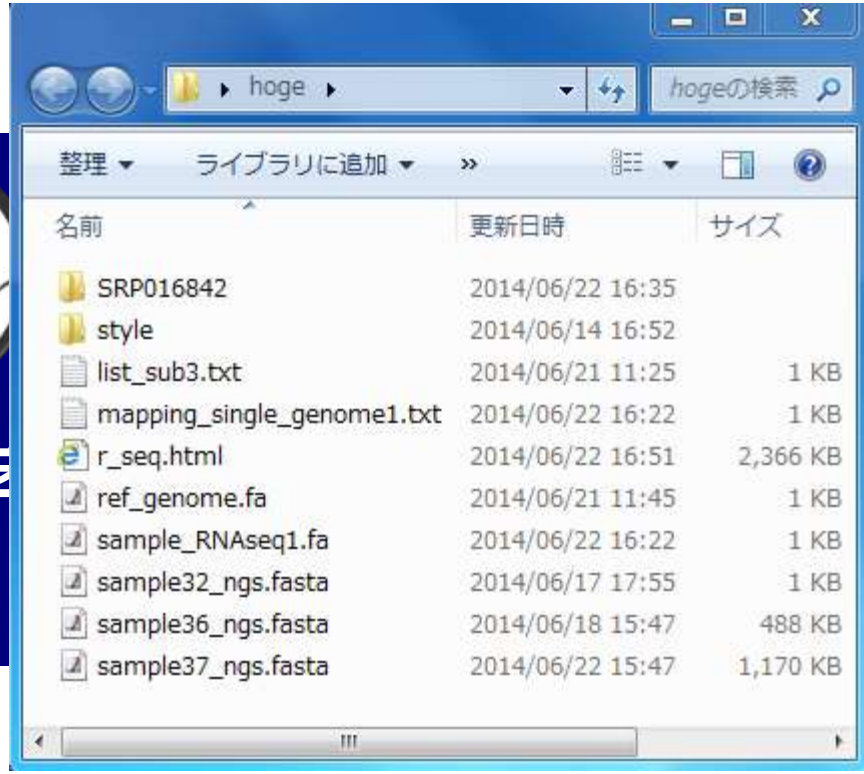
2016.02.01版

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



農学生命情報科学特論I 第3回

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
kadota@iu.a.u-tokyo.ac.jp

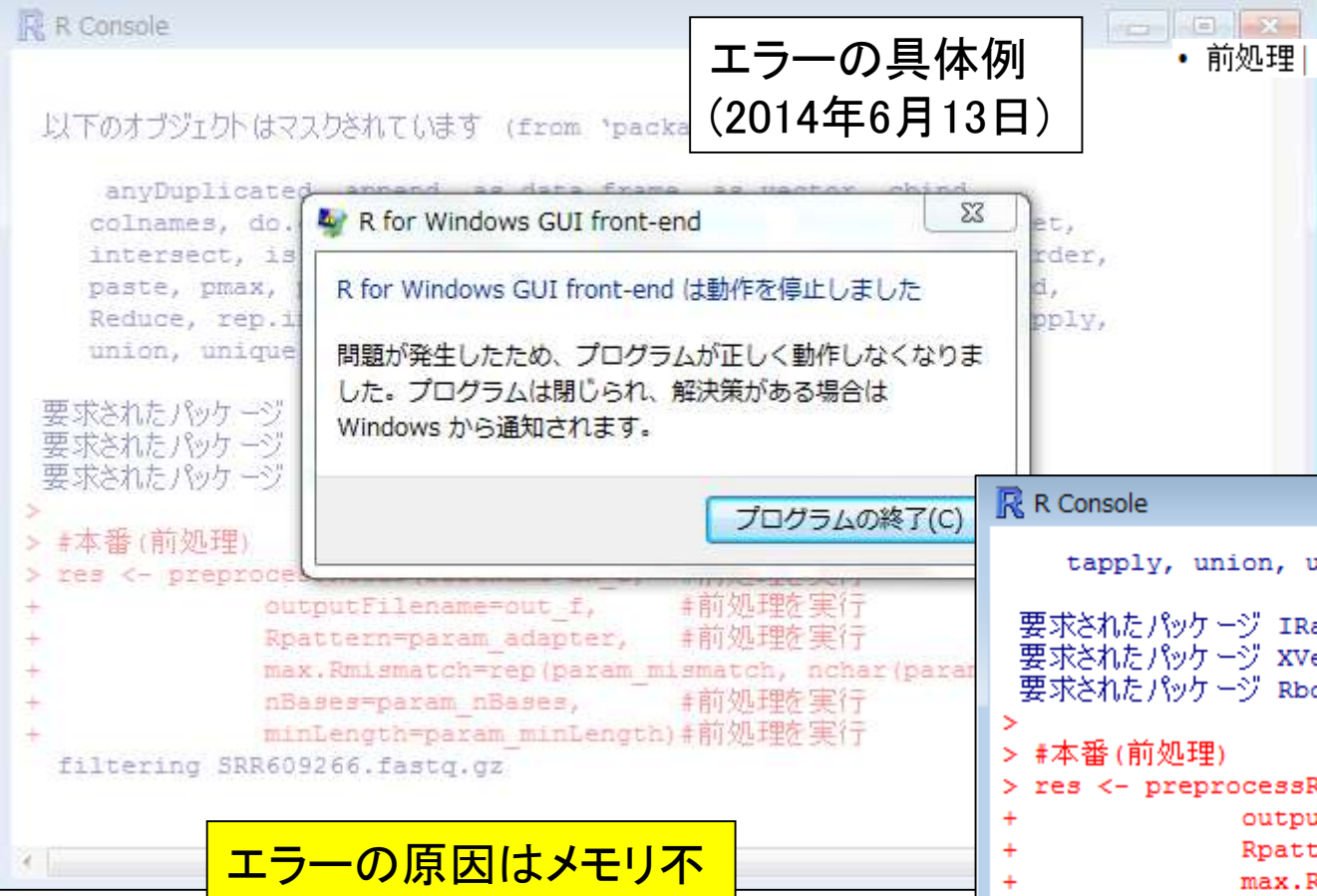
講義予定

- 第1回（2014年6月11日）
  - **西**：NSG概論。現状や展望など。講義のみ
- 第2回（2014年6月18日）
  - **門田**：データベース、データ取得、ファイル形式および変換、前処理
  - 教科書の1.3節周辺
- 第3回（2014年6月25日）
  - **門田**：アセンブル、マッピング、カウント情報取得
  - 教科書の2.3節周辺
- 第4回（2014年7月2日）
  - **門田**：クラスタリング、データ正規化、実験デザイン、分布（モデル）、発現変動解析
  - 教科書の3.3節周辺

