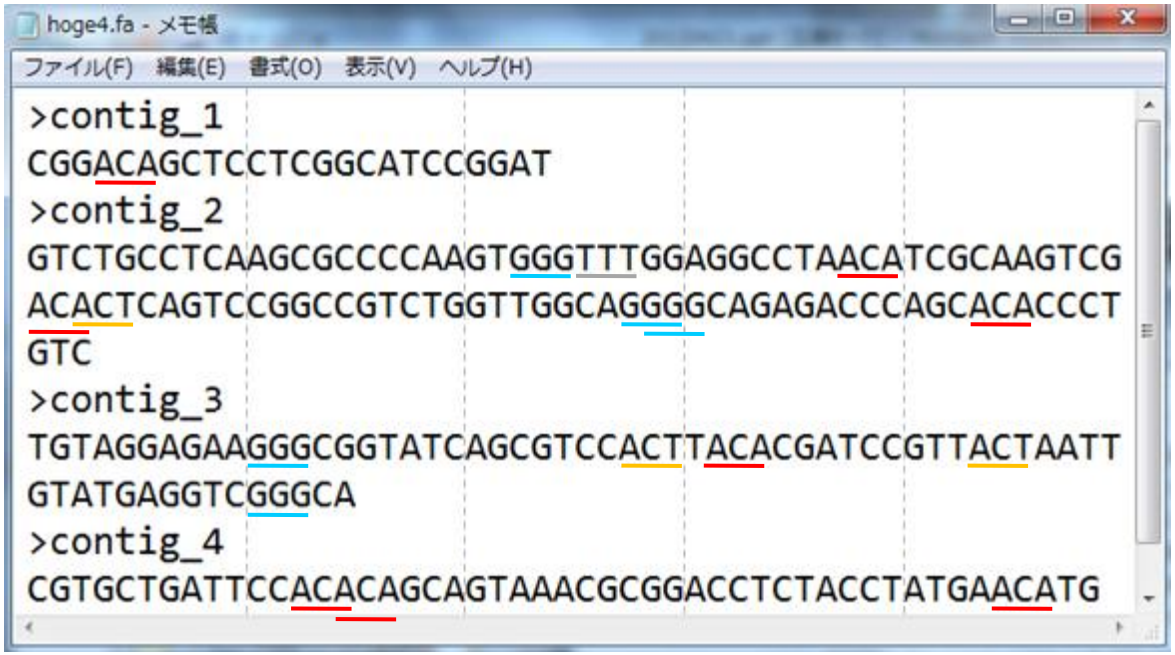
in_f2	in_f1	start	end
TTT	contig_2	26	28
GGG	contig_2	23	25
GGG	contig_2	78	80
GGG	contig_2	79	81
GGG	contig_3	11	13
GGG	contig_3	61	63
ACT	contig_2	53	55
ACT	contig_3	28	30
ACT	contig_3	44	46
ACA	contig_1	4	6
ACA	contig_2	38	40
ACA	contig_2	51	53
ACA	contig_2	94	96
ACA	contig_3	32	34
ACA	contig_4	13	15
ACA	contig_4	15	17
ACA	contig 4	45	47

multi mapper（複数個所にマップされるリード）の取り扱いは？

よく見かけるカウントデータ取得条件

- basic alignerの一つであるBowtieプログラムを利用して、リファレンス配列(ゲノム or トランスクリプトーム)の一カ所とのみ(最大2塩基ミスマッチまで許容して)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)
  - …

Unique mapperのみにすると…

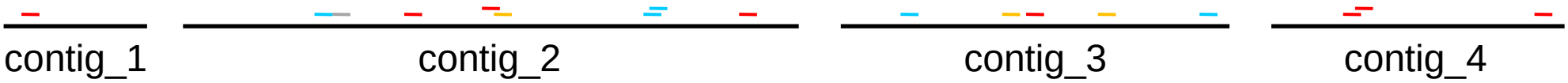


data_reads.txt

```
>seq1
TTT
>seq2
GGG
>seq3
ACT
>seq4
ACA
```

出力ファイル：hoge4.txt

in_f2	in_f1	start	end
TTT	contig_2	26	28
GGG	contig_2	23	25
GGG	contig_2	78	80
GGG	contig_2	79	81
GGG	contig_3	11	13
GGG	contig_3	61	63
ACT	contig_2	53	55
ACT	contig_3	28	30
ACT	contig_3	44	46
ACA	contig_1	4	6
ACA	contig_2	38	40
ACA	contig_2	51	53
ACA	contig_2	94	96
ACA	contig_3	32	34
ACA	contig_4	13	15
ACA	contig_4	15	17
ACA	contig 4	45	47



contig_1	1
contig_2	8
contig_3	5
contig_4	3

⇒

contig_1	0
contig_2	1
contig_3	0
contig_4	0

• 前処理 | トランスクリプトーム配列へのマップ後のファイルからマップされたリード数をカウント(BED形式ファイル)

手元に「**multi-fasta形式のマップされる側の塩基配列ファイル**(例: h_rna.fastaやhoge4.fa)」と Bowtie (Langmead et al., 2009)などのマッピングプログラムを用いて得られた「**どこにマップされたかを表すBED形式ファイル**(例: SRR002324_t.bed、sample_1.bed、sample_2.bed)」があるとします。
発現変動遺伝子(DEG)同定の目的で用いられるTCCやedgeR などのRパッケージは遺伝子(あるいは転写物)ごとに何個のリードがマップされたかという「カウント行列(count matrix)」を入力とすることを前提としています。(ざっくり言えば)BED形式ファイルは、トランスクリプトーム配列にマップした場合、「どのtranscript(or 遺伝子)の(1列目)、どの位置から(2列目; start)、どの位置まで(3列目; end)にリードがマップされた」という情報からなります。それゆえ「行数がマップされたリード数」に相当し、「遺伝子ごとの出現数がカウント数」に相当するわけです。
したがって、基本的にはBED形式ファイルを読み込んで「遺伝子名に相当する列」の部分の情報のみを用いて「どの遺伝子名が何回出現したか」をカウントすればいいだけなんですが... 実際には、二つのサンプル(or 二つのBED形式ファイル)を比較する場合などには「片方のBEDファイル中に一度も出現しない遺伝子」が存在しうるため、「multi-fasta形式のマップされる側の塩基配列ファイル」も同時に読み込んでその遺伝子リストの並びで出力してやったほうが都合がいいのでここではそうしています。

「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. マップされる側の配列 (hoge4.fa) 中のIDの並びでBED形式ファイル(sample_1.bed)中の出現回数をカウントする場合：

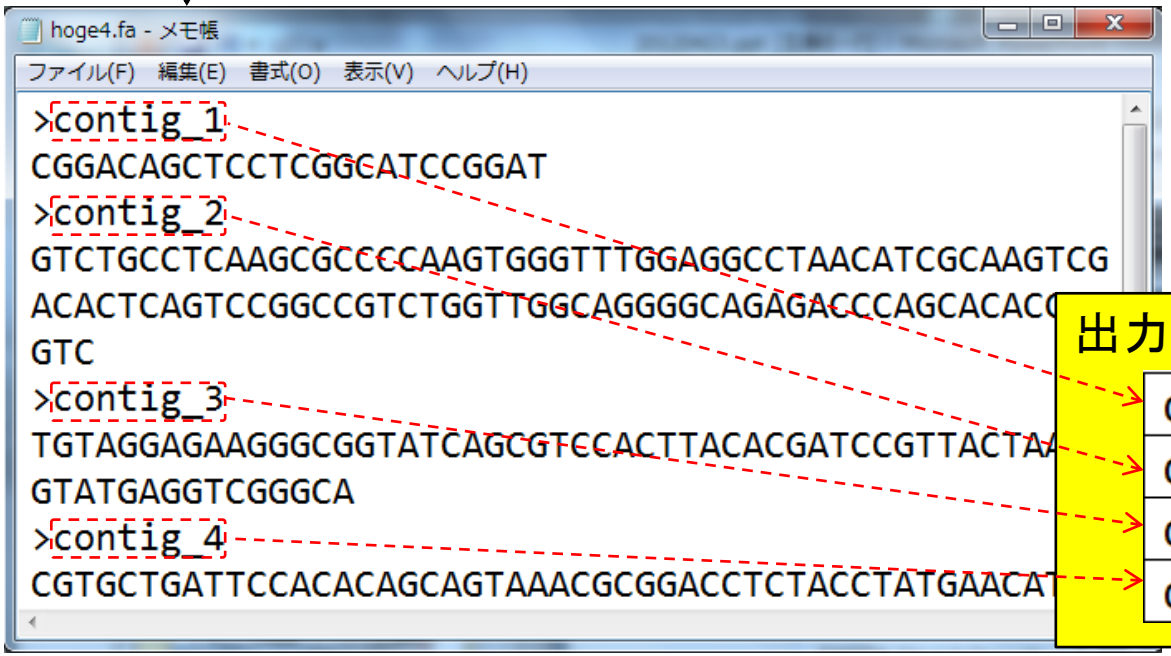
```
------    ここから    ------
in_f1 <- "sample_1.bed"                          #読み込みたいBED形式フ
in_f2 <- "hoge4.fa"                              #マップに用いたリファレ
out_f <- "output1.txt"                           #出力ファイル名を指定して out_f に格納

#必要なパッケージなどをロード
library(ShortRead)                               #パッケージの読み込み

#入力ファイルの読み込みと必要な情報の抽出など
```

1.をやってみましょう。

入力ファイルと目的のおさらい

- 入力ファイル1：sample_1.bed
  - BED形式ファイル。1列目の情報のみを用いてコンティグ（遺伝子ID）ごとのカウント（出現回数）情報取得のために利用。
- 入力ファイル2：hoge4.fa
  - マップに用いたリファレンス配列。multi-fasta形式ファイル。Description行のコンティグ名（ID）の並びで結果を出力させるために利用。

in_f1	start	end
contig_2	26	28
contig_2	23	25
contig_2	78	80
contig_2	79	81
contig_3	11	13
contig_3	61	63
contig_2	53	55
contig_3	28	30
contig_3	44	46
contig_1	4	6
contig_2	38	40
contig_2	51	53
contig_2	94	96
contig_3	32	34
contig_4	13	15
contig_4	15	17
contig_4	45	47



出力ファイル：output1.txt

contig_1	1
contig_2	8
contig_3	5
contig_4	3

http://genome.ucsc.edu/FAQ/FAQformat.html#format1

BED形式

- イントロダクション | NGS | マッピング | (short) readの出力形式について

マッピング | (short) readを眺めると、いろいろな出力形式があることがわかります。
注目すべきは、Sequence Alignment/Map (SAM) formatです。この形式は国際共同研究の1000人のゲノムを解析するという1000 Genomes Projectで採用された(開発された)フォーマットで、("@"から始まる)header sectionと(そうでない)alignment sectionから構成されています。このヒトの目で解読可能な形式がSAMフォーマットで、このバイナリ版がBinary Alignment/Map (BAM)フォーマットというものです。今後SAM/BAM formatという記述を良く見かけるようになることでしょう

代表的な出力ファイル形式

- BED format
- ELAND format
- GFF (General Feature Format)
- GFF3 (General Feature Format)
- SAM (Sequence Alignment/Map)
- SOAP format
- ZOOM format

UCSC Genome Bioinformatics

Home - Genomes - Blat - Tables - Gene Sorter - PCR - Proteome - Help

Frequently Asked Questions: Data File Formats

- BED format
- bigBed format
- BED (detail) format
- PSL format
- GFF format
- GTF format

BED format

BED format provides a flexible way to define the data lines that are displayed in an annotation track. BED lines have three r
and nine additional optional fields. The number of fields per line must be consistent throughout any single set of data in an ann
The order of the optional fields is binding: lower-numbered fields must always be populated if higher-numbered fields are use

The first three required BED fields are:

1. **chrom** - The name of the chromosome (e.g. chr3, chrY, chr2_random) or scaffold (e.g. scaffold10671).
2. **chromStart** - The starting position of the feature in the chromosome or scaffold. The first base in a chromosome is nu
3. **chromEnd** - The ending position of the feature in the chromosome or scaffold. The *chromEnd* base is not included in of the feature. For example, the first 100 bases of a chromosome are defined as *chromStart=0, chromEnd=100*, an bases numbered 0-99.

The 9 additional optional BED fields are:

マッピング結果の出力ファイル形式

- (ゲノム配列の場合)どの染色体上のどの位置に(どのリードが)マッピングされたか、あるいは(トランスクリプトーム配列の場合)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(フォーマット)は複数あります:
  - **BED** (Browser Extensible Data) format
    - BEDtools (Quinlan et al., *Bioinformatics*, **26**: 841-842, 2010)
  - GFF (General Feature Format) format
  - **SAM** (Sequence Alignment/Map) format
    - SAMtools (Li et al., *Bioinformatics*, **25**: 2078-2079, 2009)
  - …

実行結果

1．マップされる側の配列（hoge4.fa）中のIDの並びでBED形式ファイル(sample_1.bed)中の出現回数をカウントする場合：

```
------    ここから    ------
in_f1 <- "sample_1.bed"                    #読み込みたいBED形式ファイル名を指定してin_f1
in_f2 <- "hoge4.fa"                        #マップに用いたリファレンス配列のファイル名を
out_f <- "output1.txt"                     #出力ファイル名を指定してout_fに格納

#必要なパッケージなどをロー
library(ShortRead)                         #パッケージの読み込み

#入力ファイルの読み込みと必
reads <- readDNAStringSet(i                #in_f2で指定したファイルを読み込んでreadsに格
data <- read.table(in
ID_list <- as.vector(
out <- rle(sort(ID_li

#本番
rawcount <- rep(0, le
names(rawcount) <- na
hoge <- out$lengths
names(hoge) <- out$va
obj <- is.element(nam
common <- intersect(n
rawcount[obj] <- hoge

#ファイル出力
tmp <- cbind(names(re
write.table(tmp, out_

------    ここまで
```

切り取り(T)
コピー(C)
貼り付け
すべて選択(A)
印刷(I)...

R Console

```
 要求されたパッケージ latticeExtra をロード中です
 要求されたパッケージ RColorBrewer をロード中です
>
> #入力ファイルの読み込みと必要な情報の抽出など
> reads <- readDNAStringSet(in_f2, format="fasta")          #in_
> data <- read.table(in_f1, header=TRUE, sep="\t")          #in_
> ID_list <- as.vector(data[,1])                            #行列
> out <- rle(sort(ID_list))                                 #どの
>
> #本番
> rawcount <- rep(0, length(reads))                         #最終
> names(rawcount) <- names(reads)                           #nam
> hoge <- out$lengths                                       #out
> names(hoge) <- out$values                                 #out
> obj <- is.element(names(rawcount), names(hoge))           #nam
> common <- intersect(names(rawcount), names(hoge))         #names(ra$
> rawcount[obj] <- hoge[common]                             #objがTRU$
>
> #ファイル出力
> tmp <- cbind(names(reads), rawcount)                      #結果ファ$
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.name$
>
>
```

出力ファイル：output1.txt

contig_1	1
contig_2	8
contig_3	5
contig_4	3

比較トランスクリプトーム解析の流れ

複数のFASTQファイル

↓

リファレンス配列の作成

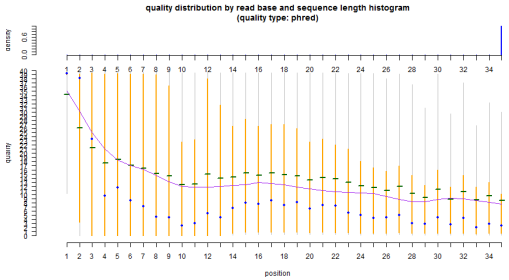
↓

マッピング

↓

データ解析

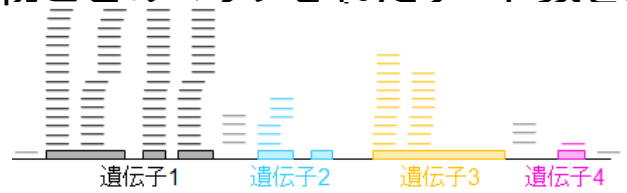
クオリティチェック



アセンブル結果（multi-fasta）ファイルから平均長やトータルの長さなどの基本情報を抽出

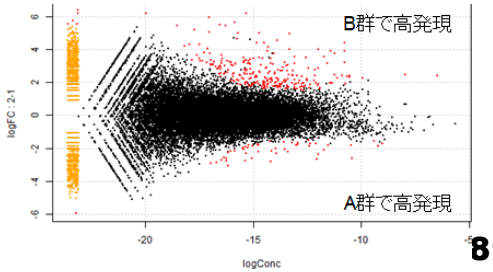
Total length (bp)	2993
Number of contigs	4
Average length	748.3
Median length	784
Max length	888
Min length	537
N50	886
GC content	0.524

マッピング結果（BED形式）ファイルを入力として、転写物ごとのマップされたリード数をカウント



遺伝子1	40
遺伝子2	6
遺伝子3	20
遺伝子4	1

発現変動遺伝子のリストアップや、作図など



研究目的別留意点

- 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場合
  - 「配列長」由来bias：長いほど沢山sequenceされる
  - 「GC含量」由来bias：カウント数の分布がGC含量依存的である
- **サンプル間**比較（sample A vs. Bなど）で、発現変動遺伝子（DEG）を調べたい場合
  - 「sequence depthの違い」：総リード数がx倍違うと全体的にx倍変動…
  - 「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…
  - RPM（CPM）正規化 → TMM正規化 → TbT正規化 → iDEGES正規化