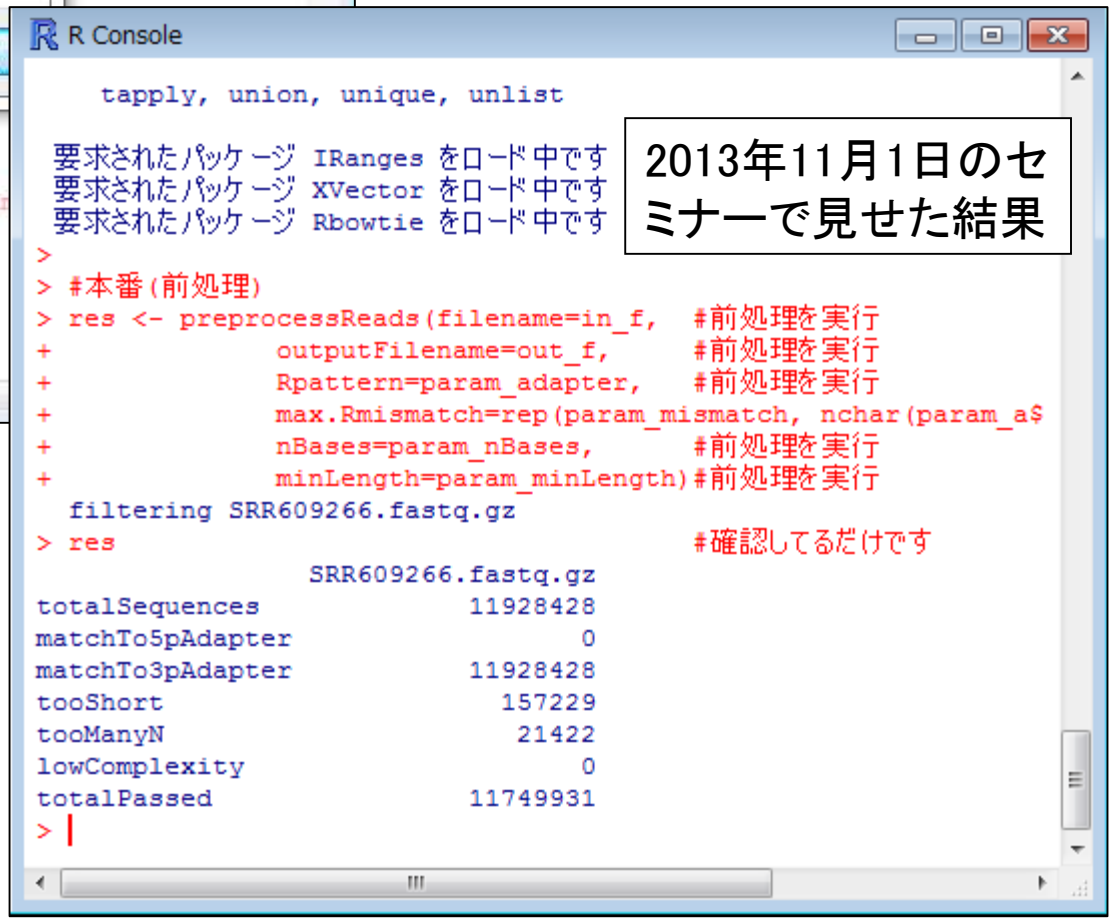
授業の目標・概要

次世代シーケンサ(NGS)の普及により、以前は主にゲノム解析系で必要とされていた配列解析のためのスキルがトランスクリプトーム解析においても要求される時代になっています。本科目では、様々な局面で応用可能な配列解析系のスキルアップを目指し、RNAシークエンス（RNA-Seq）に基づく（非モデル生物の）トランスクリプトーム解析を題材とした実習を含む講義を行います。

エラーの具体例
(2014年6月13日)



エラーの原因はメモリ不足だそうです by 孫堅強氏(2014年6月19日)



2013年11月1日のセミナーで見せた結果

課題遂行時に何人か遭遇したエラーの解説

4. FASTQ形式ファイル(SRR609266.fastq.gz)の場合:

small RNA-seqデータ(400Mb弱、11928428リード)です。圧縮ファイルもreadDNAStringSet関数で通常手順で読み込めます。原著論文(Nie et al., BMC Genomics, 2013)中の記述から GSE41841を頼りに、SRP016842にたどりつき、イントロ | NGS | 配列取得 | FASTQ or SRALite | SRAdb(Zhu 2013)の7を実行して得られたものが入力ファイルです。
原著論文中では、アダプター配列やクオリティの低いリードを除去したのち、ゲノムにマッピングしたと書いてあります。アダプター配列情報はどこにも書かれていませんでしたが、Table S2中のアダプター配列除去後の最も短いリードが18 nt (例:"GCAGTCGTGGCCGAGCGG")であり、「この18 nt」と「この配列を含む生リード配列の差分」がアダプター配列ということになります。詳細な情報は書かれていませんでしたが、おそらくアダプター配列は"TGGAATTCTCGGGTGCCAAGGAACTCCAGTC..."という感じだろうと推測できます。
アダプター配列既知("TGGAATTCTCGGGTGCCAAGGAACTCCAGTC")で、許容するミスマッチ数が2、ACGTのみからなる配列(param_nBases <- 0)、配列長の範囲指定(20:30)の組み合わせです。

```
in_f <- "SRR609266.fastq.gz"              #入力ファイル名を指定してin_fに格納(RNA-seqファイル)
out_f <- "hoge4.fasta.gz"                 #出力ファイル名を指定してout_fに格納
param_adapter <- "TGGAATTCTCGGGTGCCAAGGAACTCCAGTC"#アダプター配列を指定
param_mismatch <- 2                       #許容するミスマッチ数を指定
param_nBases <- 0
param_range <- 20:30

#必要なパッケージをロード
library(ShortRead)

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

何人かの方が、作業ディレクトリの変更も正しく行い、SRR609266.fastq.gzファイルもhogeフォルダ中に存在するにも関わらず、入力ファイル読み込み時にエラーに遭遇しました。この理由は2つ考えられます。1つめは、USBメモリにコピーする際に正しくコピーできていなかった可能性、そして2つめはUSBメモリ中のSRR609266.fastq.gzファイル段階では正しいものであったが、各自のPCにコピーする際に正しくコピーできなかった可能性です。講義中に述べたMD5チェックサム(MD5 check sum)でファイルの同一性を確認するのは重要ですね。

Contents（第3回）

- アセンブル(Assembly)
  - 2つのアプローチ(two approaches)
    - Comparative approach (reference-based assembly; resequencing)：同一生物種または近縁種のゲノム配列を利用
    - *de novo* approach：過去に配列決定されたものの中に近縁種がない場合
  - アルゴリズム（計算手順）
    - k-mer解析
  - ゲノム用、トランスクリプトーム用、雑感
- マッピング（QuasRパッケージを利用）
  - シミュレーションデータを用いたマッピングの基礎
  - リアルデータのマッピング（カイコsmall RNA-seqデータ）
  - 課題
- カウント情報取得

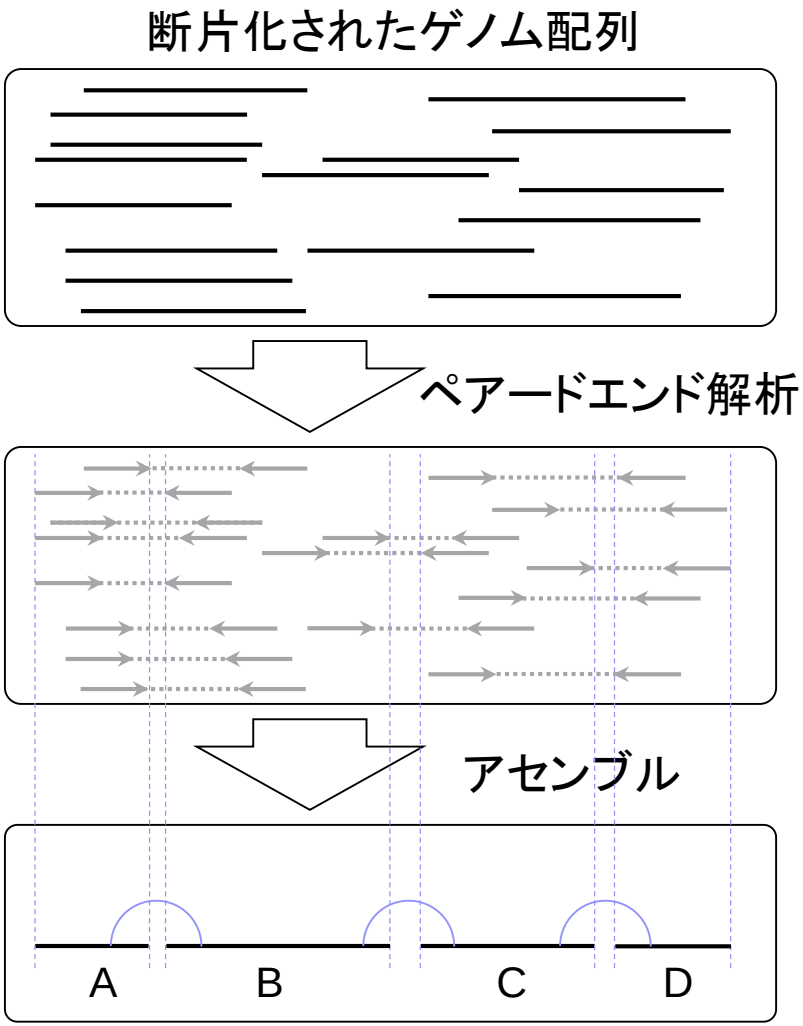
ゲノムアセンブル

- Comparative approach
  - 同一生物種または近縁種のゲノム配列を利用するreference-based assembly。Resequencingともいう。
  - ヒトゲノムresequencingやSNP解析系はこちら
    - 一個人のヒトゲノム(Wheeler et al., *Nature*, **452**: 872-876, 2008)
    - 日本人ゲノム(Fujimoto et al., *Nat. Genet.*, **42**: 931-936, 2010)
    - ENCODE project (ENCODE Project Consortium et al., *Nature*, **489**: 57-74, 2012)
- *de novo* approach
  - 過去に配列決定された生物種以外が主な対象。
  - パンダ(Li et al., *Nature*, **463**: 311-317, 2008)
  - サンゴ(Shinzato et al., *Nature*, **476**: 320-323, 2011)

(NGS由来の比較的短い)配列決定されたリードのみから、目的生物種のゲノム配列を決めること(組み立てること)

Tips

- リード(read)
  - Sequencerで読んだ塩基配列のこと
- コンティグ(contig)
  - 異なる複数のリードがACGTの切れ目なく連結されたもの
  - 右図ではA-Dの四つのコンティグ
- Scaffold (supercontig)
  - コンティグ間の位置関係を表したもの
  - 「A-D-B-C」ではなく「A-B-C-D」という関係
- N50
  - 得られた複数のコンティグを最も長いコンティグから順番に連結していったときにcombined total lengthの50%になったときのコンティグの長さ



参考

赤字の部分、間違って「小さめ」と書いてしまっていたことに2016年2月1日に気づいたので修正しました。この種のミスはいけませんね。失礼しましたm(_ _)m

Tips

- Coverage（カバレッジ）
  - ゲノム解読したいときなどに、解読するために必要とされる指標となる数値。ゲノムサイズ(X)に対する、sequencerで読んだ塩基配列長の和のこと。一般に、この数値が高いほどよい。Sequence depthという表現と実質的に同じような指標。
- アセンブル時に用いる*k*の値はいくつがいいの？
  - 複数の*k*の値を試すようです。2013年ごろから自動的に決めてくれるものが増えたようです。
- アセンブル結果の評価基準は？
  - よくわかりません。平均コンティグ長やN50が論文の表でよく記述されます。このあたりの数値を大きくするだけなら、*k*の値を**大きめ**にすればいいです。
  - ただ、ゲノムアセンブリの場合には実質的には長ければ長いほどよいという感じみたいです。
- アセンブルプログラムを実行して得られる出力ファイルはどんな感じ？
  - （基本的に）multi-FASTA形式のファイルです。