総リード数を揃えるだけ

DEGを（正確には見積もらないので）多めにトリム

正規化の手順の中で同定したDEGをトリムすることでより頑健に

律速であったDEG同定部分の改良により、より頑健且つ高速に

配列長を考慮した発現量推定のイメージ

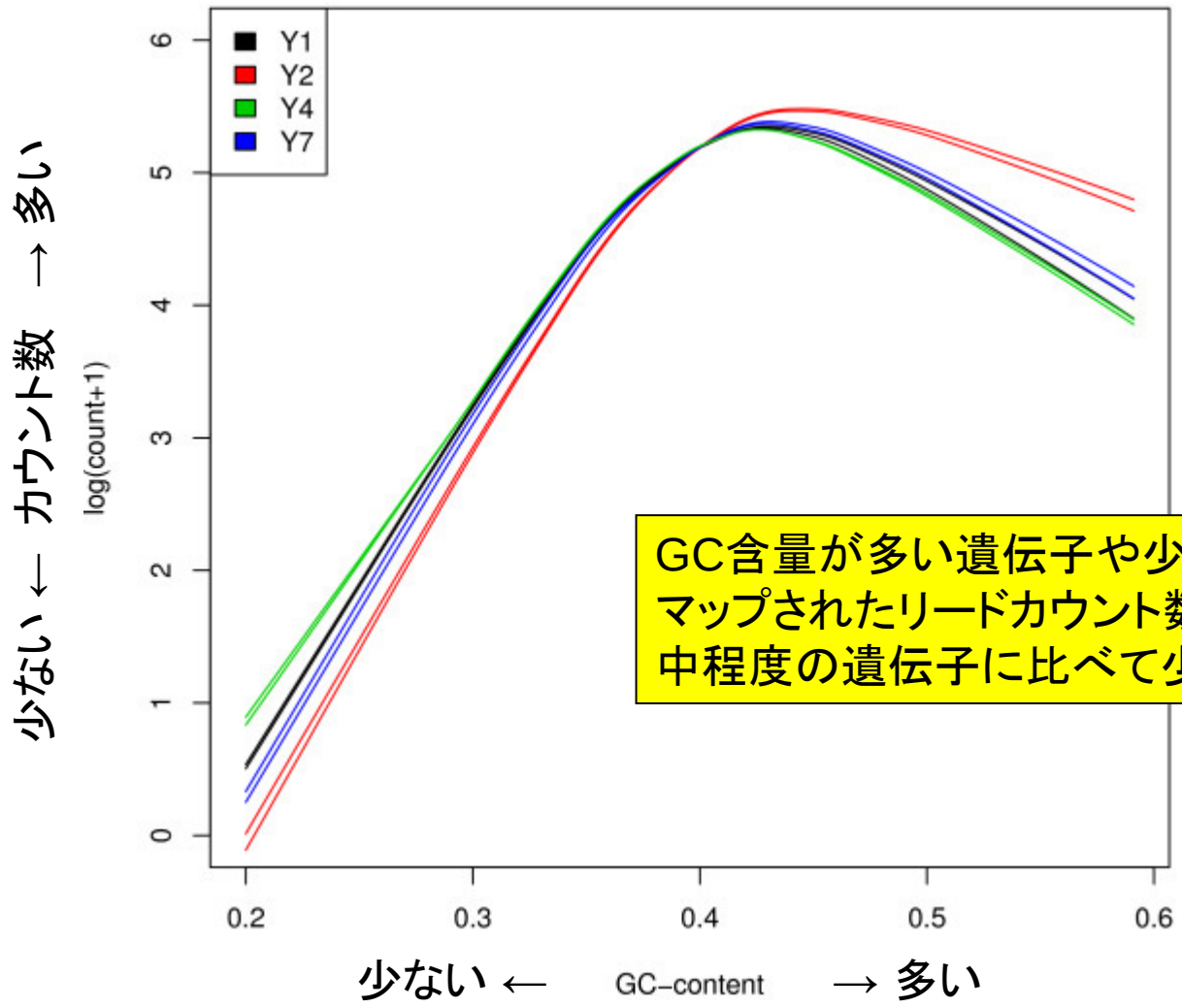
- gene1: 3 exons (middle length), 14 reads mapped (**low** coverage)
- gene2: 3 exons (middle length), 56 reads mapped (**high** coverage)
- gene3: 2 exons (**short** length), 12 reads mapped (middle coverage)
- gene4: 2 exons (**long** length), 31 reads mapped (middle coverage)

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

・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）

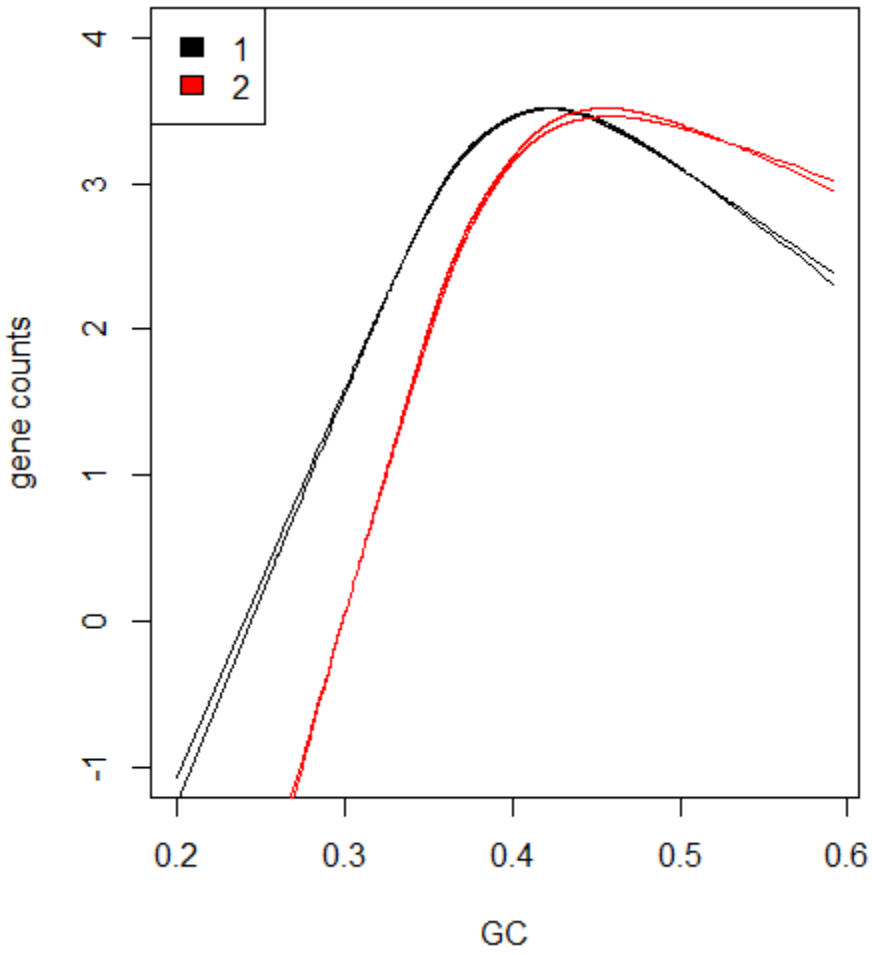
一つのサンプル内で転写物（遺伝子）間の発現レベルの大小を比較したい場合には配列長を考慮すべきである

GC biasの実例

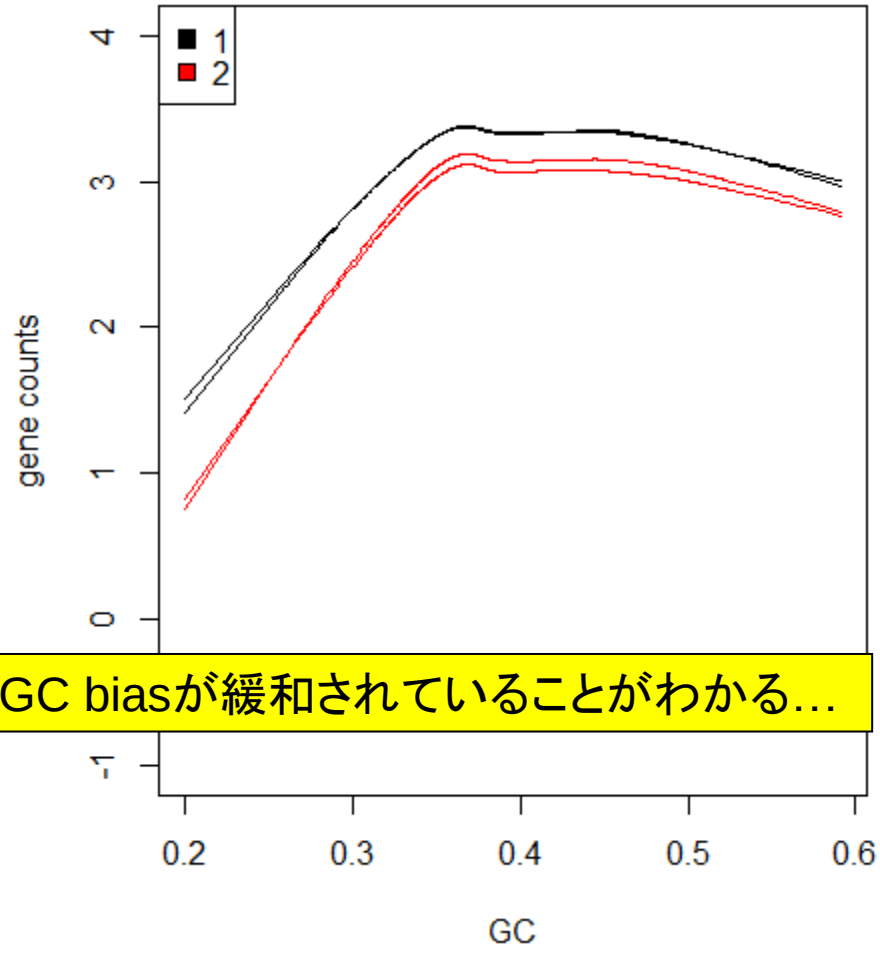


GC含量が多い遺伝子や少ない遺伝子上にマップされたリードカウント数は、GC含量が中程度の遺伝子に比べて少ない傾向にある

GC bias補正（EDASeqパッケージ）



Quantile
正規化



GC biasが緩和されていることがわかる…

研究目的別留意点

- 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場合
  - 「配列長」由来bias：長いほど沢山sequenceされる
  - 「GC含量」由来bias：カウント数の分布がGC含量依存的である
- **サンプル間**比較（sample A vs. Bなど）で、発現変動遺伝子（DEG）を調べたい場合
  - 「sequence depthの違い」：総リード数がx倍違うと全体的にx倍変動…
  - 「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…
  - RPM（CPM）正規化 → TMM正規化 → TbT正規化 → iDEGES正規化

総リード数を揃えるだけ

DEGを（正確には見積もらないので）多めにトリム

正規化の手順の中で同定したDEGをトリムすることでより頑健に

律速であったDEG同定部分の改良により、より頑健且つ高速に

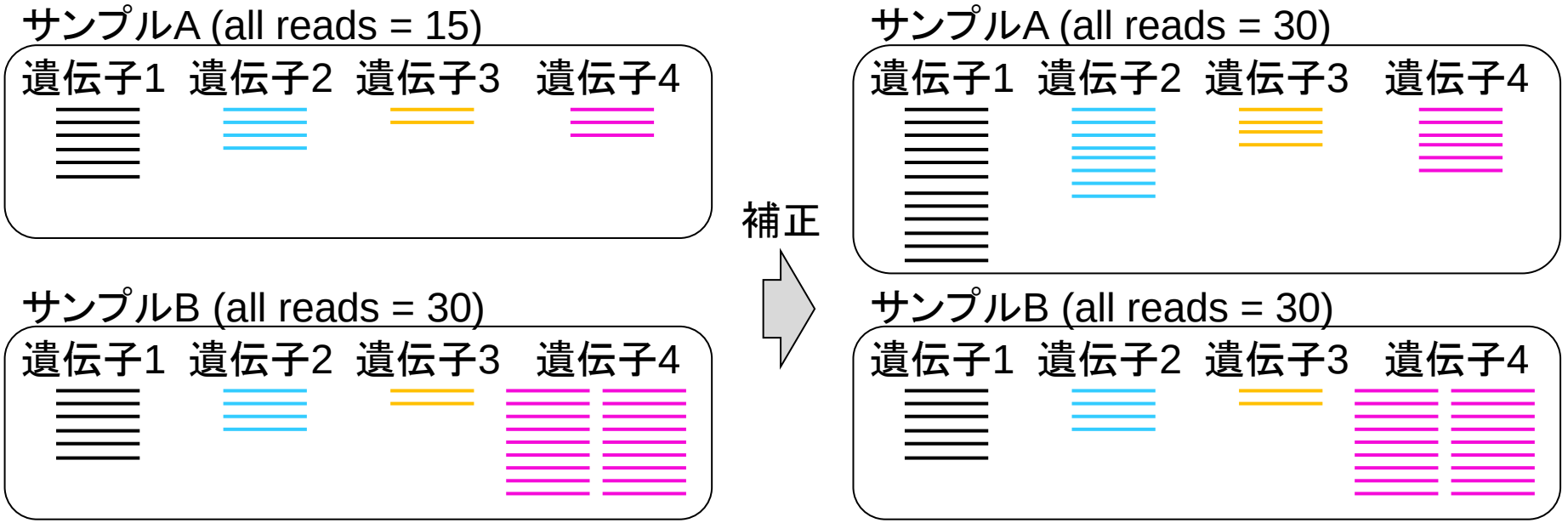
Sequence depth周辺の正規化法

- RPM (Mortazavi *et al*., *Nat. Methods*, **5**: 621-628, 2008)
  - RPKM(Reads per kilobase of exon per million mapped reads)の長さ補正を行わないバージョン
  - Reads per million mapped readsの略。
- TMM正規化 (Robinson and Oshlack, *Genome Biol*., **11**: R25, 2010)
  - Trimmed Mean of M valuesの略
  - 発現変動遺伝子（DEG）のデータ正規化時の悪影響を排除すべく、M-A plot上で周縁部にあるデータを使わずに正規化係数を決定する方法。
- TbT正規化 (Kadota *et al*., *Algorithms Mol. Biol*., **7**: 5, 2012)
  - TMM法の改良版で、TMM-baySeq-TMMという3ステップで正規化を行う方法。
  - 1st stepで得られたTMM正規化係数を用いて、2nd step (baySeq)でDEG同定を行い、3rd step (TMM)ではDEGを排除した残りのデータでTMM正規化。DEGの影響を排除しつつもできるだけ多くのnon-DEGデータを用いて頑健に正規化係数を決めるという思想（DEG elimination strategy提唱論文）。
- iDEGES正規化（Sun *et al*., submitted）
  - DEG elimination strategy (DEGES) を一般化し、より高速且つ頑健にしたもの。TbTは「複製あり」のデータのみにしか対応していなかったが、「複製なし」データにも対応。
  - iDEGES/edgeR正規化法：「複製あり」データ正規化用。TMM-$(\text{edgeR-TMM})_n$パイプライン
  - iDEGES/DESeq正規化法：「複製なし」データ正規化用。DESeq-$(\text{DESeq-DESeq})_n$パイプライン

二群間比較用

RPMの問題点

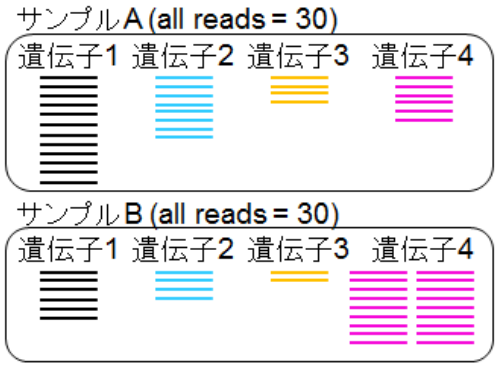
- 仮定
  - 全4遺伝子
  - 配列長は同じ
  - 遺伝子4だけが発現変動遺伝子(DEG)



補正後の解析結果：Aで高発現が3個，Bで高発現が1個

M-A plot

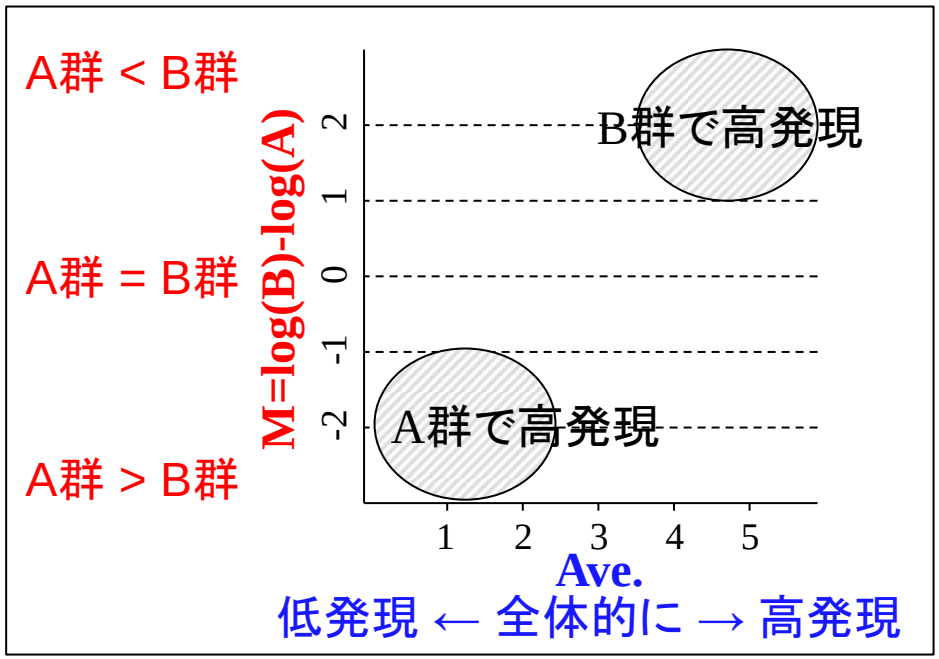
総リード数が30になるように補正した後のデータ

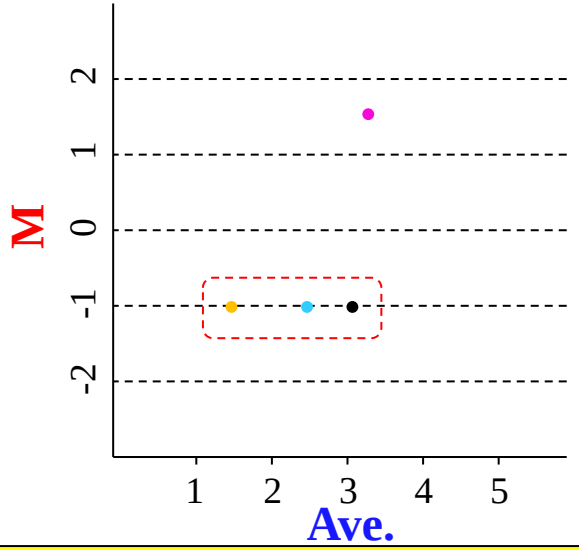


	A	B
遺伝子1	12	6
遺伝子2	8	4
遺伝子3	4	2
遺伝子4	6	18

$\log_2(A)$	$\log_2(B)$
3.58	2.58
3.00	2.00
2.00	1.00
2.58	4.17

縦軸（発現比） R: $\log_2(B/A)$	横軸（全体的な発現レベル） I: log2(sqrt(A×B))
M: $\log_2(B)-\log_2(A)$	A: $(\log_2(A)+\log_2(B))/2$
-1.00	3.08
-1.00	2.50
-1.00	1.50
1.58	3.38





「(B群で)高発現の発現変動遺伝子」の存在が悪影響を及ぼしている

おさらい（RPMの正規化手順）

- サンプルごとのlibrary size（=総リード数）を算出し、遺伝子（行）ごとの生リードカウントをlibrary sizeで割る（さらに、その結果100万を掛ける）

「総リード数は一定」という仮定に基づいてデータの正規化を行うRPM補正（全体の平均値を揃える）は高発現の発現変動遺伝子の悪影響を受ける。

生リードカウント

	A	B
遺伝子1	6	6
遺伝子2	4	4
遺伝子3	2	2
遺伝子4	3	18
sum	15	30

RPM補正後

A	B
400000	200000
266667	133333
133333	66667
200000	600000
1E+06	1E+06

$\log_2$(A)	$\log_2$(B)
18.61	17.61
18.02	17.02
17.02	16.02
17.61	19.19

縦軸 **R**: $\log_2$(B/A)	横軸 **I**: $\log_2$(sqrt(A×B))
M: $\log_2$(B)-$\log_2$(A)	**A**: ($\log_2$(A)+$\log_2$(B))/2
-1.00	18.11
-1.00	17.52
-1.00	16.52
1.58	18.40

やりたいこと：発現変動していない遺伝子（ピンク以外；non Differentially Expressed Genes (non-DEG)）の発現比（M値に相当）の要約統計量（平均とか中央値のこと）が正規化後のデータでできるだけ0になるようにしたい。
RPM補正では-1になっており0から大きく外れていることがわかる

Sequence depth周辺の正規化法

- RPM (Mortazavi *et al*., *Nat. Methods*, **5**: 621-628, 2008)
  - RPKM(Reads per kilobase of exon per million mapped reads)の長さ補正を行わないバージョン
  - Reads per million mapped readsの略。
- TMM正規化 (Robinson and Oshlack, *Genome Biol*., **11**: R25, 2010)
  - Trimmed Mean of M valuesの略
  - 発現変動遺伝子（DEG）のデータ正規化時の悪影響を排除すべく、M-A plot上で周縁部にあるデータを使わずに正規化係数を決定する方法。
- TbT正規化 (Kadota *et al*., *Algorithms Mol. Biol*., **7**: 5, 2012)
  - TMM法の改良版で、TMM-baySeq-TMMという3ステップで正規化を行う方法。
  - 1st stepで得られたTMM正規化係数を用いて、2nd step (baySeq)でDEG同定を行い、3rd step (TMM)ではDEGを排除した残りのデータでTMM正規化。DEGの影響を排除しつつもできるだけ多くのnon-DEGデータを用いて頑健に正規化係数を決めるという思想（DEG elimination strategy提唱論文）。
- iDEGES正規化（Sun *et al*., submitted）
  - DEG elimination strategy (DEGES) を一般化し、より高速且つ頑健にしたもの。TbTは「複製あり」のデータのみにしか対応していなかったが、「複製なし」データにも対応。
  - iDEGES/edgeR正規化法：「複製あり」データ正規化用。TMM-$(\text{edgeR-TMM})_n$パイプライン
  - iDEGES/DESeq正規化法：「複製なし」データ正規化用。DESeq-$(\text{DESeq-DESeq})_n$パイプライン

TMM正規化法

- (発現比に相当する)M値の要約統計量の上位下位それぞれ30%をトリムした後の平均値(trimmed mean)が揃うような正規化係数(TMM正規化係数)をlibrary sizeに掛けることでeffective library sizeを算出し、その値で割る

生リードカウント

	A	B
遺伝子1	6	6
遺伝子2	4	4
遺伝子3	2	2
遺伝子4	3	18
sum	15	30

RPM補正後

A	B
400000	200000
266667	133333
133333	66666.7
200000	600000
1000000	1000000

TMM補正後

A	B
300000	300000
200000	200000
100000	100000
150000	900000
750000	1500000

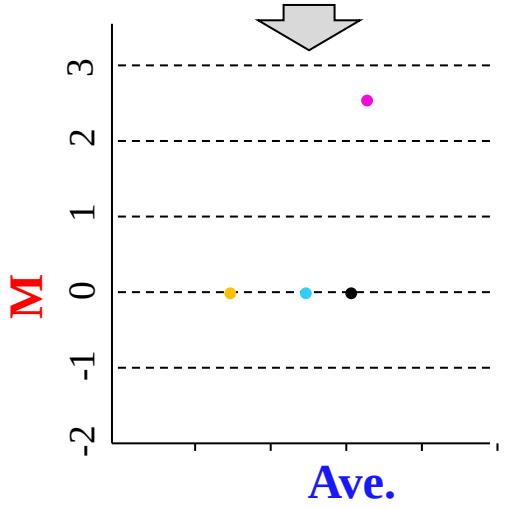
$\log_2(A)$	$\log_2(B)$
18.19	18.19
17.61	17.61
16.61	16.61
17.19	19.78

縦軸(M) 横軸(A)

引き算	足し算
0.00	18.19
0.00	17.61
0.00	16.61
2.58	18.49

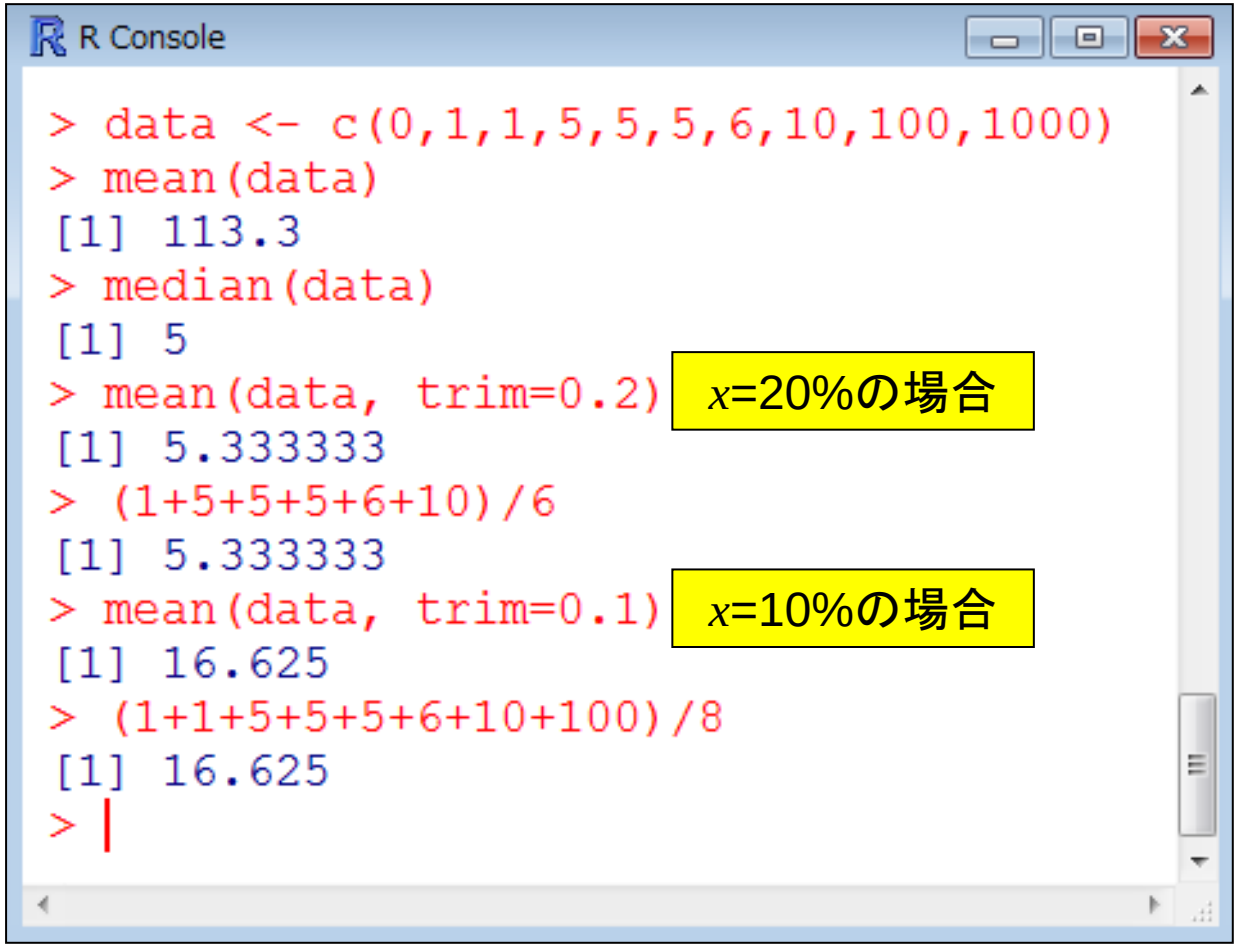
RPM法:生リードカウントを「library size」で割る
TMM法:「library size × TMM正規化係数」で割る

	A	B	
library size	**15**	**30**	
TMM正規化係数	2	1	log2(B/A) = −1
TMM正規化係数	**1.3333**	**0.6666**	log2(B/A) = −1
effective library size	20	20	



Trimmed meanの計算イメージ

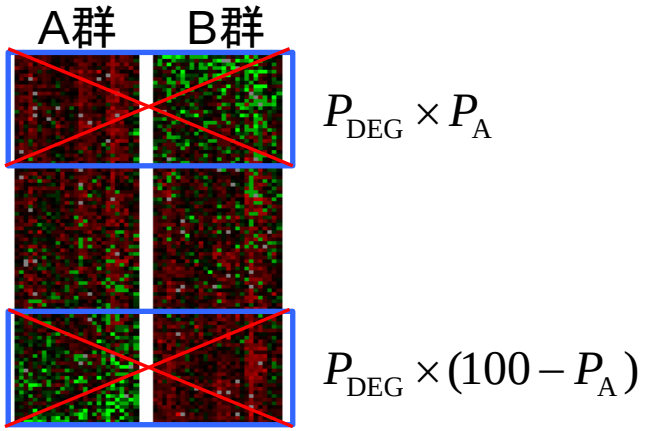
- ある10個の要素からなる数値ベクトル(0,1,1,5,5,5, 6,10,100,1000)があったときに、上位下位それぞれx%を除いて(トリムして)計算する平均値のこと



TMM補正の有無で結論が異なることも…

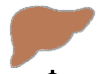
- 得られた発現変動遺伝子（DEG）セット中の割合
  - TMM補正**なし**（Marioni *et al*., *Genome Res*., **18**: 1509-1517, 2008）
    - サンプルA（Kidney）：78%
    - サンプルB（Liver）：22%
  - TMM補正**あり**（Robinson and Oshlack, *Genome Biol*., **11**:R25, 2010）
    - サンプルA（Kidney）：53%
    - サンプルB（Liver）：47%
- TMM法で使用されているパラメータ(一部)
  - $\log_2$(B/A)で発現変動順にランキングし、全体で全遺伝子数の60%分をTrim (P_{DEG} = 60%)。その内訳は、サンプルA側とサンプルB側で高発現なものを各50%とする(P_A = 50%) 。

Trim 後に残ったデータのみを用いて正規化係数を決定



A群 vs. B群の二群間比較

- （当時は常識だった）RPM補正後のデータを用いて、二群で発現の異なる遺伝子（Differentially Expressed Genes; DEGs）を同定した



	A1	A2	A3	A4	A5	B1	B2	B3	B4	B5
EnsemblGeneID	R1 L1 Kidney	R1 L3 Kidney	R1 L7 Kidney	R2 L2 Kidney	R2 L6 Kidney	R1 L2 Liver	R1 L4 Liver	R1 L6 Liver	R1 L8 Liver	R2 L3 Liver
ENSG00000146556	0	0	0	0	0	0	0	0	0	0
ENSG00000197194	0	0	0	0	0	0	0	0	0	0
ENSG00000197490	0	0	0	0	0	0	0	0	0	0
ENSG00000205292										
ENSG00000177693										
ENSG00000209338										
ENSG00000196573										
ENSG00000177799	0	0	0	0	0	0	0	0	0	0
ENSG00000209341	0	0	0	0	0	0	0	0	0	0
ENSG00000209342	0	0	2	4	3	0	0	0	1	0
ENSG00000209343	0	0	0	0	0	0	0	0	0	0
ENSG00000209344	0	0	0	0	0	0	0	0	0	0
ENSG00000209346	0	0	0	0	0	0	0	0	0	0
ENSG00000209349	0	0	0	0	0	0	0	0	0	0
ENSG00000209350	4	7	3	6	7	35	32	31	29	34
ENSG00000209351	0	0	0	0	0	0	0	0	0	0
ENSG00000209352	0	0	1	1	0	2	0	0	0	0
ENSG00000212679	110	131	149	112	118	177	135	141	148	145
ENSG00000212678	12685	13204	12403	13031	13268	9246	9312	8746	8496	9070
ENSG00000185097	0	0	0	0	0	0	0	0	0	0

32000行

得られたDEGセットを眺めてみると、A群（kidney）で高発現なものが78%を占め、B群（liver）で高発現なものが22%しかなかった。

hogeフォルダ中の「SupplementaryTable2_changed.txt」ファイル

偏りの原因は…

- ごく一部のB群（liver）で高発現の発現変動遺伝子（DEG）が存在していたため

真実（遺伝子4のみDEG）をうまく反映

生リードカウント

	A	B
遺伝子1	6	6
遺伝子2	4	4
遺伝子3	2	2
遺伝子4	3	18
sum	15	30

RPM補正後

A	B
400000	200000
266667	133333
133333	66666.7
200000	600000
1000000	1000000

TMM補正後

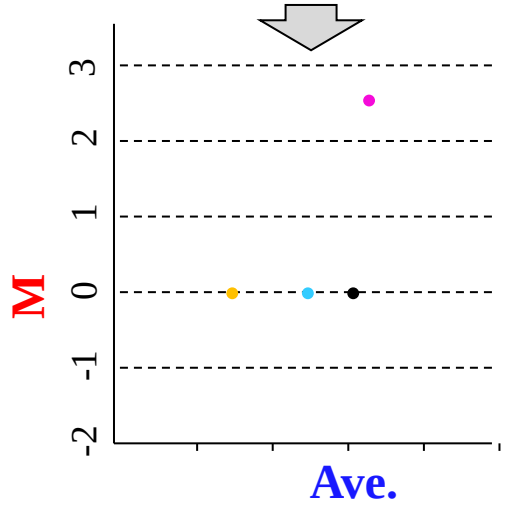
A	B
300000	300000
200000	200000
100000	100000
150000	900000
750000	1500000

$\log_2$(A)	$\log_2$(B)
18.19	18.19
17.61	17.61
16.61	16.61
17.19	19.78

縦軸（M） 横軸（A）

引き算	足し算
0.00	18.19
0.00	17.61
0.00	16.61
2.58	18.49

（liverで超高発現の）少数のDEGの影響により、その他の3遺伝子の発現レベルが過小評価されている
→A群（kidney）で高発現のDEGが多く検出される結果になっていた!