```
>contig1
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
…
>contig2
ACGATGCAGCCTTAACGA…
>contig3
…
```

ゲノムアセンブルの手順

1. 前処理(pre-processing filtering)
   - クオリティの低いリードやコンタミを除去するステップ。塩基置換(substitution)やインデル(indels; insertion/deletion)を含むリードの除去や補正(error correction)。
   - 4つのアプローチ：**k-mer**, suffix tree/array, multiple sequence alignment, hybrid
2. グラフ構築(graph construction)
   - 前処理後のリードを用いて、リード間のオーバーラップ(overlap)を頼りにつなげていくステップ。シークエンスエラー(sequencing error)と多型(polymorphism)の違いを見るべく、グラフ構築時にエラー補正を行うものもある。
   - 4つのアプローチ：OLC, **de Bruijn graph (k-mer)**, greedy, hybrid
3. グラフ簡易化(graph simplification)
   - グラフ構築後に、複雑化したグラフをシンプルにしていくステップ。連続したノード(nodes; 頂点) やバブルのマージ作業に相当。
4. 後処理(post-processing)
   - コンティグ(contigs)やスカッフォールド(scaffolds)を得るステップ。ミスアセンブリの同定も含む。

大きく分けて4つの手順からなる

- 前処理 | トリミング | 指定した末端塩基数だけ除去 (last modified 2013/06/15)
- アセンブル | について (last modified 2014/06/16) NEW
- アセンブル | ゲノム用 (last modified 2014/06/15) NEW
- アセンブル | トランスクリプトーム(転写物)用 (last modified 2014/06/10) NEW

- アセンブル | について

アセンブル | について NEW

アセンブルという言葉がなかなか理解しずら
本一本の配列だけからでは到底到達できな
とで差し支えないと思います。
出力ファイル形式はFASTAがデファクトスタ
アセンブルによって、まず一本一本のリード
コンティグ間のギャップサイズ("N"の数で表
のコンティグの並びがどうなっているかを表
一般にFASTA形式のアセンブルされた配列
ティグ長(maximum length)、平均コンティグ長
N50などです。
ちなみにこのN50というのは、「最も長いコン
コンティグの長さ」です。「combined total leng
という言い方もできます。length-weighted m
アセンブリを行う際に問題となるのは「リピー
ペアードエンド(paired-end)の一部がリピート
リピート配列の素性がかなりわかります。
後者については、いわゆる感度・特異度の議
い)ためには多少ミスアセンブル(特異度が

デノボゲノムアセンブリ(de novo genome ass

ゲノムアセンブリの手順は、以下に示すよ
Biol., 2013):

1. 前処理(pre-processing filtering; erro
2. グラフ構築(graph construction proce
3. グラフ簡易化(graph simplification p
4. 後処理(post-processing filtering)

1. 前処理(pre-processing filtering; error correction):

ここで行う処理は、塩基置換(substitution; mismatch)、インデル(indels; insertion/deletion)、曖昧な塩基(N)を含むリードの除去や補正です。基本的な戦略は「エラーの頻度は低い」ですが、原理的に高頻度で出現するリピート配列の悪影響を受けるようです。
シークエンスエラー(sequencing error)と多型(polymorphism)の違いはグラフ構築後でないとわからないので、2.のグラフ構築ステップ時に行う場合もあるようです。また、アセンブルプログラムの中に完全に組み込まれていたりなど切り分けは若干ややこしいですがざっとリストアップしておきます。エラー補正プログラムとほぼ同義です。ちなみに、初期のアセンブラはこのエラー補正ステップがないようです(El-Metwally et al., PLoS Comput Biol., 2013)。この理由は、ABI3730のような800bp程度まで読めるロングリードのアセンブリの場合は配列一致部分も長い(long overlap)ので、一致部分に多少のエラーを含もうが影響は限定的だったからです。
エラー補正は大きく4つのアプローチに分けられるそうです:K-spectrum (k-mer), Suffix Tree/Array (STA), Multiple Sequence Alignment (MSA), Hybrid。

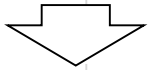
- FreClu: Qu et al., Genome Res., 2009
- SHREC(STA correction): Schröder et al., Bioinformatics, 2009
- Hybrid SHREC(Hybrid correction): Salmela L., Bioinformatics, 2010
- Reptile(k-mer correction): Yang et al., Bioinformatics, 2010
- EDAR: Zhao et al., J Comput Biol., 2010
- Quake(k-mer correction): Kelley et al., Genome Biol., 2010
- HiTEC(STA correction): Ilie et al., Bioinformatics, 2011
- REDEEM: Yang et al., BMC Bioinformatics, 2011
- DecGPU: Liu et al., BMC Bioinformatics, 2011
- Coral(MSA correction): Salmela et al., Bioinformatics, 2011
- ECHO(MSA correction): Kao et al., Genome Res., 2011
- Hammer(k-mer correction): Medvedev et al., Bioinformatics, 2011
- PBcR(Hybrid correction): Koren et al., Nat Biotechnol., 2012
- KEC and ET: Skums et al., BMC Bioinformatics, 2012
- BayesHammer: Nikolenko et al., BMC Genomics, 2013
- SEECER: Le et al., Nucleic Acids Res., 2013
- RACER: Ilie et al., Bioinformatics, 2013
- CorQ: Iyer et al., PLoS One, 2013
- Sleep's method: Sleep et al., BMC Bioinformatics, 2013
- BLESS: Heo et al., Bioinformatics, 2014
- HECTOR: Wirawan et al., BMC Bioinformatics, 2014
- Blue: Greenfield et al., Bioinformatics, 2014

様々な戦略があります。
k-merを利用する方法
の基本を紹介します

1. 前処理：k-mer

- k-mer頻度解析
  - NGSデータを入力として、リード長より短いk連続塩基からなる部分文字列を発生させるのが最初のステップ。発生させたk-merの出現頻度情報をもとに、カバレッジ、ゲノムサイズ推定、コンタミリードの除去などを行う
  - 例1：20塩基長のNGSリードをk=19で分割すると2個のk-merを発生可能

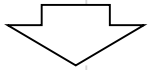
CACCAGGACATGAAGACGCG



CACCAGGACATGAAGACGC
ACCAGGACATGAAGACGCG

  - 例2：20塩基長のNGSリードをk=17で分割すると4個のk-merを発生可能

CACCAGGACATGAAGACGCG



CACCAGGACATGAAGAC
ACCAGGACATGAAGACG
CCAGGACATGAAGACGC
CAGGACATGAAGACGCG

L塩基長のNGSリードをk-merに分割すると(L − k + 1)個のk-merを発生可能

項目名は変更する可能性あり

- 解析 | 一般 | 上流配列解析 | Relative Appearance Ra (Yamamoto 2011)(last m
- 解析 | 基礎 | k-mer | ゲノムサイズ推定(基礎) | qrqc modified 2014/06/17) N
- 解析 | 基礎 | 平均-分散プロット | Technical replicates(st modified 2014/02/18)
- 解析 | 基礎 | 平均-分散プロット | Biological replicates(last modified 2014/02/21)
- 解析 |
- 解析 |
- 解析 |
- 解析 |

解析 | 基礎 | k-mer | ゲノムサイズ推定(基礎) | qrqc NEW

qrqcパッケージを用いてNGSデータのk-mer解析を行うやり方を示します。ゲノムサイズ推定を行うための基本的な考え方を示します。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

1. サンプルデータ32のFASTA形式ファイル(sample32_ngs.fasta)の場合:

4X coverageであることが分かっているデータです。理由は、50塩基長のリファレンス配列から20塩基長のリードを10個ランダム抽出したものだからです。k-merのkの値は、リード配列長以下に設定します。この場合は20塩基以下ですので19を指定しています。