TMM論文の実際の図

このあたりのB群（liver）で高発現のDEGの存在により、それ以外がA群（kidney）で高発現側に偏っていることがわかる

A群(kidney) < B群(liver)

A群 = B群

A群(kidney) > B群(liver)

Sequence depth周辺の正規化法

- RPM (Mortazavi *et al*., *Nat. Methods*, **5**: 621-628, 2008)
  - RPKM(Reads per kilobase of exon per million mapped reads)の長さ補正を行わないバージョン
  - Reads per million mapped readsの略。
- TMM正規化 (Robinson and Oshlack, *Genome Biol*., **11**: R25, 2010)
  - Trimmed Mean of M valuesの略
  - 発現変動遺伝子の影響を排除した後に正規化を行うという戦略
- TbT正規化 (Kadota *et al*., *Algorithms Mol. Biol*., **7**: 5, 2012)
  - TMM法の改良版で、TMM-baySeq-TMMという3ステップで正規化を行う方法。
  - 1st stepで得られたTMM正規化係数を用いて、2nd step (baySeq)でDEG同定を行い、3rd step (TMM)ではDEGを排除した残りのデータでTMM正規化。DEGの影響を排除しつつもできるだけ多くのnon-DEGデータを用いて頑健に正規化係数を決めるという思想（DEG elimination strategy提唱論文）。
- iDEGES正規化（Sun *et al*., submitted）
  - DEG elimination strategy (DEGES) を一般化し、より高速且つ頑健にしたもの。TbTは「複製あり」のデータのみにしか対応していなかったが、「複製なし」データにも対応。
  - iDEGES/edgeR正規化法：「複製あり」データ正規化用。TMM-(edgeR-TMM)$_n$パイプライン
  - iDEGES/DESeq正規化法：「複製なし」データ正規化用。DESeq-(DESeq-DESeq)$_n$パイプライン

DEG elimination strategy (DEGES)
発現変動遺伝子の影響を排除した後に正規化を行うという戦略

DEGESって何デゲス？

～ アラフォー達の略称に関する議論 ～
門田:「DESで行くデス」
西山:「DEGESはいかが？」
門田:「面白くないので却下！」
西山:「左様デゲスか…DEGESって何デゲス？」
門田:「採用！」

- 概念図

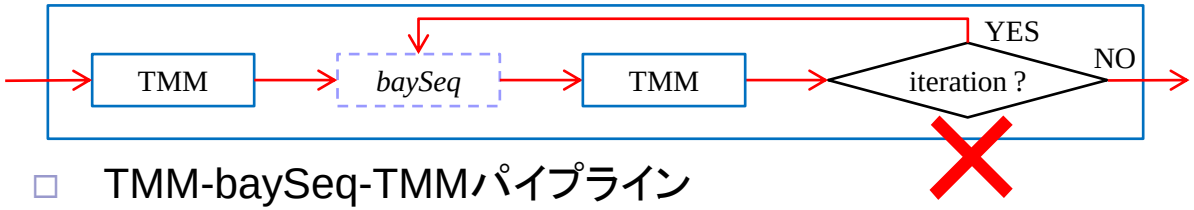
RNA-seqなどから得られるタグカウントデータの正規化をmulti-stepで行う概念の総称

DEG同定を正確に行うのが正規化の目的の一つではあるが、正規化時にDEGの存在自体がDEGとして同定されるのを阻むことがわかった（自爆テロ）。それゆえ、正規化時にDEGの検出を行って、non-DEGのみ利用するのがポイント

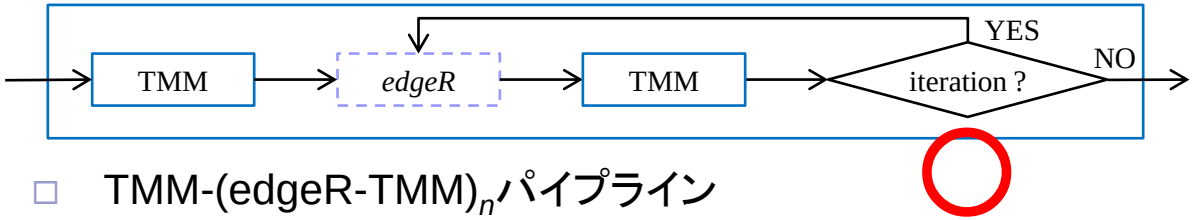
DEGESって何デゲス？

- DEGESのstep1-3で内部的に用いる方法は実用上なんでも?!よい

  - TbT正規化法（Kadota et al., 2012）



    - TMM-baySeq-TMMパイプライン
    - step2でbaySeqパッケージ中のDEG同定法（経験ベイズ）を利用しているため**遅い**…
    - Iterative TbT（step2-3を繰り返してより頑健な正規化係数を得る）は**非現実的**
  - iDEGES/*edgeR*正規化法 (Sun et al., submitted)

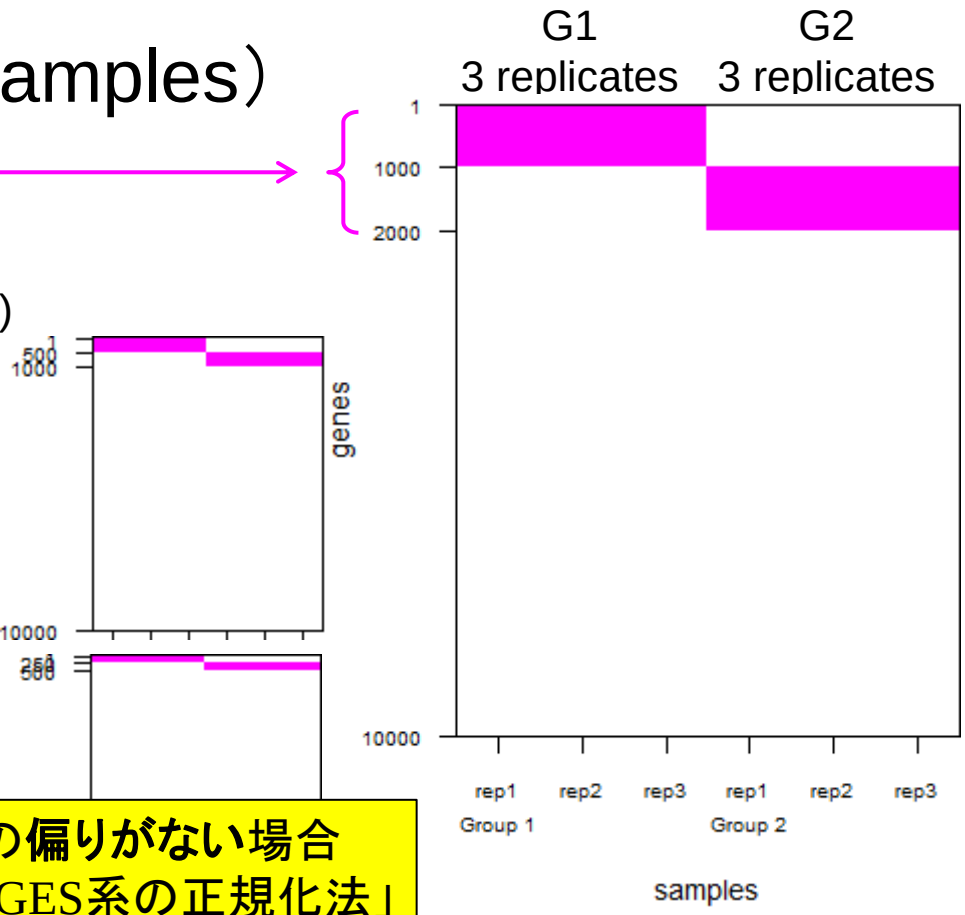


    - TMM-(edgeR-TMM)$_n$パイプライン
    - Step2でedgeRパッケージ中のDEG同定法（exact test）を利用しているため**速い**!
    - DEGESをiterativeに行う頑健なiDEGES（愛デゲス）パイプラインを利用可能

TCCパッケージ（ver. 1.0.0）に実装済み

どういうデータのときに有効デゲスか？

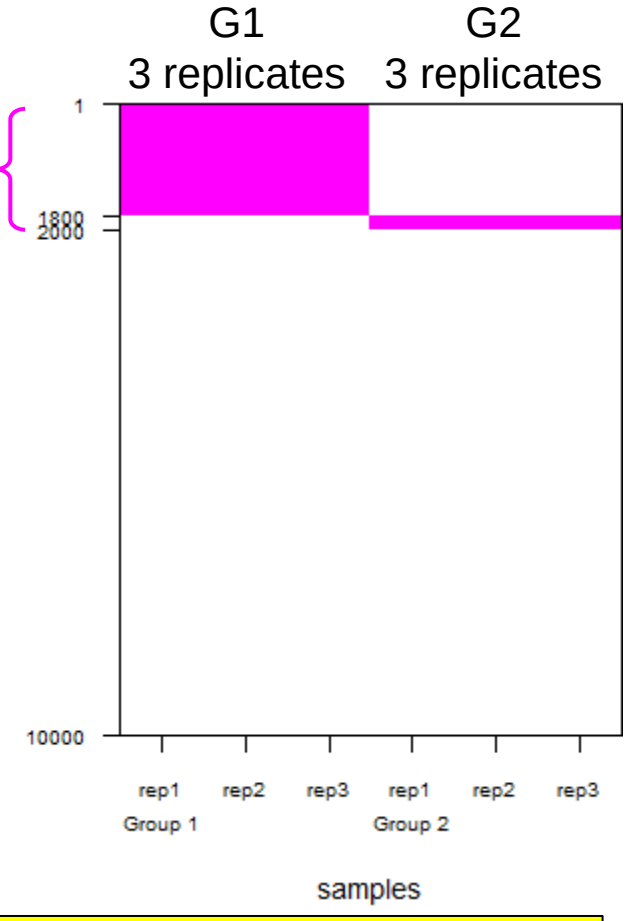
- 仮想データ（10,000 genes × 6 samples）
  - 2,000 DEGs (20%がDEG)
    - Group1 (G1)で高発現：gene1~1000 (**50%**)
    - Group2 (G2)で高発現：gene1001~2000 (**50%**)
  - 1,000 DEGs (10%がDEG)
    - Group1 (G1)で高発現：gene1~500 (**50%**)
    - Group2 (G2)で高発現：gene501~1000 (**50%**)
  - 500 DEGs (5%がDEG)
    - Group1 (G1)で高発現：gene1~250 (**50%**)
    - Group2 (G2)で高発現：gene251~500 (**50%**)

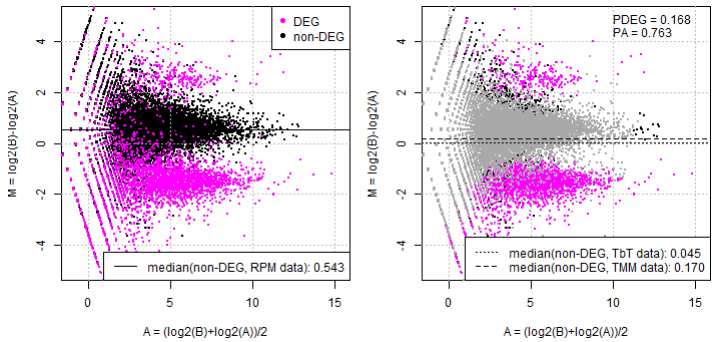


DEG数のGroup間での偏りがない場合「TMM正規化法」と「DEGES系の正規化法」の理論上の性能は互角デゲス。

どういうデータのときに有効デゲスか？

- 仮想データ（10,000 genes × 6 samples）
  - 2,000 DEGs (20%がDEG)
    - Group1 (G1)で高発現：gene1~1800 (**90%**)
    - Group2 (G2)で高発現：gene1801~2000 (**10%**)
  - 1,500 DEGs (15%がDEG)
    - Group1 (G1)で高発現：gene1~900 (**60%**)
    - Group2 (G2)で高発現：gene901~1500 (**40%**)
  - 1,000 DEGs (10%がDEG)
    - Group1 (G1)で高発現：gene1~200 (**20%**)
    - Group2 (G2)で高発現：gene201~1000 (**80%**)





DEGES系正規化法は、DEG数のGroup間での**偏りが大きいほど有効**なんデゲス！

研究目的別留意点

- 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場合
  - 「配列長」由来bias：長いほど沢山sequenceされる
  - 「GC含量」由来bias：カウント数の分布がGC含量依存的である
- **サンプル間**比較（sample A vs. Bなど）で、発現変動遺伝子（DEG）を調べたい場合
  - 「sequence depthの違い」：総リード数がx倍違うと全体的にx倍変動…
  - 「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…
  - RPM（CPM）正規化 → TMM正規化 → TbT正規化 → iDEGES正規化

DEGES系の方法が有効であるという根拠は？

総リード数を揃えるだけ

DEGを（正確には見積もらないので）多めにトリム

正規化の手順の中で同定したDEGをトリムすることでより頑健に

律速であったDEG同定部分の改良により、より頑健且つ高速に

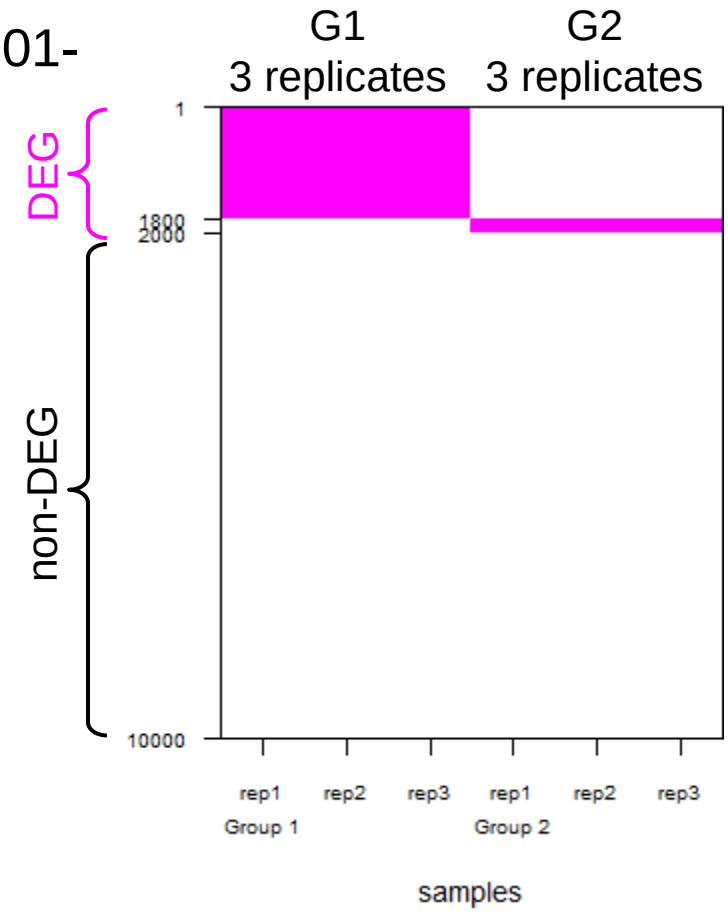
正規化後のデータのnon-DEGの分布

- よりよい正規化法ほど、正規化後にnon-DEGデータ（2,001-10,000行目）の分布が揃っているはず

デスクトップ – hoge - data_hypodata_3vs3.txt

	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	220	120	80	95	156
gene_9998	42	73	67	62	44	37
gene_9999	5	1	2	3	4	11
gene_10000	2	4	5	2	0	0

DEG
G1で高発現
G2で高発現
non-DEG



「non-DEGの$\log_2$(G2/G1)の中央値」が0に近いほどよい正規化法
$\log_2$(G2/G1) =（M-A plotの）M値

Sequence depth周辺の正規化法

- RPM (Mortazavi *et al*., *Nat. Methods*, **5**: 621-628, 2008)
  - RPKM(R
  - Reads pe

iDEGES/edgeR正規化後のデータから得られるnon-DEG由来median(M) 値 vs. TMM正規化後のデータから得られるnon-DEG由来median(M) 値。0に近いのは？

- TMM正規化 (Robinson and Oshlack, *Genome Biol*., **11**: R25, 2010)
  - Trimmed Mean of M valuesの略
  - 発現変動遺伝子（DEG）のデータ正規化時の悪影響を排除すべく、M-A plot上で周縁部にあるデータを使わずに正規化係数を決定する方法。
- TbT正規化 (Kadota *et al*., *Algorithms Mol. Biol*., **7**: 5, 2012)
  - TMM法の改良版で、TMM-baySeq-TMMという3ステップで正規化を行う方法。
  - 1st stepで得られたTMM正規化係数を用いて、2nd step (baySeq)でDEG同定を行い、3rd step (TMM)ではDEGを排除した残りのデータでTMM正規化。DEGの影響を排除しつつもできるだけ多くのnon-DEGデータを用いて頑健に正規化係数を決めるという思想（DEG elimination strategy提唱論文）。
- iDEGES正規化（Sun *et al*., submitted）
  - DEG elimination strategy (DEGES) を一般化し、より高速且つ頑健にしたもの。TbTは「複製あり」のデータのみにしか対応していなかったが、「複製なし」データにも対応。
  - iDEGES/edgeR正規化法：「複製あり」データ正規化用。TMM-(edgeR-TMM)$_n$パイプライン
  - iDEGES/DESeq正規化法：「複製なし」データ正規化用。DESeq-(DESeq-DESeq)$_n$パイプライン

- 前処理｜発現レベルの定量化について(last modified 2013/01/22) NEW
- 前処理｜正規化について(last modified 2013/01/24) NEW
- 前処理｜正規化(サンプル内)｜GC bias補正｜full quantile (FQ)正規化(Risso_2011)(last modified 2013/01/22) NEW
- 前処理｜正規化(サンプル内)｜全般｜GAM正規化(Zheng_2011)(last modified 2013/01/18)
- 前処理｜正規化(サンプル間)｜二群間用｜複製なし｜iDEGES/DESeq正規化(Sun_submitted)(last modified 2013/01/28
- 前処理｜正規化(サンプル間)｜二群間用｜複製あり｜iDEGES/edgeR正規化(Sun_submitted)(last modified 2013/01/24
- 前処理｜正規化(サンプル間)｜二群間用｜複製あり｜TbT正規化(Kadota_2012)(last modified 2013/01/23) NEW
- 前処理｜正規化(サンプル間)｜二群間用｜複製非依存｜TMM正規化(Robinson_2010)(last modified 2013/01/23) NEW
- 前処理
- 前処理
- 前処理
- 前処理

- **前処理｜正規化(サンプル間)｜二群間用｜複製あり｜iDEGES/edgeR正規化(Sun_submitted)**

iDEGES/edgeR法は、TbT法(Kadota et al., 2012)の改良版で、オリジナルのTbT法よりも100倍程度高速(デフォルトの場合)で理論性能は上です。TbT法の正規化の基本戦略は「正規化時に悪さをするDEGを正規化前に除いてしまえ(DEG elimination strategy; DEGES; でげす)」で、実体は「1.TMM法(Robinson and Oshlack, 2010) → 2.baySeq (Hardcast
ジ中の経
行うため
DEGES/
パッケー
はオリジ
TbT論文
規化係数
時間が非
の手順を
完結する
iDEGES/
程度計算
DEGES/e
ここでは、
本的に正
また、サン
ません(理

1. サンプルデータ13の10,000 genes×6 samplesの「複製あり」タグカウントデータ(data_hypodata_3vs3.txt)
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。

```
------    ここから    ------
in_f <- "data_hypodata_3vs3.txt"                    #読み込みたい発現データファイルを指定してin_fに格納
out_f <- "data_hypodata_3vs3_iDEGESedgeR.txt"       #出力ファイル名を指定
param_A <- 3                                        #A群(G1群)のサンプル数を指定
param_B <- 3                                        #B群(G2群)のサンプル数を指定
param1 <- 3                                         #TMM-(edgeR-TMM)nパイプラインのnの値(iterationの回数)

#必要なパッケージなどをロード
library(TCC)

#発現データの読み込みとTCCクラスオブジェクトの作成
data <- read.table(in_f, header=TRUE, row.names=1, sep="¥t", quote="")#発現データファイルの読み込み
tcc <- new("TCC", as.matrix(data), c(param_A, param_B))  #TCCクラスオブジェクトtccを作成

#iDEGES/edgeR正規化の実行
tcc <- calcNormFactors(tcc, iteration=param1)           #正規化を実行した結果をtccに格納

#正規化後のデータをファイルに出力
normalized.count <- getNormalizedData(tcc)              #正規化後のデータを取り出してnormalized.countに格納
tmp <- cbind(rownames(normalized.count), normalized.count)  #「rownames情報」と「正規化後のデータ」を列方向で結合
write.table(tmp, out_f, sep="¥t", append=F, quote=F, row.names=F)#tmpの中身をout_fで指定したファイル名で保存
```

「…ファイルに出力」までをコピペすれば、
iDEGES/edgeR正規化後のデータが得られます

1. サンプルデータ13の10,000 genes×6 samplesの「複製あり」タグカウントデータ(data_hypo
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
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。
------　ここから　------

```
in_f <- "data_hypodata_3vs3.txt"
out_f <- "data_hypodata_3vs3_iDEGESe
param_A <- 3
param_B <- 3
param1 <- 3

#必要なパッケージなどをロード
library(TCC)

#発現データの読み込みとTCCクラスオブ
data <- read.table(in_f, header=TRUE
tcc <- new("TCC", as.matrix(data), c

#iDEGES/edgeR
tcc <- calcNo

#正規化後のデ
normalized.co
tmp <- cbind(
write.table(t
```

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

出力ファイル（の一部）

	G1_rep1	G1_rep2	G1_rep3	G2_rep1	G2_rep2	G2_rep3
gene_1	35.5	55.8	142.0	2.0	1.0	0.0
gene_2	82.8	151.4	122.3	52.5	37.5	28.3
gene_3	583.4	836.5	788.8	152.5	260.6	201.9
gene_4	0.0	8.0	3.9	1.0	1.0	3.0
gene_5	31.5	31.9	0.0	1.0	1.0	0.0
gene_6	3.9	0.0	23.7	4.0	10.1	0.0
gene_7	339.0	239.0	232.7	76.8	67.9	71.7
gene_8	1245.7	780.7	1045.2	214.1	185.5	180.7
gene_9	90.7	87.6	82.8	21.2	22.3	33.3
gene_10	63.1	47.8	94.7	24.2	13.2	12.1

R Console

```
> param_A <- 3                                                    #A群(G1群$
> param_B <- 3                                                    #B群(G2群$
> param1 <- 3                                                     #TMM-(edg$
>
> #必要なパッケージなどをロード
> library(TCC)                                                    #パッケー$
>
> #発現データの読み込みとTCCクラスオブジェクトの作成
> data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="")#発$
> tcc <- new("TCC", as.matrix(data), c(param_A, param_B))          #TCCクラ$
>
> #iDEGES/edgeR正規化の実行
> tcc <- calcNormFactors(tcc, iteration=param1)                   #正規化を$
TCC::INFO: Calculating normalization factors using
TCC::INFO: (iDEGES pipeline : tmm - [ edger - tmm ]
TCC::INFO: Done.
>
> #正規化後のデータをファイルに出力
> normalized.count <- getNormalizedData(tcc)
> tmp <- cbind(rownames(normalized.count), normalized.count)      #「rownam$
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)#tmpの中$
>
>
```

ちなみに、出力ファイルは「行名部分」と「正規化後のデータ部分」をcbind関数を用いて列方向で結合したものなので…

R Console

```
> param1 <- 3                                                    #TMM-(e
>
> #必要なパッケージなどをロード
> library(TCC)                                                   #パッケー
>
> #発現データの読み込みとTCCクラスオブジェクトの作成
> data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="")#発
> tcc <- new("TCC", as.matrix(data), c(param_A, param_B))         #TCCクラ
>
> #iDEGES/edgeR正規化の実行
> tcc <- calcNormFactors(tcc, iteration=param1)                  #正規化
TCC::INFO: Calculating normalization factors using DEGES
TCC::INFO: (iDEGES pipeline : tmm - [ edger - tmm ] X 3 )
TCC::INFO: Done.
>
> #正規化後のデータをファイルに出力
> normalized.count <- getNormalizedData(tcc)                     #正規化後$
> tmp <- cbind(rownames(normalized.count), normalized.count)     #「rownam$
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)#tmpの中$
>
> tmp <- cbind(rownames(normalized.count), round(normalized.count))
> write.table(tmp, "gege.txt", sep="\t", append=F, quote=F, row.names=F)
>
```

出力ファイル（の一部）

	G1_rep1	G1_rep2	G1_rep3	G2_rep1	G2_rep2	G2_rep3
gene_1	35.5	55.8	142.0	2.0	1.0	0.0
gene_2	82.8	151.4	122.3	52.5	37.5	28.3
gene_3	583.4	836.5	788.8	152.5	260.6	201.9
gene_4	0.0	8.0	3.9	1.0	1.0	3.0
gene_5	31.5	31.9	0.0	1.0	1.0	0.0
gene_6	3.9	0.0	23.7	4.0	10.1	0.0
gene_7	339.0	239.0	232.7	76.8	67.9	71.7
gene_8	1245.7	780.7	1045.2	214.1	185.5	180.7
gene_9	90.7	87.6	82.8	21.2	22.3	33.3
gene_10	63.1	47.8	94.7	24.2	13.2	12.1

gege.txt

	G1_rep1	G1_rep2	G1_rep3	G2_rep1	G2_rep2	G2_rep3
gene_1	35	56	142	2	1	0
gene_2	83	151	122	53	38	28
gene_3	583	836	789	152	261	202
gene_4	0	8	4	1	1	3
gene_5	32	32	0	1	1	0
gene_6	4	0	24	4	10	0
gene_7	339	239	233	77	68	72
gene_8	1246	781	1045	214	186	181
gene_9	91	88	83	21	22	33
gene_10	63	48	95	24	13	12

「正規化後のデータ部分」にround関数を適用した結果を出力することによって、最も近い整数値に丸めることができます

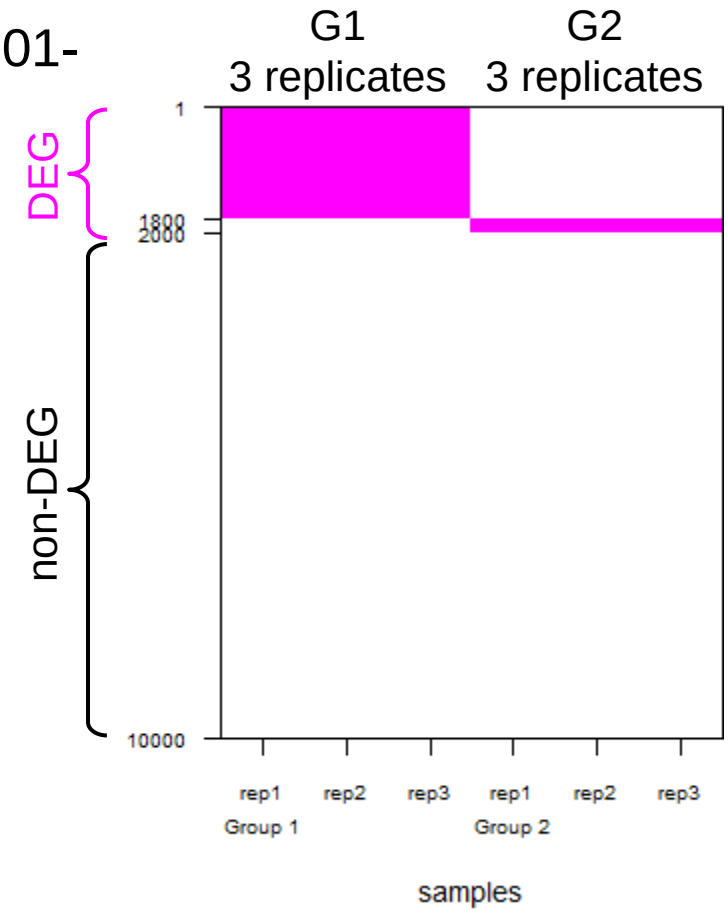
正規化後のデータでM-A plot

- よりよい正規化法ほど、正規化後にnon-DEGデータ（2,001-10,000行目）の分布が揃っているはず

data_hypodata_3vs3_iDEGESedgeR.txt

	G1_rep1	G1_rep2	G1_rep3	G2_rep1	G2_rep2	G2_rep3
gene_1	35.5	55.8	142.0	2.0	1.0	0.0
gene_2	82.8	151.4	122.3	52.5	37.5	28.3
gene_3	583.4	836.5	788.8	152.5	260.6	201.9
gene_4	0.0	8.0	3.9	1.0	1.0	3.0
gene_5	31.5	31.9	0.0	1.0	1.0	0.0
...						
gene_1801	33.5	85.6	23.7	286.8	182.5	367.5
gene_1802	4.9	1.0	3.0	0.0	162.2	24.2
gene_1803	56.2	55.8	50.3	250.5	194.7	222.1
gene_1804	28.6	24.9	31.6	129.3	206.8	161.6
gene_1805	41.4	28.9	43.4	185.8	158.2	92.9
...						
gene_2001	3.9	8.0	8.9	13.1	12.2	4.0
gene_2002	86.7	138.4	39.4	22.2	44.6	21.2
gene_2003	919.5	664.2	455.5	897.8	401.5	447.3
gene_2004	47.3	36.8	13.8	36.4	57.8	71.7
gene_2005	285.8	336.6	545.3	322.2	212.9	508.9
...						
gene_9996	105.5	66.7	102.5	35.3	65.9	45.4
gene_9997	142.9	219.1	118.3	80.8	96.3	157.5
gene_9998	41.4	72.7	66.1	62.6	44.6	37.4
gene_9999	4.9	1.0	2.0	3.0	4.1	11.1
gene_10000	2.0	4.0	4.9	2.0	0.0	0.0

DEG
G1で高発現
G2で高発現
non-DEG



「non-DEGの$\log_2$(G2/G1)の中央値」が0に近いほどよい正規化法
$\log_2$(G2/G1) =（M-A plotの）M値

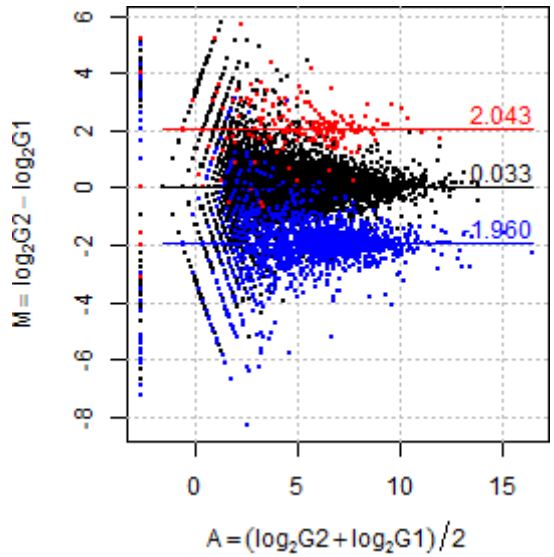
M-A plot（基本形）

- data_hypodata_3vs3_iDEGESedgeR.txtのM-A plotを作成

	G1_rep1	G1_rep2	G1_rep3	G2_rep1	G2_rep2	G2_rep3
gene_1	35.5	55.8	142.0	2.0	1.0	0.0
gene_2	82.8	151.4	122.3	52.5	37.5	28.3
gene_3	583.4	836.5	788.8	152.5	260.6	201.9
…						
gene_22	0.0	15.9	0.0	0.0	0.0	0.0
…						
gene_36	0.0	0.0	0.0	2.0	0.0	0.0
..						
gene_95	0.0	0.0	0.0	0.0	0.0	0.0
…						
gene_9999	4.9	1.0	2.0	3.0	4.1	11.1
gene_10000	2.0	4.0	4.9	2.0	0.0	0.0

G1	G2
77.7	1.0
118.8	39.4
736.2	205.0
…	…
5.3	0.0
…	…
0.0	0.7
…	…
0.0	0.0
…	…
2.6	6.1
3.6	0.7

M	A
-6.265	3.148
-1.591	6.097
-1.845	8.602
…	…
#NUM!	#NUM!
…	…
#NUM!	#NUM!
…	…
#NUM!	#NUM!
…	…
1.204	1.998
-2.430	0.644



どちらかの群で0になっているものは（特にM値が-Inf or +Infとなるので）、M-A plot不可能…

M-A plot（TCCパッケージの0カウント対策）

- data_hypodata_3vs3_iDEGESedgeR.txtのM-A plotを作成

	G1_rep1	G1_rep2	G1_rep3	G2_rep1	G2_rep2	G2_rep3
gene_1	35.5	55.8	142.0	2.0	1.0	0.0
gene_2	82.8	151.4	122.3	52.5	37.5	28.3
gene_3	583.4	836.5	788.8	152.5	260.6	201.9
…						
gene_22	0.0	15.9	0.0	0.0	0.0	0.0
…						
gene_36	0.0	0.0	0.0	2.0	0.0	0.0
..						
gene_95	0.0	0.0	0.0	0.0	0.0	0.0
…						
gene_9999	4.9	1.0	2.0	3.0	4.1	11.1
gene_10000	2.0	4.0	4.9	2.0	0.0	0.0

G1	G2
77.7	1.0
118.8	39.4
736.2	205.0
…	…
5.3	0.0
…	…
0.0	0.7
…	…
0.0	0.0
…	…
2.6	6.1
3.6	0.7

M	A
-6.265	3.148
-1.591	6.097
-1.845	8.602
…	…
#NUM!	#NUM!
…	…
#NUM!	#NUM!
…	…
#NUM!	#NUM!
…	…
1.204	1.998
-2.430	0.644

②

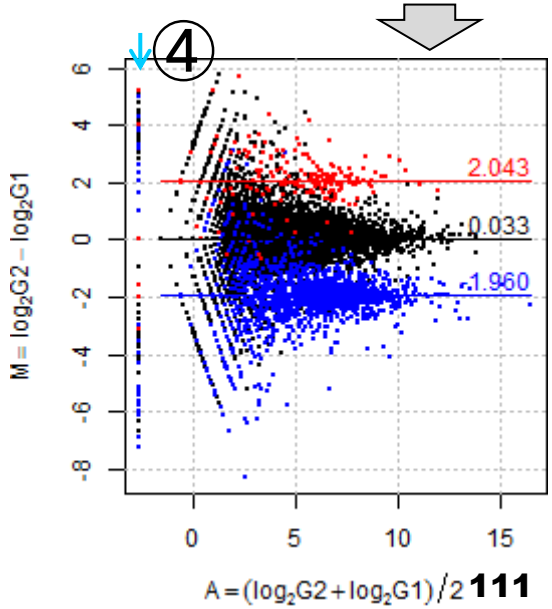
G1	G2
77.743	1.011
118.804	39.433
736.239	204.998
…	…
5.311	0.337
…	…
0.329	0.673
…	…
0.329	0.337
…	…
2.632	6.064
3.628	0.673

③

M	A
-6.265	3.148
-1.591	6.097
-1.845	8.602
…	…
-3.980	0.419
…	…
1.035	-1.088
…	…
0.035	-1.588
…	…
1.204	1.998
-2.430	0.644