```
in_f <- "sample32_ngs.fasta"              #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.png"                      #出力ファイル名を指定してout_fに格納
param_k <- 19                             #k-merのkの値を指定
param_fig <- c(400, 380)                  #ヒストグラム描画時の横幅と縦幅を指定(単

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

#入力ファイルの読み込み
hoge <- readSeqFile(in_f, type="fasta", kmer=T, k=param_k)#in_fで指定したフ
hoge                                      #確認してるだけです

#本番(k-mer出現頻度解析)
kmer <- table(hoge@kmer$kmer)             #k-merごとの出現頻度を抽出
kmer                                      #確認してるだけです
length(kmer)                              #k-merの種類数を表示(sequence errorが

#ファイルに保存(ヒストグラム)
png(out_f, pointsize=13, width=param_fig[1], height=param_fig[2])#出力ファイ
hist(kmer, ylab="Frequency(number of k-mers)",#ヒストグラムを描画
     xlab=paste("Number of occurences at k=", param_k, sep=""))#ヒストグラム
dev.off()                                 #おまじない
```

```
>kkk_24_43
CACCAGGACATGAAGACGCG
>kkk_28_47
AGGACATGAAGACGCGGACG
>kkk_12_31
TTGAACTCACTACACCAGGA
>kkk_4_23
GTTGTCTTTTGAACTCACTA
>kkk_5_24
TTGTCTTTTGAACTCACTAC
>kkk_7_26
GTCTTTTGAACTCACTACAC
>kkk_12_31
TTGAACTCACTACACCAGGA
>kkk_23_42
ACACCAGGACATGAAGACGC
>kkk_1_20
GAAGTTGTCTTTTGAACTCA
>kkk_3_22
AGTTGTCTTTTGAACTCACT
```

k=19としてk-mer頻度分布を作成。
入力ファイルについて説明します。

- 過去のお知らせ (last modified 2014/06/13) NEW
- Rのインストールと起動 (last modified 2014/0
- サンプルデータ (last modified 2014/06/17) N
- 書籍 | トランスクリプトームについて (last mod
- 書籍 | トランスクリプトーム解析 | 2.3.1 RNA-seqデータ(FASTQファイル)

>kkk
GAAGTTGTCTTTTGAACTCACTACACCAGGACATGAAGACGCGGACGGCA

- サンプルデータ

サンプルデータ NEW

1. Illumina/36bp/single-end/human (SRA000299) data (Marioni et al., Genome Res., 2008)
「Kidney 7 samples vs Liver 7 samples」のRNA-seqの遺伝子発現行列データ

32. k-mer解析用のランダム配列から生成したFASTA形式ファイル(sample32_ref.fastaとsample32_ngs.fasta)です。50塩基の長さのリファレンス配列を生成したのち、20塩基長の部分配列を10リード分だけランダム抽出したものです。塩基の存在比はAが22%, Cが28%, Gが28%, Tが22%にしています。リファレンス配列(仮想ゲノム配列)がsample32_ref.fastaで、10リードからなる仮想NGSデータがsample32_ngs.fastaです。リード長20塩基で10リードなのでトータル200塩基となり、50塩基からなる元のゲノム配列の4倍シーケンスしていることになります(4X coverageに相当)。イントロ | NGS | 配列取得 | シミュレーションデータ | ランダムな塩基配列の生成からと基本的に同じです。

```
out_f1 <- "sample32_ref.fasta"          #出力ファイル名を指定してout_f1に格納
out_f2 <- "sample32_ngs.fasta"          #出力ファイル名を指定してout_f2に格納
param_len_ref <- 50                     #リファレンス配列の長さを指定
narabi <- c("A","C","G","T")            #以下の数値指定時にACGTの並びを間違えないように
param_composition <- c(22, 28, 28, 22) #(A,C,G,Tの並びで)各塩基の存在比率を指定
param_len_ngs <- 20                     #リード長を指定
param_num_ngs <- 10                     #リード数を指定
param_desc <- "kkk"                     #FASTA形式ファイルのdescription行に記述する内

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

#本番(リファレンス配列生成)
set.seed(1010)                          #おまじない(同じ乱数になるようにするため)
ACGTset <- rep(narabi, param_composition)#narabi中の塩基がparam_compositionで指定し
reference <- paste(sample(ACGTset, param_len_ref, replace=T), collapse="")#ACGTset
reference <- DNAStringSet(reference)    #DNA塩基配列だと認識させるDNAStringSet関数を通
names(reference) <- param_desc          #description
reference                               #確認してるだ

#本番(シミュレーションデータ生成)
s_posi <- sample(1:(param_len_ref-param_len_ngs), p
```

```
>kkk_24_43
CACCAGGACATGAAGACGCG
>kkk_28_47
AGGACATGAAGACGCGGACG
>kkk_12_31
TTGAACTCACTACACCAGGA
>kkk_4_23
GTTGTCTTTTGAACTCACTA
>kkk_5_24
TTGTCTTTTGAACTCACTAC
>kkk_7_26
GTCTTTTGAACTCACTACAC
>kkk_12_31
TTGAACTCACTACACCAGGA
>kkk_23_42
ACACCAGGACATGAAGACGC
>kkk_1_20
GAAGTTGTCTTTTGAACTCA
>kkk_3_22
AGTTGTCTTTTGAACTCACT
```

sample32_ngs.fastaは、50塩基からなるランダム塩基配列をリファレンスとしています。20塩基からなる10個の部分配列をNGSリードとしています。

- 過去のお知らせ (last modified 2014/06/13) NEW
- Rのインストールと起動 (last modified 2014/0
- サンプルデータ (last modified 2014/06/17) N
- 書籍 | トランスクリプトームについて (last mod
- 書籍 | トランスクリプトーム解析 | 2.3.1 RNA-seqデータ(FASTQファイル)

>kkk
GAAGTTGTCTTTTGAACTCACTACACCAGGACATGAAGACGCGGACGGCA

- サンプルデータ

サンプルデータ NEW

1. Illumina/36bp/single-end/human (SRA000299) data (Marioni et al., Genome Res., 2008)
「Kidney 7 samples vs Liver 7 samples」のRNA-seqの遺伝子発現行列データ

32. k-mer解析用のランダム配列から生成したFASTA形式ファイル(sample32_ref.fastaとsample32_ngs.fasta)です。50塩基の長さのリファレンス配列を生成したのち、20塩基長の部分配列を10リード分だけランダム抽出したものです。塩基の存在比はAが22%, Cが28%, Gが28%, Tが22%にしています。リファレンス配列(仮想ゲノム配列)がsample32_ref.fastaで、10リードからなる仮想NGSデータがsample32_ngs.fastaです。リード長20塩基で10リードなのでトータル200塩基となり、50塩基からなる元のゲノム配列の4倍シーケンスしていることになります(4X coverageに相当)。イントロ | NGS | 配列取得 | シミュレーションデータ | ランダムな塩基配列の生成からと基本的に同じです。