① 0.329 0.337

④



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

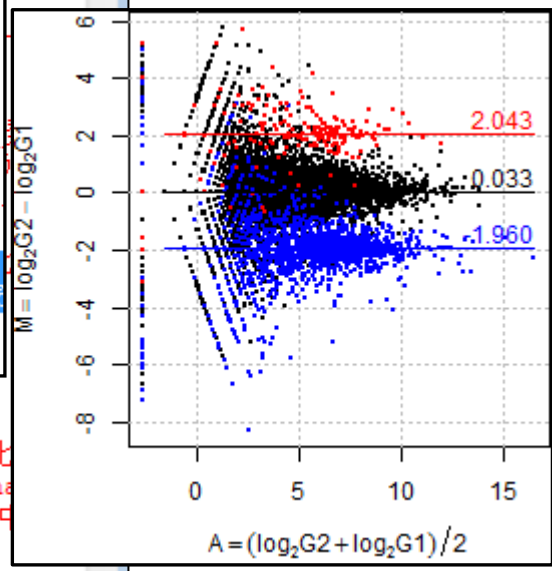
1. サンプルデータ13の10,000 genes×6 samplesの「複製あり」タグカウントデータ(data_hypodata_3vs3.txt
Biological replicatesを模倣したシミュレーションデータ(G1群3サンプル vs. G2群3サンプル)です。
gene_1～gene_2000までがDEG（最初の1800個がG1群で高発現、残りの200個がG2群で高発現）
gene_2001～gene_10000までがnon-DEGであることが既知です。
------　ここから　------

```
#以下は（こんなこともできますという）おまけ
#正規化後のデータでM-A plotを描画
plot(tcc)                                                    #M-A plotを描画

#正規化後のデータでnon-DEGsに相当する2001-10000行目のデータのみを用いて(logスケールで)boxplot
nonDEG <- 2001:10000                                         #non-DEGの位置情報をnonDEGに格納
boxplot(log(normalized.count[nonDEG, ]))                     #boxplot （non-DEGなので分布が揃っ

#正規化後のデータでnon-DEGsに相当する2001-10000行目のデータのみについて要約統計量を表示（サンプル間
summary(normalized.count[nonDEG, ])                          #各列(つまりサンプル)の要約統計量
apply(normalized.count[nonDEG, ], 2, median)                 #各列(つまりサンプル)のmedian(中央

#iDEGES/edgeR正規化のstep2で検出されたpotential DEGの結果(step3で使われないものたち)を表示
table(tcc$private$DEGES.potentialDEG)                        #0 (nonDEGに相当)が8,679個、1 (G1

#iDEGES/edgeR正規化係数を表示
tcc$norm.factors                                             #正規化係数を表示(0.7469719 0.834

#このデータは「答え」がわかっているものなので、答え(DEG or non-DEG)の情報込みで正規化後のデータのM
tcc$private$simulation <- TRUE                               #おまじない
tcc$simulation$trueDEG <- c(rep                              #真の情報からなるベクトルをtcc
plot(tcc, median.lines=TRUE)                                 #log-ratio(縦軸の値)のmedian値をゼ

------　ここまで　------
```

切り取り(T)
コピー(C)
貼り付け

1.の最後の三行分をコピペしてM-A plotを描画

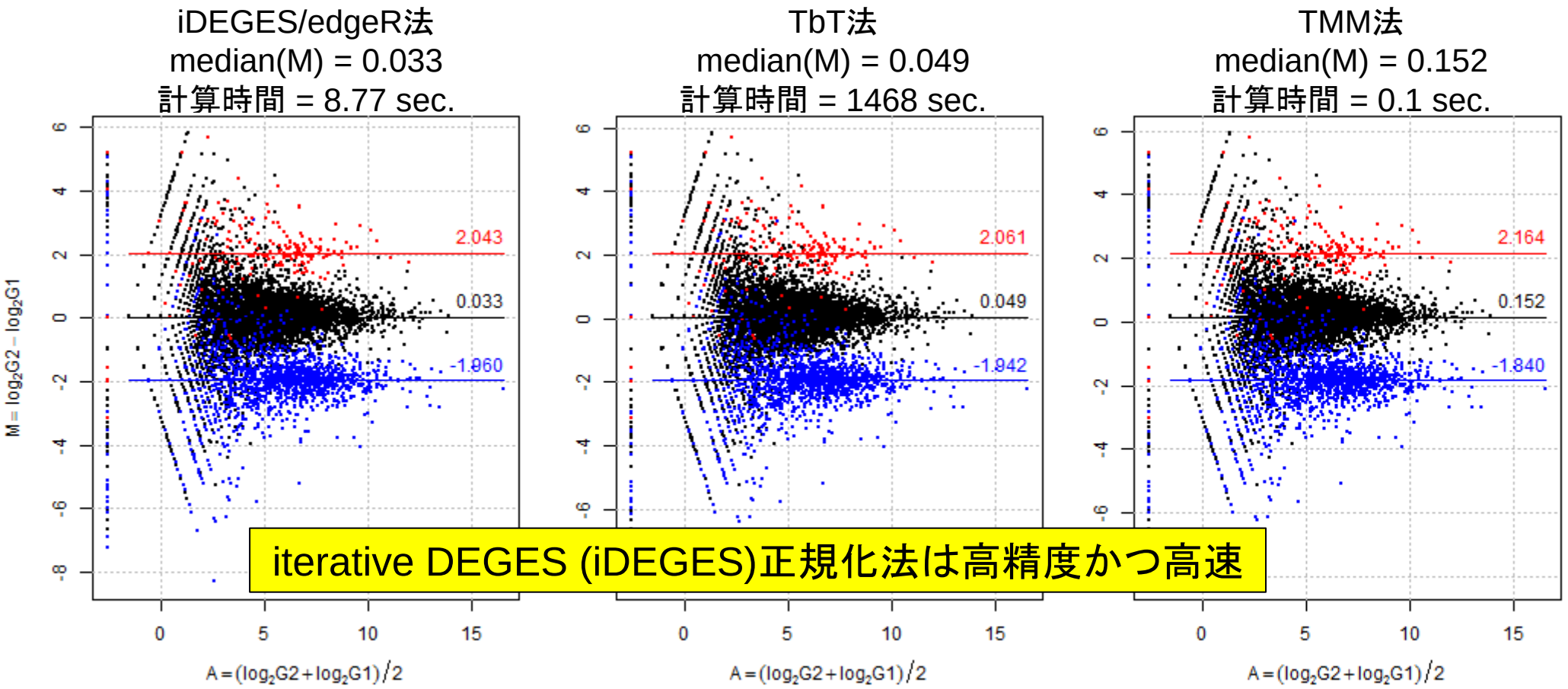
```
> normalized.count <- getNormalizedData(tcc)                 #正規化
> tmp <- cbind(rownames(normalized.count), normalized.count)  #「rowna
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)#tmpの中
>
> tcc$private$simulation <- TRUE                              #おまじな$
> tcc$simulation$trueDEG <- c(rep(1, 1800), rep(2, 200), rep(0, 8000))#真の$
> plot(tcc, median.lines=TRUE)                                #log-rati$
>
>
```



- 前処理｜正規化(サンプル間)｜二群間用｜複製あり｜iDEGES/edgeR正規化(Sun_submitted)
- 前処理｜正規化(サンプル間)｜二群間用｜複製あり｜TbT正規化(Kadota_2012)(last modified
- 前処理｜正規化(サンプル間)｜二群間用｜複製非依存｜TMM正規化(Robinson_2010)(last m

性能評価(仮想データ:偏りあり)

- データ:non-DEG: 8000個、G1で高発現のDEG: 1800個、G2で高発現のDEG: 200個
- 評価基準:non-DEGのmedian(M)値が0に近いほどよい正規化法

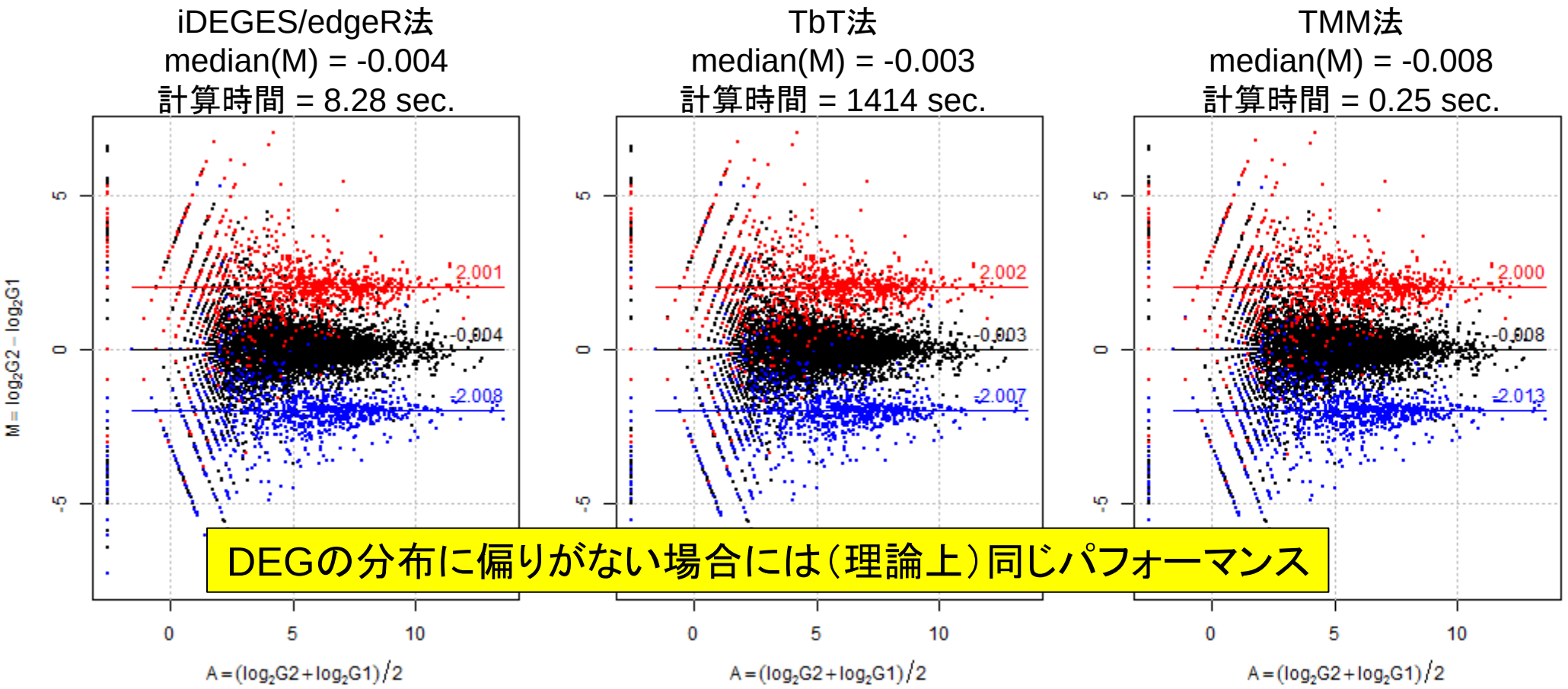


iterative DEGES (iDEGES)正規化法は高精度かつ高速

- 前処理｜正規化(サンプル間)｜二群間用｜複製あり｜iDEGES/edgeR正規化(Sun_submitted)
- 前処理｜正規化(サンプル間)｜二群間用｜複製あり｜TbT正規化(Kadota_2012)(last modified
- 前処理｜正規化(サンプル間)｜二群間用｜複製非依存｜TMM正規化(Robinson_2010)(last m

性能評価（仮想データ:偏りなし）

- データ：non-DEG: 8000個、G1で高発現のDEG: 1000個、G2で高発現のDEG: 1000個
- 評価基準：non-DEGのmedian(M)値が0に近いほどよい正規化法



DEGの分布に偏りがない場合には（理論上）同じパフォーマンス

TCC（ver. 1.0.0）の主な機能

- DEG elimination strategy (DEGES)に基づくデータ**正規化**法を実装

  - 複製ありデータ用

	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
…						

    - TbT正規化法（Kadota et al., 2012）: TMM-baySeq-TMMパイプライン
    - **iDEGES/edgeR正規化法**：TMM-(edgeR-TMM)$_n$パイプライン
  - 複製なしデータ用

	G1_rep1	G2_rep1
gene_1	36	2
gene_2	84	52
gene_3	592	151
gene_4	0	1
…		

    - **iDEGES/DESeq正規化法**：DESeq-(DESeq-DESeq)$_n$パイプライン
- 既存パッケージ中の**DEG検出**法を呼び出して利用可能
  - （正規化のところと同じく）edgeR, baySeq, DESeqパッケージ中の関数群を内部的に利用
- シミュレーションデータ作成機能
  - 二群（複製あり and/or なし）、三群、四群、、、多群
  - 発現変動の度合いを調整可能（Fold-Change, gamma分布）

TCCの売りは（既存の手法を組み合わせることで）
データ正規化部分の精度向上に貢献

二群間比較用Rパッケージ

- *DEGSeq* (Wang *et al*., *Bioinformatics*, **26**: 136-138, 2010)
- *edgeR* (Robinson *et al*., *Bioinformatics*, **26**: 139-140, 2010)
- *GPseq* (Srivastava and Chen, *Nucleic Acids Res*., **38**: e170, 2010)
- *baySeq* (Hardcastle and Kelly, *BMC Bioinformatics*, **11**: 422, 2010)
- *DESeq* (Anders and Huber, *Genome Biol*., **11**: R106, 2010)
- *NBPSeq* (Di *et al*., *SAGMB*, **10**: article24, 2011)
- *NOISeq* (Tarazona *et al*., *Genome Res*., **21**: 2213-2223, 2011)
- *BitSeq* (Glaus *et al*., *Bioinformatics*, **28**: 1721-1728, 2012)
- *TCC* (Sun *et al*., submitted)

「(TCC中で利用可能な)TbT正規化法」と「edgeR, DESeq, baySeq, NBPSeq中のDEG検出法」との組合せが有効であることは既に実証済み(Kadota et al., 2012)

R以外

- *GFOLD* (Feng *et al*., *Bioinformatics*, **28**: 2782-2788, 2012)
- *Cuffdiff 2* (Trapnell *et al*., *Nat. Biotechnol*., **31**: 46-54, 2013)

TCCで正規化→DEG同定まで(複製あり)

- iDEGES/edgeR正規化後にedgeRパッケージ中のDEG同定法を利用する場合

- 解析 | NGS(RNA-seq) | 発現変動遺伝子 | 二群間 | について (last modified 2013/01/22) NEW
- 解析 | NGS(RNA-seq) | 発現変動遺伝子 | 二群間 | 複製なし | DESeq with iDEGES/DESeq (Sun_submitted) (last modified 2013/01/28)rec
- 解析 | NGS(RNA-seq) | 発現変動遺伝子 | 二群間 | 複製なし | DESeq (Anders_2010) (last modified 2013/01/28) NEW
- 解析 | NGS(RNA-seq) | 発現変動遺伝子 | 二群間 | 複製あり | edgeR with iDEGES/edgeR (Sun_submitted) (last modified 2013/01/28)reco
- 解析 | NGS(RNA-seq) | 発現変動遺伝子 | 二群間 | 複製あり | edgeR with DEGES/TbT (Kadota_2012) (last modified 2013/01/28) NEW
- 解析 | NGS(RNA-seq) | 発現変動遺伝子 | 二群間 | 複製あり | edgeR coupled with TbT normalization (上と同じで昔の記述) (last modified 2
- 解析 | NGS(RNA-seq) | 発現変動遺伝子 | 二群間 | 複製あり | NBPSeq coupled with TbT normalization (Kadota 2012) (last modified 2012/
- 解析 |
- 解析 |
- 解析 |

• 解析 | NGS(RNA-seq) | 発現変動遺伝子 | 二群間 | 複製あり | edgeR with iDEGES/edgeR (Sun_submitted)

iDEGES/edgeR正規化(Sun_submitted)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)検出を行うやり方を示します。全てTCC (≥ ver. 1.0.0)パッケージ内で完結します。
ちなみに、param_DEmethodのところでの"edger"は「DEG検出の手段としてedgeRパッケージ中の方法を利用する」ことに相当しますが、以下に示す他の二つの選択肢もあります:
・DEG検出はDESeqパッケージ中のnegative binomial testにしたい場合:"deseq"
・DEG検出はbaySeqパッケージ中のempirical Bayesにしたい場合:"bayseq"
また、param_FDRでは0.05を指定していますが、これは「**DEGと判定したものの中で**どの程度の数の偽物(本当はDEGではないのにDEGと判定されたもの)を許容するか？」というfalse discovery rate (FDRと略す)の閾値を指定するところです。0.05だと5%偽物が含まれることを許容することに相当し、一般にこの数を低く設定するほどDEGと判定する数が少なくなります。
ちなみによくp値の閾値(有意水準; significance level)と混同しがちですが、例えばp-value < 0.05は「**non-DEGの中で**(本当はDEGではないにもかかわらず)DEGと判定してしまうのが5%程度になるような閾値」に相当します。つまり、「分子は同じですが、分母が異なる」ということです。
なぜ、p値を使わずにFDR(p値に対応する正しい用語は本当はq-value)がよく用いられるかというと、例えばある10000遺伝子からなるランダムデータの二群間比較を行って、p < 0.05の閾値を満たすものをDEGと判定するとしましょう。p値の定義からいって、大体5%(10000×0.05 = 500個)程度がDEGと判定されます。この得られた500個が本当のDEG...ではないですよね？つまり、実データの場合で、「有意水準を満たすDEG数」が「搭載遺伝子数×設定した有意水準」以下であればそのデータセット中にDEGはないという判定がくだされるべき、という常識的な判断を下せなければなりません。逆に、10000遺伝子からなる実データの二群間比較を行って、p < 0.05の閾値を満たすものが1200個あったとしましょう。この中には偶然にDEGと判定されたものが500個程度含まれると考えるべきですが、残りの700個程度は本物と考えていいですよね。言い換えると1200個中(500/1200)×100 = 41.6667%がfalse positiveということであり、「FDR p < 0.416667を満たす遺伝子数は1200個あった」という言い方ができます。実用上は、上記事柄が理解できていれば十分でしょう。
つまり
「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「複製あり」タグカウントデータ(data_hypodata_3vs3.txt)