```
out_f1 <- "sample32_ref.fasta"          #出力ファイル名を指定してout_f1に格納
out_f2 <- "sample32_ngs.fasta"          #出力ファイル名を指定してout_f2に格納
param_len_ref <- 50                     #リファレンス配列の長さを指定
narabi <- c("A","C","G","T")            #以下の数値指定時にACGTの並びを間違えないように
param_composition <- c(22, 28, 28, 22) #(A,C,G,Tの並びで)各塩基の存在比率を指定
param_len_ngs <- 20                     #リード長を指定
param_num_ngs <- 10                     #リード数を指定
param_desc <- "kkk"                     #FASTA形式ファイルのdescription行に記述する内

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

#本番(リファレンス配列生成)
set.seed(1010)                          #おまじない(同じ乱数になるようにするため)
ACGTset <- rep(narabi, param_composition)#narabi中の塩基がparam_compositionで指定し
reference <- paste(sample(ACGTset, param_len_ref, replace=T), collapse="")#ACGTset
reference <- DNAStringSet(reference)    #DNA塩基配列だと認識させるDNAStringSet関数を通
names(reference) <- param_desc          #description行に相当する記述を追加している
reference                               #確認してるだけです

#本番(シミュレーションデータ生成)
s_posi <- sample(1:(param_len_ref-param_len_ngs), param_num_ngs, repl
```

```
>kkk_24_43
CACCAGGACATGAAGACGCG
>kkk_28_47
AGGACATGAAGACGCGGACG
>kkk_12_31
TTGAACTCACTACACCAGGA
>kkk_4_23
GTTGTCTTTTGAACTCACTA
>kkk_5_24
TTGTCTTTTGAACTCACTAC
>kkk_7_26
GTCTTTTGAACTCACTACAC
>kkk_12_31
TTGAACTCACTACACCAGGA
>kkk_23_42
ACACCAGGACATGAAGACGC
>kkk_1_20
GAAGTTGTCTTTTGAACTCA
>kkk_3_22
AGTTGTCTTTTGAACTCACT
```

各リード中のdescription部分の記述が部分配列の位置情報に相当

解析 | 基礎 | k-mer | ゲノムサイズ推定(基礎) | qrqc NEW

qrqcパッケージを用いてNGSデータのk-mer解析を行うやり方を示します。ゲノムサイズ推定を行うための基本的な考え方を示します。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

1. サンプルデータ32のFASTA形式ファイル(sample32_ngs.fasta)の場合:

4X coverageであることが分かっているデータです。理由は、50塩基長のリードを10個ランダム抽出したものだからです。この場合は20塩基以下ですので19を指定し

```
in_f <- "sample32_ngs.fasta"
out_f <- "hoge1.png"
param_k <- 19
param_fig <- c(400, 380)

#必要なパッケージをロード
library(qrqc)

#入力ファイルの読み込み
hoge <- readSeqFile(in_f, type="fast
hoge

#本番(k-mer出現頻度解析)
kmer <- table(hoge@kmer$kmer)
kmer
length(kmer)

#ファイルに保存(ヒストグラム)
png(out_f, pointsize=13, width=param
hist(kmer, ylab="Frequency(number of
     xlab=paste("Number of occurence
dev.off()
```

R Console

```
> #入力ファイルの読み込み
> hoge <- readSeqFile(in_f, type="fasta", km$
> hoge                                        #確$
Quality Information for: sample32_ngs.fasta
 10 sequences, 9 unique with 0.10 being samp$
 min sequence length: 20
 max sequence length: 20
>
> #本番(k-mer出現頻度解析)
> kmer <- table(hoge@kmer$kmer)               #k-$
> kmer                                        #確$

AAGTTGTCTTTTGAACTCA ACACCAGGACATGAAGACG
                  1                   1
ACCAGGACATGAAGACGCG AGGACATGAAGACGCGGAC
                  1                   1
AGTTGTCTTTTGAACTCAC CACCAGGACATGAAGACGC
                  1                   2
GAAGTTGTCTTTTGAACTC GGACATGAAGACGCGGACG
                  1                   1
GTCTTTTGAACTCACTACA GTTGTCTTTTGAACTCACT
                  1                   2
TCTTTTGAACTCACTACAC TGAACTCACTACACCAGGA
                  1                   1
TGTCTTTTGAACTCACTAC TTGAACTCACTACACCAGG
                  1                   1
TTGTCTTTTGAACTCACTA
                  2
>
```

```
>kkk_24_43
CACCAGGACATGAAGACGCG
>kkk_28_47
AGGACATGAAGACGCGGACG
>kkk_12_31
TTGAACTCACTACACCAGGA
>kkk_4_23
GTTGTCTTTTGAACTCACTA
>kkk_5_24
TTGTCTTTTGAACTCACTAC
>kkk_7_26
GTCTTTTGAACTCACTACAC
>kkk_12_31
TTGAACTCACTACACCAGGA
>kkk_23_42
ACACCAGGACATGAAGACGC
>kkk_1_20
GAAGTTGTCTTTTGAACTCA
>kkk_3_22
AGTTGTCTTTTGAACTCACT
```

k=19としてk-merの種類ごとに頻度情報を取得したのがkmerオブジェクト

解析 | 基礎 | k-mer | ゲノムサイズ推定(基礎) | qrqc NEW

qrqcパッケージを用いてNGSデー
の基本的な考え方を示します。
「ファイル」-「ディレクトリの変更」で

1. サンプルデータ32のFASTA形

4X coverageであることが分かって
長のリードを10個ランダム抽出し
す。この場合は20塩基以下です(

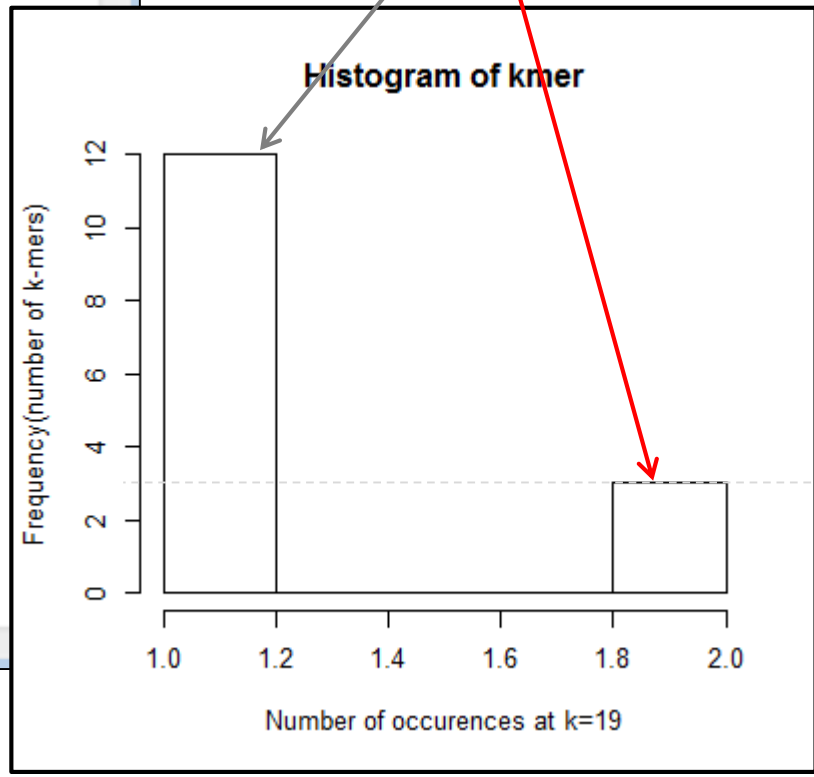
```
in_f <- "sample32_ngs.fa
out_f <- "hoge1.png"
param_k <- 19
param_fig <- c(400, 380)