1.をやってみましょう。

1. サンプルデータ13の10,000 genes×6 samplesの「複製あり」タグカウントデータ(data_hypodata_3vs3.txt)
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。

```
------    ここから    ------
in_f <- "data_hypodata_3vs3.txt"                                    #読み込みたい発現データファイルを指定してin_fに格納
out_f <- "data_hypodata_3vs3_iDEGESedgeR_DE.txt"                    #出力ファイル名を指定
param_A <- 3                                                        #A群(G1群)のサンプル数を指定
param_B <- 3                                                        #B群(G2群)のサンプル数を指定
param1 <- 3                                                         #正規化におけるTMM-(edgeR-TMM)nパイプラインのnの値(iterationの回数)
param_DEmethod <- "edger"                                           #DEG検出法を指定
param_FDR <- 0.05                                                   #DEG検出時のfalse discovery rate (FDR)閾値を指定

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

#発現データの読み込みとTCCクラスオブジェクトの作成
data <- read.table(in_f, header=TRUE, row.names=1, sep="¥t", quote="")#発現データファイルの読み込み
tcc <- new("TCC", as.matrix(data), c(param_A, param_B))             #TCCクラスオブジェクトtccを作成

#iDEGES/edgeR正規化の実行
tcc <- calcNormFactors(tcc, iteration=param1)                       #正規化を実行した結果をtccに格納

#DEG検出の実行と結果の抽出
tcc <- estimateDE(tcc, test.method=param_DEmethod, FDR=param_FDR)#DEG検出を実行した結果をtccに格納
result <- getResult(tcc, sort=FALSE)                               #p値などの結果を抽出してをresultに格納

#結果をまとめたものをファイルに出力
tmp <- cbind(rownames(tcc$count), tcc$count, result)                #「rownames情報」、「カウントデータ」、「DEG検出結果」を列方向で結合
write.table(tmp, out_f, sep="¥t", append=F, quote=F, row.names=F)#tmpの中身をout_fで指定したファイル名で保存


#以下は（こんなこともできますという）おまけ
#正規化後のデータでM-A plotを描画
plot(tcc, FDR=param_FDR)                                            #param_FDRで指定した閾値を満たすDEGをマゼンタ色にしてM-A plotを描画
```

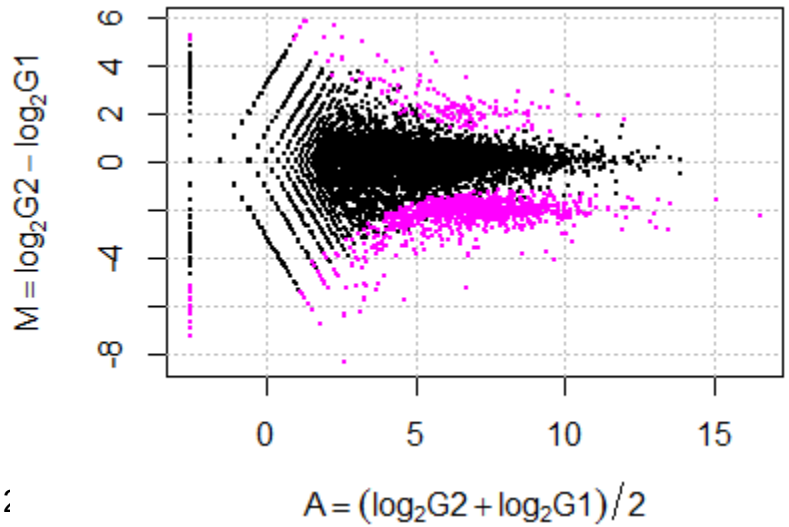
コピペ

出力ファイルの説明

入力データ

p-valueとその順位

rowname	G1_rep1	G1_rep2	G1_rep3	G2_rep1	G2_rep2	G2_rep3	id	a.value	m.value	p.value	q.value	rank	estimatedDEG
gene_1	36	56	144	2	1	0	gene_1	3.15	−6.26	6.48E−11	5.79E−08	11	1
gene_2	84	152	124	52	37	28	gene_2	6.10	−1.59	0.000974	0.0103	946	1
gene_3	592	840	800	151	257	200	gene_3	8.60	−1.84	6.82E−06	0.000151	452	1
gene_4	0	8	4	1	1	3	gene_4	1.37	−1.24	0.487688	1	4529	0
gene_5	32	32	0	1	1	0	gene_5	1.92	−4.97	0.001454	0.014614	995	1
gene_6	4	0	24	4	10	0	gene_6	2.72	−0.96	0.512219	1	4704	0
gene_7	344	240	236	76	67	71	gene_7	7.13	−1.91	5.29E−06	0.000123	429	1
gene_8	1264	784	1060	212	183	179	gene_8	8.80	−2.40	4.14E−09	9.20E−07	45	1
gene_9	92	88	84	21	22	33	gene_9	5.56	−1.76	0.000894	0.009578	933	1
gene_10	64	48	96	24	13	12	gene_10	5.07	−2.05	0.000982	0.010375	947	1
…													



M-A plotのA値とM値

q-value

(param_FDRで指定した)FDR閾値(<0.05)を満たすDEG。*q*-value < 0.05のものが0以外の値をとる。1:G1で高発現、2:G2で高発現。

http://www.iu.a.u-tokyo.ac.jp/~kadota/TCC/

TCC関連参考ウェブページ

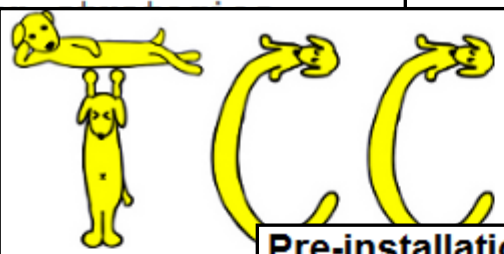
TCC: TCC: Differential expression analysis for tag count data with robust normalizatio

This package provides functions for performing differential expr
expressed gene elimination strategy. A simple unified interface i
functions to calculate normalization factors and estimate differen
edgeR, baySeq, and DESeq packages. The appropriate combi
more robust and accurate estimation performed easily than dire
Functions to produce simulation data under various conditions a
provided.

Version:	1.0.0
Depends:	R (≥ 2.15), edgeR, baySeq, DESeq, ROC, m
Published:	2013-01-10
Author:	Sun Jianqiang, Tomoaki Nishiyama, Kentaro S
Maintainer:	Tomoaki Nishiyama <tomoakin at staff.kanaza
License:	GPL-2
Copyright:	Authors listed above
URL:	http://www.iu.a.u-tokyo.ac.jp/~kadota/TCC
Citation:	TCC citation info
CRAN checks:	TCC results

Downloads:

Package source:	TCC_1.0.0.tar.gz
MacOS X binary:	not available, see check log.
Windows binary:	TCC_1.0.0.zip
Reference manual:	TCC.pdf
News/ChangeLog:	NEWS



TCC: an R p
with robust

The R package, T
to perform differer
expression analys
throughput sequer
gene expression.
method (TbT; Kac
replicates. The str
expressed genes
demonstrated that
normalization is e
DEGs are top-ran
integrated analysi
compared with oth

Pre-installation

The main functionalities in **TCC** consist of combinations of functions from the above three R/Bioconductor packages (edgeR, DESeq, and baySeq). If those packages have not been installed in your R environment, you need to enter the following commands after starting R:

```
source("http://bioconductor.org/biocLite.R")
biocLite(c("edgeR", "baySeq", "DESeq", "ROC"))
```

Installation

This package is available at the CRAN repository. To install the package, visit the repository for TCC or enter the following command after starting R:

```
install.packages("TCC", type = "source")
```

Bioconductor likeなUser's Guide (Vignette)もあります

Documentation

User's Guide　　R script
Reference Manual

その他

• 解析｜NGS(RNA-seq)｜発現変動遺伝子｜二群間｜について

イントロ

私のマイ
す。NGS
ここでの
は用いる

NGSデー
生のカウ
項分布(

一般的な
確率をp
NGSに置
うことに

ポアソン
す。つま
λ=np)が
というわ
デルを仮
ちなみに
です。
いわゆる

technical replicatesとbiological replicatesの意味合いについて、もうすこしざっくりと説明します。例えば「人間である私の体内の二つの組織間(例えば大脳(Brain; B)と皮膚(Skin; S))で発現変動している遺伝子を検出したい」という目的があったとします。同じ組織でもばらつきを考慮する必要があるので大脳由来サンプルをいくつかに分割して(これがtechnical replicatesという意味)データを取得します。皮膚由来サンプルについても同様です。これで得られたデータは(B1, B2, ...) vs. (S1, S2, ...)のようなreplicatesありのデータになります。この私一個人のtechnical replicatesデータから発現変動遺伝子が得られたとします。一般的に「私一個人内でのデータのばらつき(technical variation)」は「同じ人間だけど複数の別人の大脳(or 皮膚)由来サンプル間のばらつき(biological variation)」よりも小さいです。ですので、technical replicates由来データで発現変動遺伝子(Differentially Expressed Genes; DEGs)を同定した結果は「そんなあほな！！」というくらいのDEGs数がレポートされます。ここで注意しなければいけないのは、この結果は「私一個人の二つのサンプル間で発現変動している遺伝子がこれだけあった」ということを言っているだけで、「(私が所属する)人間全般に言えることではない！」という点です。そしてもちろん一般には「私個人の事象についてではなく人間(ある特定の生物種)全般にいえる事象に興味がある」わけです。それゆえ(biological replicatesのデータでないとユニバーサルな事象を理解するのが難しいために)ある程度理解が進んだ人はbiological replicatesが重要だと言っているわけです。

「モデルだ～分布だ～」ということと「ユーザーがプログラムを実行して得られる結果(発現変動遺伝子数や特徴)」との関係についてですが、重要なのは「比較している二群間でどの程度差があれば発現変動していると判断するか？」です。これをうまく表現するために、同一サンプルの(biological) replicatesデータから発現変動していないデータがどの程度ばらついているかをまず見積もります。(このときにどういう数式でばらつきを見積もるのか？というところでいろんな選択肢が存在するがために沢山のRパッケージが存在しうるというわけですが、、、まあこれはおいといて) 負の二項分布モデル(negative binomial model; NB model)では分散VARをVAR = MEAN + ϕ MEAN^2という数式($\phi > 0$)で表現するのが一般的です。ちなみにポアソンモデルというのはNBモデルの $\phi = 0$の場合(つまりVAR = MEAN; ある遺伝子の平均発現量がわかるとその遺伝子の分散が概ね平均と同じ、という意味)です。計算手順としては、同一サンプルの(biological) replicatesデータからϕがどの程度の値かを見積もります(推定します)。ここまでで「ある発現変動していない遺伝子(non-DEGs)の平均発現量がこれくらいのときの分散はこれくらい」という情報が得られます。これで発現変動していないものの分布(モデル)が決まります。その後、比較する二つのサンプルのカウント(発現量)データを一緒にして分散を計算し、non-DEGsの分布のどの程度の位置にいるか(p値)を計算し、形式的には帰無仮説(Null Hypothesis)を棄却するかどうかという手順を経てDEG or non-DEGの判定をします。

用いる正規化法の違いが(non-DEGの)モデル構築にどう影響するか？どう関係するのか？についてですが、これは極めてシンプルで、DEGかどうかの判定をする際に「比較する二つのサンプルのカウント(発現量)データを一緒にして分散を計算し...」を行う際に、サンプル間の正規化がうまくできていなければバラツキ(分散)を正しく計算できません。ただそれだけですが、用いる正規化法次第で結果が大きく異なりうる(Robinson and Oshlack, 2010)のです。そして発現変動解析結果はその後の解析(GO解析など)にほとんどそのまま反映されますので私を含む多くのグループが正規化法の開発や性能評価といったあたりの研究を行っています。

理想的な実験デザイン(二群間比較)

- サンプルA vs. Bの比較(Kidney vs. Liver; 腎臓 vs. 肝臓)
  - 生のリードカウントのデータ(整数値)

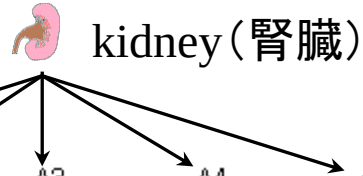
Gene ID	A1	A2	A3	A4	…	B1	B2	B3	B4	…
Gene1										
Gene2										
Gene3										
Gene4										
Gene5										
Gene6										
Gene7										
…										

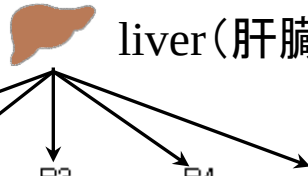
A1: ある生物の**腎臓**
A2: 同じ生物種の別個体の**腎臓**
A3: 同じ生物種のさらに別個体の**腎臓**
…
B1: ある生物の**肝臓**
B2: 同じ生物種の別個体の**肝臓**
…

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

分布の話

- 例題：Marioni et al., *Genome Res.*, **18**: 1509-1517, 2008のデータ（の一部）





	A1	A2	A3	A4	A5	B1	B2	B3	B4	B5
EnsemblGeneID	R1 L1 Kidney	R1 L3 Kidney	R1 L7 Kidney	R2 L2 Kidney	R2 L6 Kidney	R1 L2 Liver	R1 L4 Liver	R1 L6 Liver	R1 L8 Liver	R2 L3 Liver
ENSG00000146556	0	0	0	0	0	0	0	0	0	0
ENSG00000197194										
ENSG00000197490										
ENSG00000205292										
ENSG00000177693										
ENSG00000209338										
ENSG00000196573										
ENSG00000177799										
ENSG00000209341										
ENSG00000209342										
ENSG00000209343										
ENSG00000209344										
ENSG00000209346										
ENSG00000209349										
ENSG00000209350	4	7	3	6	7	35	32	31	29	34
ENSG00000209351	0	0	0	0	0	0	0	0	0	0
ENSG00000209352	0	0	1	1	0	2	0	0	0	0
ENSG00000212679	110	131	149	112	118	177	135	141	148	145
ENSG00000212678	12685	13204	12403	13031	13268	9246	9312	8746	8496	9070
ENSG00000185097	0	0	0	0	0	0	0	0	0	0

Technical replicatesのデータ

サンプル内の技術的なばらつき（例：レーン間の違い）の度合いを調べるためのデータであり、このようなデータで二群間比較し、**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無意味**

解析例：アリエナイ?!数（50%とか）が発現変動遺伝子として検出される

理由：Biological variation > Technical variation

分布の話

- 例題：Marioni et al., *Genome Res.*, **18**: 1509-1517, 2008のデータ（の一部）

kidney（腎臓）

EnsemblGeneID	A1	A2	A3	A4	A5
ENSG00000146556	0	0	0	0	0
ENSG00000197194	0	0	0	0	0
ENSG00000197490	0	0	0	0	0
ENSG00000205292	0	0	0	0	0
ENSG00000177693	0	0	0	0	0
ENSG00000209338	0	0	0	0	0
ENSG00000196573	0	0	0	0	0
ENSG00000177799	0	0	0	0	0
ENSG00000209341	0	0	0	0	0
ENSG00000209342	0	0	2	4	3
ENSG00000209343	0	0	0	0	0
ENSG00000209344	0	0	0	0	0
ENSG00000209346	0	0	0	0	0
ENSG00000209349	0	0	0	0	0
ENSG00000209350	4	7	3	6	7
ENSG00000209351	0	0	0	0	0
ENSG00000209352	0	0	1	1	0
ENSG00000212679	110	131	149	112	118
ENSG00000212678	12685	13204	12403	13031	13268
ENSG00000185097	0	0	0	0	0
…	…	…	…	…	…
総リード数	1804977	1855190	1742426	1927517	1963420

RPM
正規化

EnsemblGeneID	A1	A2	A3	A4	A5
ENSG00000209342	0.0	0.0	1.1	2.1	1.5
ENSG00000209350	2.2	3.8	1.7	3.1	3.6
ENSG00000209352	0.0	0.0	0.6	0.5	0.0
ENSG00000212679	60.9	70.6	85.5	58.1	60.1
ENSG00000212678	7027.8	7117.3	7118.2	6760.5	6757.6
ENSG00000197049	0.0	0.0	0.0	0.5	0.0
ENSG00000177757	1.1	0.0	1.1	0.5	1.5
ENSG00000177750	0.6	2.2	1.7	1.6	3.6
ENSG00000177741	0.6	0.5	0.0	3.1	0.0
ENSG00000198907	3.3	0.0	3.4	1.0	0.0
ENSG00000187634	27.1	23.2	23.5	21.8	23.9
ENSG00000188976	40.4	41.5	39.0	36.3	41.8
ENSG00000187961	8.3	8.1	7.5	6.2	7.6
ENSG00000187583	0.6	0.5	1.7	0.0	1.5
ENSG00000187642	2.2	2.7	6.9	4.7	4.6
ENSG00000188290	5.0	5.4	6.9	5.2	6.6
ENSG00000187608	6.6	5.9	4.0	8.3	6.6
ENSG00000188157	227.1	223.2	200.9	239.7	240.4
ENSG00000131591	5.5	4.9	4.0	6.2	8.1
ENSG00000215916	5.5	4.9	4.6	6.7	8.7
…	…	…	…	…	…
総リード数	1000000	1000000	1000000	1000000	1000000

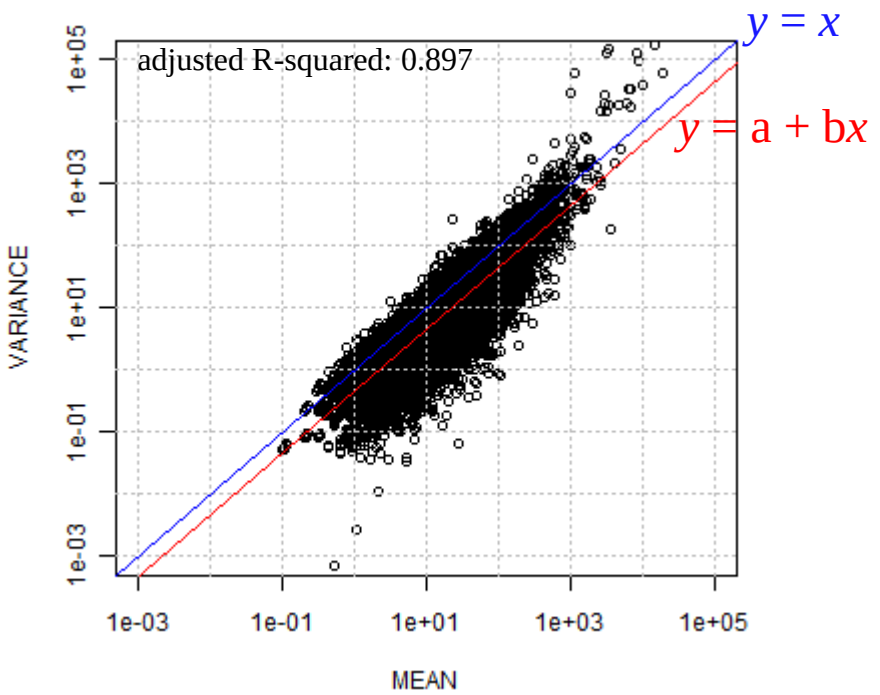
$$12{,}685 \times \frac{1{,}000{,}000}{1{,}804{,}977} = 7027.8$$

分布の話

- 例題：Marioni et al., *Genome Res.*, **18**: 1509-1517, 2008のデータ（の一部）


kidney（腎臓）

EnsemblGeneID	A1	A2	A3	A4	A5	MEAN	VARIANCE
ENSG00000209342	0.0	0.0	1.1	2.1	1.5	1.0	0.9
ENSG00000209350	2.2	3.8	1.7	3.1	3.6	2.9	0.8
ENSG00000209352	0.0	0.0	0.6	0.5	0.0	0.2	0.1
ENSG00000212679	60.9	70.6	85.5	58.1	60.1	67.1	129.8
ENSG00000212678	7027.8	7117.3	7118.2	6760.5	6757.6	6956.3	33770.4
ENSG00000197049	0.0	0.0	0.0	0.5	0.0	0.1	0.1
ENSG00000177757	1.1	0.0	1.1	0.5	1.5	0.9	0.4
ENSG00000177750	0.6	2.2	1.7	1.6	3.6	1.9	1.2
ENSG00000177741	0.6	0.5	0.0	3.1	0.0	0.8	1.7
ENSG00000198907	3.3	0.0	3.4	1.0	0.0	1.6	2.9
ENSG00000187634	27.1	23.2	23.5	21.8	23.9	23.9	3.9
ENSG00000188976	40.4	41.5	39.0	36.3	41.8	39.8	5.0
ENSG00000187961	8.3	8.1	7.5	6.2	7.6	7.5	0.7
ENSG00000187583	0.6	0.5	1.7	0.0	1.5	0.9	0.5
ENSG00000187642	2.2	2.7	6.9	4.7	4.6	4.2	3.4
ENSG00000188290	5.0	5.4	6.9	5.2	6.6	5.8	0.8
ENSG00000187608	6.6	5.9	4.0	8.3	6.6	6.3	2.4
ENSG00000188157	227.1	223.2	200.9	239.7	240.4	226.3	258.8
ENSG00000131591	5.5	4.9	4.0	6.2	8.1	5.8	2.5
ENSG00000215916	5.5	4.9	4.6	6.7	8.7	6.1	2.8
…	…	…	…	…	…	…	…
総リード数	1000000	1000000	1000000	1000000	1000000		

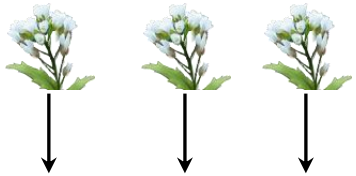


Technical replicatesのデータは：
・（遺伝子の）VARIANCEはそのMEANで説明可能である
・VARIANCE ≒ MEAN
・ポアソン分布に従う
・ポアソンモデルが適用可能

分布の話

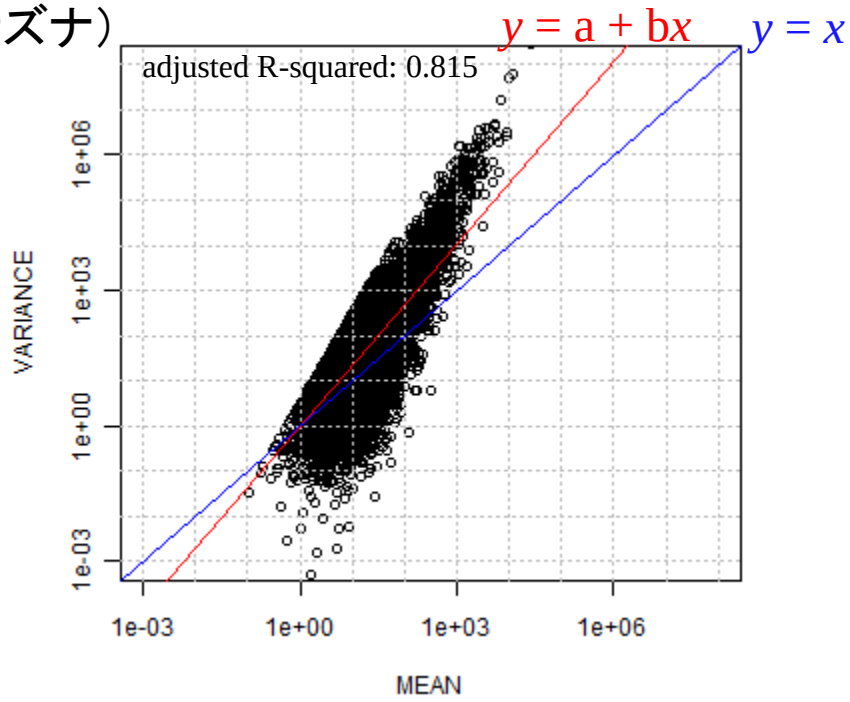
生物アイコン(http://biosciencedbc.jp/taxonomy_icon/taxonomy_icon.cgi)

- 例題:Cumbie et al., *PLoS ONE*, **6**: e25279, 2011のデータ(の一部)



Arabidopsis(シロイヌナズナ)

	mock1	mock2	mock3	MEAN	VARIANCE
AT1G01010	18.4	39.8	12.3	23.5	209.1
AT1G01020	22.6	23.3	9.8	18.6	57.5
AT1G01030	8.4	12.4	8.0	9.6	6.0
AT1G01040	37.9	22.2	19.6	26.6	97.1
AT1G01050	25.8	40.3	27.6	31.2	62.9
AT1G01060	0.0	7.8	0.6	2.8	18.6
AT1G01070	8.4	17.6	1.8	9.3	62.5
AT1G01080	89.4	98.8	117.2	101.8	200.2
AT1G01090	153.0	178.9	172.7	168.2	183.1
AT1G01100	59.4	64.6	75.5	66.5	67.1
AT1G01110	0.0	0.5	0.3	0.3	0.1
AT1G01120	119.9	97.7	82.8	100.1	347.3
AT1G01130	4.7	5.7	0.3	3.6	8.2
AT1G01140	95.2	62.0	43.6	66.9	683.3
…	…	…	…	…	…
総リード数	1000000	1000000	1000000		



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

倍率変化がだめな理由をデモ

- 例題：Marioni et al., *Genome Res.*, **18**: 1509-1517, 2008のデータ

	A1	A2	A3	A4	A5	B1	B2	B3	B4	B5
EnsemblGeneID	R1 L1 Kidney	R1 L3 Kidney	R1 L7 Kidney	R2 L2 Kidney	R2 L6 Kidney	R1 L2 Liver	R1 L4 Liver	R1 L6 Liver	R1 L8 Liver	R2 L3 Liver
ENSG00000146556	0	0	0	0	0	0	0	0	0	0
ENSG00000197194	0	0	0	0	0	0	0	0	0	0
ENSG00000197490	0	0	0	0	0	0	0	0	0	0
ENSG00000205292	0	0	0	0	0	0	0	0	0	0
ENSG00000177693	0	0	0	0	0	0	0	0	0	0
ENSG00000209338	0	0	0	0	0	0	0	0	0	0
ENSG00000196573	0	0	0	0	0	0	0	0	0	0
ENSG00000177799	0	0	0	0	0	0	0	0	0	0
ENSG00000209341	0	0	0	0	0	0	0	0	0	0
ENSG00000209342	0	0	2	4	3	0	0	0	1	0
ENSG00000209343	0	0	0	0	0	0	0	0	0	0
ENSG00000209344	0	0								
ENSG00000209346	0	0								
ENSG00000209349	0	0	0	0	0	0	0	0	0	0
ENSG00000209350	4	7	3	6	7	35	32	31	29	34
ENSG00000209351	0	0	0	0	0	0	0	0	0	0
ENSG00000209352	0	0	1	1	0	2	0	0	0	0
ENSG00000212679	110	131	149	112	118	177	135	141	148	145
ENSG00000212678	12685	13204	12403	13031	13268	9246	9312	8746	8496	9070
ENSG00000185097	0	0	0	0	0	0	0	0	0	0

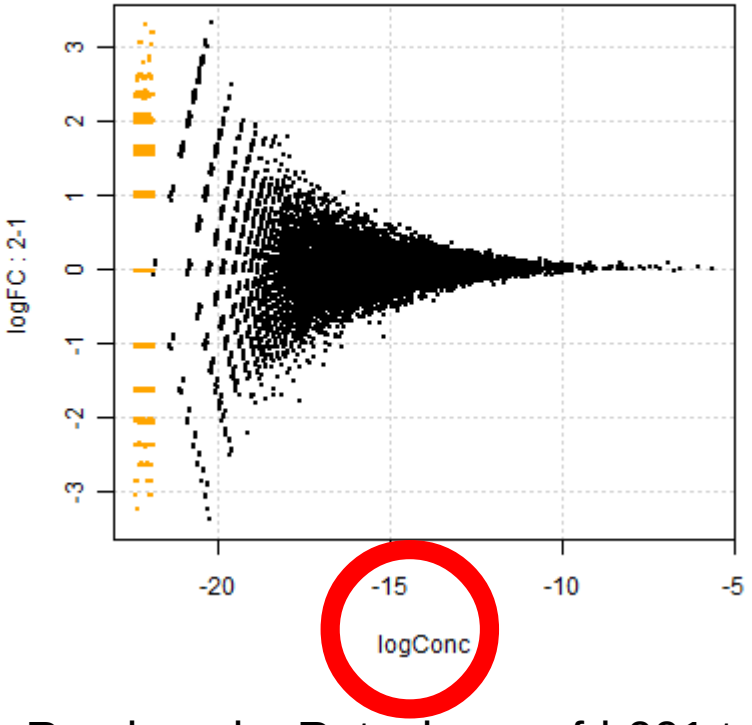
発現変動遺伝子がないデータで二群間比較をしてみる

A群　　B群

倍率変化がだめな理由をデモ

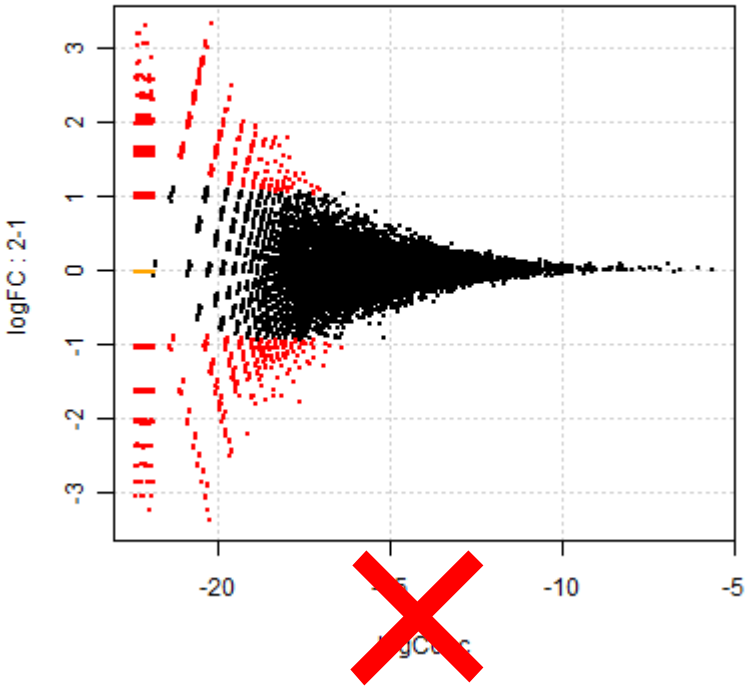
- 例題：Marioni et al., *Genome Res.*, **18**: 1509-1517, 2008のデータ（の一部）
  - (A1, A2) vs. (A3, A4)の二群間比較結果

*edgeR*でFDR < 0.01を満たすものは0個



Rcode_edgeR_tech_rep_fdr001.txt

（*edgeR*で）2倍以上発現変動しているものは3814個

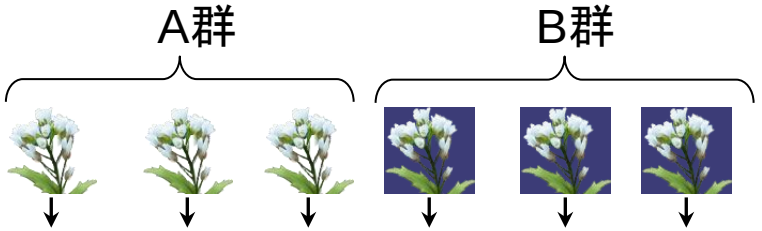


Rcode_edgeR_tech_rep_fc2.txt

低発現領域でlog比が大きくなる現象をうまくモデル化することが重要

Biological replicatesの3 vs. 3サンプル

- 例題：Cumbie et al., *PLoS ONE*, **6**: e25279, 2011のArabidopsis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

data_arab.txt

オリジナルは” AT4G32850”のものが重複して存在していたため19520行目のデータを予め除去している

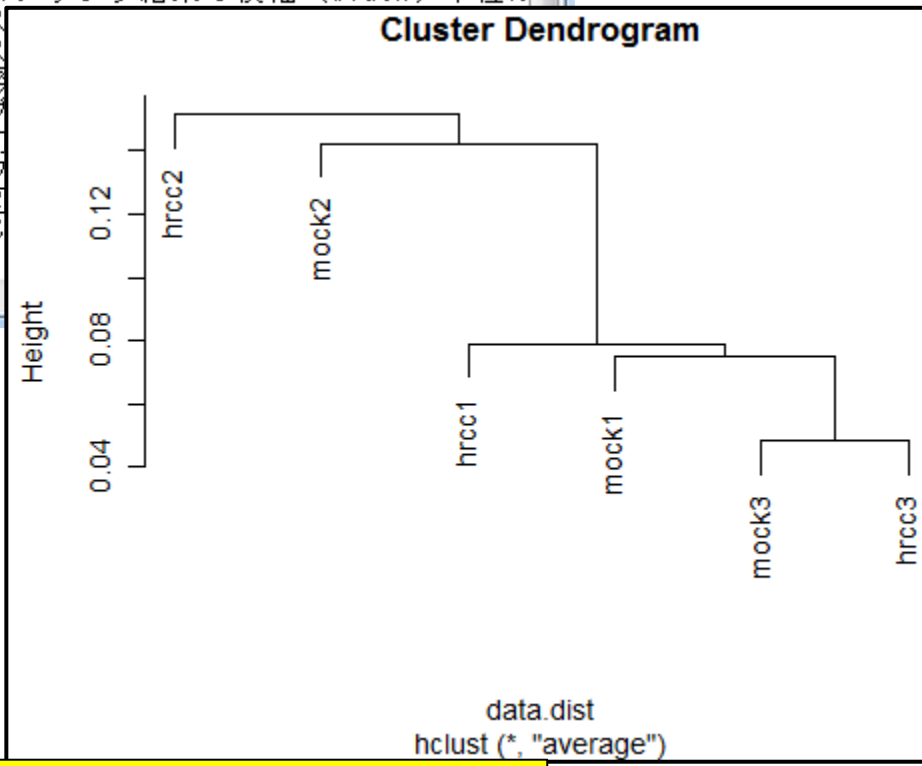
サンプル間クラスタリングも重要です

サンプル間クラスタリングも重要です

Rcode_clustering.txt - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

```
in_f <- "data_arab.txt"                                    #入力ファイル名(発現データファイル)を指
param2 <- "average"                                        #方法(method)を指定
out_f <- "result_cluster.png"                              #出力ファイル名(クラスタリング結果ファイ
param3 <- 500                                              #クラスタリング結果の横幅 (width; 単位は
param4 <- 400                                              #クラス
param5 <- 14                                               #クラス
data <- read.table(in_f, header=TRUE, row.names=1, sep="¥t", quote="")#
data.dist <- as.dist(1 - cor(data, method="spearman"))     #サンプ
out <- hclust(data.dist, method=param2)                    #階層的
png(out_f, pointsize=param5, width=param3, height=param4)  #出力フ
plot(out)                                                  #樹形図
dev.off()                                                  #おまじ
```



データ中に発現変動遺伝子がありそうかどうかは
クラスタリング結果を眺めるだけでかなりわかる

東大生以外の方も受講可能です（平成25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）

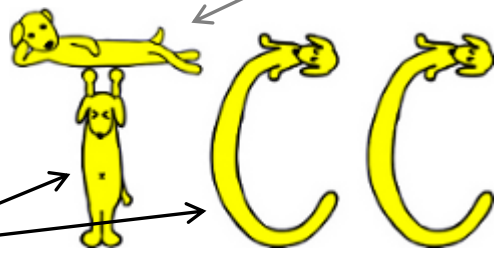
孫 建強　氏（東京大学・大学院農学生命科学研究科・修士課程1年）

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)

（妻の）門田 雅世さま作

挿絵やTCCのロゴなど



（有能な秘書の）三浦 文さま作