#必要なパッケージをロード
library(qrqc)

#入力ファイルの読み込み
hoge <- readSeqFile(in_f
hoge

#本番(k-mer出現頻度解析)
kmer <- table(hoge@kmer$
kmer
length(kmer)

#ファイルに保存(ヒストグラ
png(out_f, pointsize=13,
hist(kmer, ylab="Frequen
     xlab=paste("Number
dev.off()
```

R Console

```
> #入力ファイルの読み込み
> hoge <- readSeqFile(in_f, type="fasta", km$
> hoge                                        #確$
Quality Information for: sample32_ngs.fasta
 10 sequences, 9 unique with 0.10 being samp$
 min sequence length: 20
 max sequence length: 20
>
> #本番(k-mer出現頻度解析)
> kmer <- table(hoge@kmer$kmer)               #k-$
> kmer                                        #確$

AAGTTGTCTTTTGAACTCA ACACCAGGACATGAAGACG
                  1                   1
ACCAGGACATGAAGACGCG AGGACATGAAGACGCGGAC
                  1                   1
AGTTGTCTTTTGAACTCAC CACCAGGACATGAAGACGC
                  1                   2
GAAGTTGTCTTTTGAACTC GGACATGAAGACGCGGACG
                  1                   1
GTCTTTTGAACTCACTACA GTTGTCTTTTGAACTCACT
                  1                   2
TCTTTTGAACTCACTACAC TGAACTCACTACACCAGGA
                  1                   1
TGTCTTTTGAACTCACTAC TTGAACTCACTACACCAGG
                  1                   1
TTGTCTTTTGAACTCACTA
                  2
>
```

• 解析 | 基礎 | k-mer | ゲノムサイズ推定(基礎) | qrqc

1回出現したk-merが12個、
2回出現したk-merが3個、
という解釈をする



3. サンプルデータ32のFASTA形式ファイル(sample32_ngs.fasta)の場合:

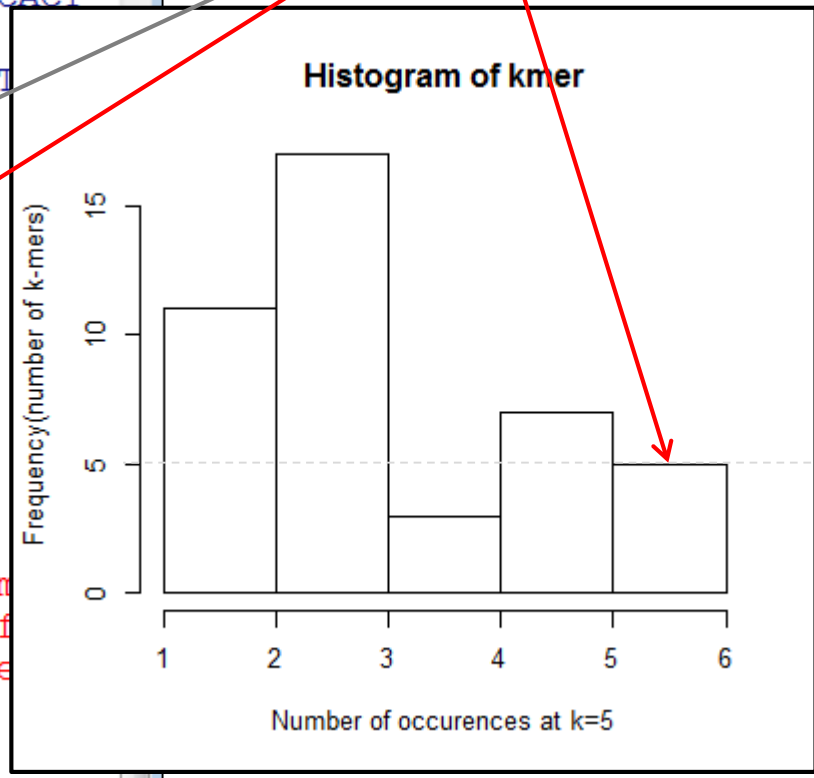
4X coverageであることが分かって
基長のリードを10個ランダム抽出
します。この場合は20塩基以下で

```
in_f <- "sample32_ngs.fa
out_f <- "hoge3.png"
param_k <- 5
param_fig <- c(400, 380)

#必要なパッケージをロード
library(qrqc)

#入力ファイルの読み込み
hoge <- readSeqFile(in_f
hoge

#本番(k-mer出現頻度解析)
kmer <- table(hoge@kmer$
kmer
length(kmer)
table(kmer)

#ファイルに保存(ヒストグラ
png(out_f, pointsize=13,
hist(kmer, ylab="Frequen
     xlab=paste("Number
```

R Console

```
> kmer                                                    $

AACTC AAGAC AAGTT ACACC ACATG ACCAG ACGCG
    6     3     1     2     3     3     2
ACTAC ACTCA AGACG AGGAC AGTTG ATGAA CACCA
    3     6     3     3     2     3     3
CACTA CAGGA CATGA CCAGG CGCGG CGGAC CTACA
    4     3     3     3     1     1     2
CTCAC CTTTT GAACT GAAGA GAAGT GACAT GACGC
    5     5     6     3     1     3     3
GCGGA GGACA GGACG GTCTT GTTGT TACAC TCACT
    1     3     1     5     3     2
TCTTT TGAAC TGAAG TGTCT TTGAA TTGTC TT
    5     6     3     4     6     4
TTTTG
    5
> length(kmer)
[1] 43
> table(kmer)
kmer
 1  2  3  4  5  6
 6  5 17  3  7  5
>
> #ファイルに保存(ヒストグラム)
> png(out_f, pointsize=13, width=param
> hist(kmer, ylab="Frequency(number of
+      xlab=paste("Number of occurence
> dev.off()
windows
      2
```

1回出現したk-merが6個、
6回出現したk-merが5個、
という解釈をする



3. サンプルデータ32のFASTA形式ファイル(sample32_ngs.fasta)の場合:

4X coverageであることが分かって
基長のリードを10個ランダム抽出
します。この場合は20塩基以下で

```
in_f <- "sample32_ngs.fa
out_f <- "hoge3.png"
param_k <- 5
param_fig <- c(400, 380)

#必要なパッケージをロード
library(qrqc)

#入力ファイルの読み込み
hoge <- readSeqFile(in_f
hoge

#本番(k-mer出現頻度解析)
kmer <- table(hoge@kmer$
kmer
length(kmer)
table(kmer)

#ファイルに保存(ヒストグラ
png(out_f, pointsize=13,
hist(kmer, ylab="Frequen
     xlab=paste("Number
```

R Console

```
> kmer                                                    $

AACTC AAGAC AAGTT ACACC ACATG ACCAG ACGCG
    6     3     1     2     3     3     2
ACTAC ACTCA AGACG AGGAC AGTTG ATGAA CACCA
    3     6     3     3     2     3     3
CACTA CAGGA CATGA CCAGG CGCGG CGGAC CTACA
    4     3     3     3     1     1     2
CTCAC CTTTT GAACT GAAGA GAAGT GACAT GACGC
    5     5     6     3     1     3     3
GCGGA GGACA GGACG GTCTT GTTGT TACAC TCACT
    1     3     1     5     3     2     5
TCTTT TGAAC TGAAG TGTCT TTGAA TTGTC TTTGA
    5     6     3     4     6     4     5
TTTTG
    5
> length(kmer)
[1] 43
> table(kmer)
kmer
 1  2  3  4  5  6
 6  5 17  3  7  5
>
> #ファイルに保存(ヒストグラム)
> png(out_f, pointsize=13, width=param_fi$
> hist(kmer, ylab="Frequency(number of k-$
+      xlab=paste("Number of occurences a$
> dev.off()                                               $
windows
      2
```

入力ファイル中にAAGACという部分文字列は3つ存在するということ

```
>kkk_24_43
CACCAGGACATGAAGACGCG
>kkk_28_47
AGGACATGAAGACGCGGACG
>kkk_12_31
TTGAACTCACTACACCAGGA
>kkk_4_23
GTTGTCTTTTGAACTCACTA
>kkk_5_24
TTGTCTTTTGAACTCACTAC
>kkk_7_26
GTCTTTTGAACTCACTACAC
>kkk_12_31
TTGAACTCACTACACCAGGA
>kkk_23_42
ACACCAGGACATGAAGACGC
>kkk_1_20
GAAGTTGTCTTTTGAACTCA
>kkk_3_22
AGTTGTCTTTTGAACTCACT
```

リマインドです

- 過去のお知らせ (last modified 2014/06/13) NEW
- Rのインストールと起動 (last modified 2014/0
- サンプルデータ (last modified 2014/06/17) N
- 書籍 | トランスクリプトームについて (last mod
- 書籍 | トランスクリプトーム解析 | 2.3.1 RNA-seqデータ(FASTQファイル)

```
>kkk
GAAGTTGTCTTTTGAACTCACTACACCAGGACATGAAGACGCGGACGGCA
```

- サンプルデータ

サンプルデータ NEW

1. Illumina/36bp/single-end/human (SRA000299) data (Marioni et al., Genome Res., 2008)
「Kidney 7 samples vs Liver 7 samples」のRNA-seqの遺伝子発現行列データ

32. k-mer解析用のランダム配列から生成したFASTA形式ファイル(sample32_ref.fastaとsample32_ngs.fasta)です。50塩基の長さのリファレンス配列を生成したのち、20塩基長の部分配列を10リード分だけランダム抽出したものです。塩基の存在比はAが22%, Cが28%, Gが28%, Tが22%にしています。リファレンス配列(仮想ゲノム配列)がsample32_ref.fastaで、10リードからなる仮想NGSデータがsample32_ngs.fastaです。リード長20塩基で10リードなのでトータル200塩基となり、50塩基からなる元のゲノム配列の4倍シーケンスしていることになります(4X coverageに相当)。イントロ | NGS | 配列取得 | シミュレーションデータ | ランダムな塩基配列の生成からと基本的に同じです。

```
out_f1 <- "sample32_ref.fasta"          #出力ファイル名を指定してout_f1に格納
out_f2 <- "sample32_ngs.fasta"          #出力ファイル名を指定してout_f2に格納
param_len_ref <- 50                     #リファレンス配列の長さを指定
narabi <- c("A","C","G","T")            #以下の数値指定時にACGTの並びを間違えないように
param_composition <- c(22, 28, 28, 22) #(A,C,G,Tの並びで)各塩基の存在比率を指定
param_len_ngs <- 20                     #リード長を指定
param_num_ngs <- 10                     #リード数を指定
param_desc <- "kkk"                     #FASTA形式ファイルのdescription行に記述する内

#必要な
library

#本番(
set.se
ACGTse
refere                                                    ACGTset
reference <- DNAStringSet(reference)    #DNA塩基配列だと認識させるDNAStringSet関数を通
names(reference) <- param_desc          #description行に相当する記述を追加している
reference                               #確認してるだけです

#本番(シミュレーションデータ生成)
s_posi <- sample(1:(param_len_ref-param_len_ngs), param_num_ngs, replace=T)#部分塩
```

sample32_ngs.fastaは、50塩基からなるランダム塩基配列をリファレンスとしています。20塩基からなる10個の部分配列をリードとしています。総塩基数は200なのでリファレンス配列の4倍の長さ、つまり4X coverageです。

```
>kkk_24_43
CACCAGGACATGAAGACGCG
>kkk_28_47
AGGACATGAAGACGCGGACG
>kkk_12_31
TTGAACTCACTACACCAGGA
>kkk_4_23
GTTGTCTTTTGAACTCACTA
>kkk_5_24
TTGTCTTTTGAACTCACTAC
>kkk_7_26
GTCTTTTGAACTCACTACAC
>kkk_12_31
TTGAACTCACTACACCAGGA
>kkk_23_42
ACACCAGGACATGAAGACGC
>kkk_1_20
GAAGTTGTCTTTTGAACTCA
>kkk_3_22
AGTTGTCTTTTGAACTCACT
```

- 過去のお知らせ (last modified 2014/06/13) NEW
- Rのインストールと起動 (last modified 2014/05/1
- サンプルデータ modified 2014/06/17) NEW
- 書籍 | トランスクリプトームについて (last modifie
- 書籍 | トランスクリプトーム解析 | 2.3.1 RNA-seq

sample36_ngs.fastaは、10,000塩基からなるランダム塩基配列をリファレンスとしています。80塩基からなる5,000個の部分配列をリードとしています。総塩基数は400,000なのでリファレンス配列の40倍の長さ、つまり**40X coverage**です。

- サンプルデータ

サンプルデータ NEW

36. k-mer解析用のランダム配列から生成したFASTA形式ファイル(sample36_ref.fastaとsample36_ngs.fasta)です。
10000塩基の長さのリファレンス配列を生成したのち、80塩基長の部分配列を5000リード分だけランダム抽出したものです。塩基の存在比はAが22%, Cが28%, Gが28%, Tが22%にしています。リファレンス配列(仮想ゲノム配列)がsample36_ref.fastaで、5,000リードからなる仮想NGSデータがsample36_ngs.fastaです。リード長80塩基で5,000リードなのでトータル400,000塩基となり、10,000塩基からなる元のゲノム配列の40倍シーケンスしていることになります(40X coverageに相当)。イントロ | NGS | 配列取得 | シミュレーションデータ | ランダムな塩基配列の生成からと基本的に同じです。

```
out_f1 <- "sample36_ref.fasta"
out_f2 <- "sample36_ngs.fasta"
param_len_ref <- 10000
narabi <- c("A","C","G","T")
param_composition <- c(22, 28, 28, 22)
param_len_ngs <- 80
param_num_ngs <- 5000
param_desc <- "kkk"

#必要なパッケージをロード
library(Biostrings)

#本番(リファレンス配列生成)
set.seed(1010)
ACGTset <- rep(narabi, param_compositic
reference <- paste(sample(ACGTset, para
reference <- DNAStringSet(reference)
names(reference) <- param_desc
reference

#本番(シミュレーションデータ生成)
s_posi <- sample(1:(param_len_ref-paran
```

R Console

```
> fasta                                          #確認してるだけです
  A DNAStringSet instance of length 5000
       width seq                                                names          $
   [1]    80 ACGGTATGCAACCCTATTTA...CTATAAAGCCGAGCGGCCCA kkk_634_713
   [2]    80 TACCTTGCACAAGCTCATAG...AGCACAAGCCTGGATGTCAT kkk_7573_7652
   [3]    80 CACCGATCCTGCATTCAAAC...CCGGGAGTCCCGTGGATAAT kkk_3393_3472
   [4]    80 GGCATTAAGTTCGTTCATTA...CCGATCCTCATACATGCAGA kkk_7257_7336
   [5]    80 ATAGAGTAGAAAATGTATCT...TCATACAGAGTTTACATCGT kkk_7589_7668
   ...   ... ...
[4996]    80 GGCCGCTAACACAGGCTGAC...TACCCTGGGGGCCGCCATGA kkk_2710_2789
[4997]    80 CTGATCGTGGTTTTACCTGA...GCTCGAGGGCTTTCGGCGTA kkk_1842_1921
[4998]    80 ATTCGACCCCCGTCTTGACC...GCACCGTGGTTGAGCACACC kkk_7031_7110
[4999]    80 AGTATCCCTGGCAAGGTCCG...AAATCAAGGGGACCACATGG kkk_7804_7883
[5000]    80 TGGCGTTGGTCCCCCCGGAG...TGCCCCCGGATGATTCCACC kkk_2987_3066
>
> #ファイルに保存(仮想リファレンス配列と仮想NGS配列)
> writeXStringSet(reference, file=out_f1, format="fasta", width=50)#re$
> writeXStringSet(fasta, file=out_f2, format="fasta", width=50)#fasta$
> reference
  A DNAStringSet instance of length 1
    width seq                                              names          $
[1] 10000 GAAGTTGTCTTTTGAACTCACT...CCTAACTGATTGCTAAAGCGT kkk
>
```

7. サンプルデータ36のFASTA形式ファイル(sample36_ngs.fasta)の場合:

40X coverageであることが分かっているデータです。理由は、10,000塩基長のリファレンス配列から80塩基長のリードを5,000個ランダム抽出したものだからです。k-merのkの値は、リード配列長以下に設定します。この場合は80塩基以下ですので21を指定しています。

```
in_f <- "sample36_ngs.fasta"
out_f <- "hoge7.png"
param_k <- 21
param_fig <- c(400, 380)

#必要なパッケージをロード
library(qrqc)

#入力ファイルの読み込み
hoge <- readSeqFile(in_f, type
hoge

#本番(k-mer出現頻度解析)
kmer <- table(hoge@kmer$kmer)
kmer
length(kmer)
table(kmer)

#ファイルに保存(ヒストグラム)
png(out_f, pointsize=13, width
hist(kmer, ylab="Frequency(num
     xlab=paste("Number of occ
```

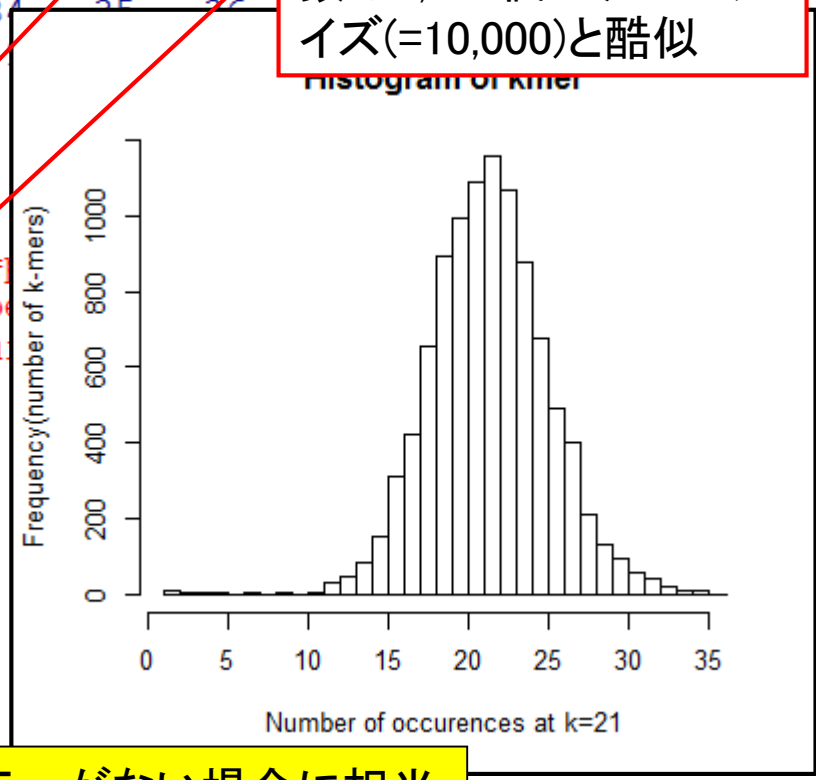
R Console

```
> table(kmer)                                    #k-$
kmer
   1    2    3    4    5    6    7    8    9
   6    8    9    2    6    2    3    2    8
  10   11   12   13   14   15   16   17   18
  11   17   36   54   79  190  362  508  712
  19   20   21   22   23   24   25   26   27
 751  851  926  998 1040  978  760  603  430
  28   29   30   31   32   33   34   35   36
 270  153   86   42   34   25
  37
   2
>
> #ファイルに保存(ヒストグラム)
> png(out_f, pointsize=13, width=
> hist(kmer, ylab="Frequency(numbe
+      xlab=paste("Number of occu
+      breaks=max(kmer))
> dev.off()
null device
          1
> median(kmer)
[1] 22
> length(kmer)
[1] 9978
>
```

(k-merの種類は問わずに)37回出現したk-merが2個あったということ

出現回数のmedianは22

可能なk-merの種類数は4^21個。実際の種類数は9,978個でゲノムサイズ(=10,000)と酷似



40Xの分布。シークエンスエラーがない場合に相当

7. サンプルデータ36のFASTA形式ファイル(sample36_ngs.fasta)の場合:

40X coverageであることが分かっているデータです。理由は、10,000塩基長のリファレンス配列から80塩基長のリードを5,000個ランダム抽出したものだからです。k-merのkの値は、リード配列長以下に設定します。この場合は80塩基以下ですので21を指定しています。


```
in_f <- "sample36_ngs.fasta"
out_f <- "hoge7.png"
param_k <- 21
param_fig <- c(400, 380)

#必要なパッケージをロード
library(qrqc)

#入力ファイルの読み込み
hoge <- readSeqFile(in_f, type
hoge

#本番(k-mer出現頻度解析)
kmer <- table(hoge@kmer$kmer)
kmer
length(kmer)
table(kmer)

#ファイルに保存(ヒストグラム)
png(out_f, pointsize=13, width
hist(kmer, ylab="Frequency(num
     xlab=paste("Number of occ
```

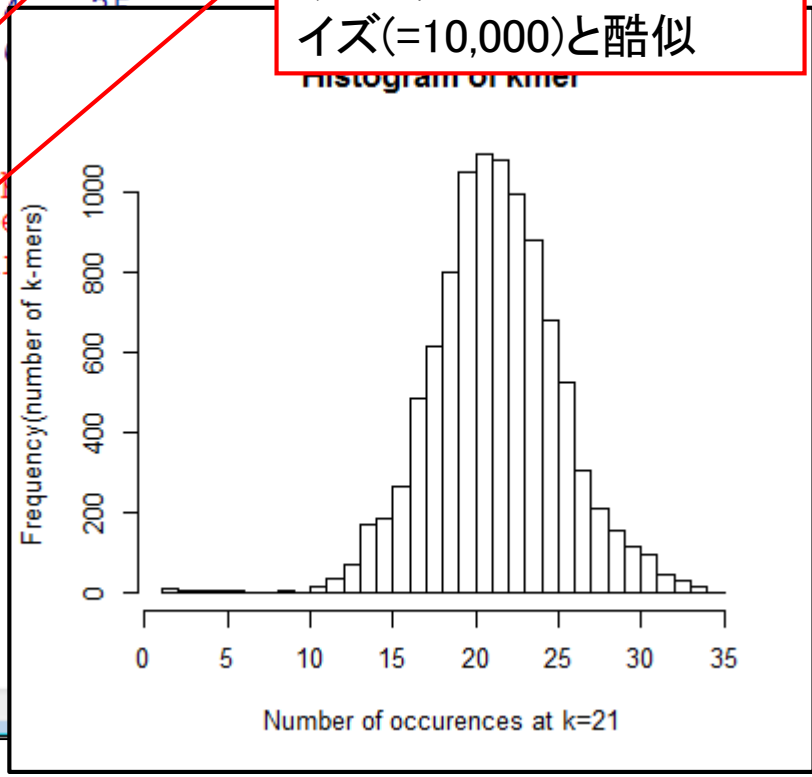
R Console

```
> table(kmer)                                    #k-$
kmer
   1    2    3    4    5    6    7    8    9
   6    5    5    6    4    4    3    2    8
  10   11   12   13   14   15   16   17   18
   3   17   35   69  170  184  265  486  613
  19   20   21   22   23   24   25   26   27
 802 1051 1094 1079  995  881  681  527  307
  28   29   30   31   32   33   34
 211  158  115   97   47   29   1
>
> #ファイルに保存(ヒストグラム)
> png(out_f, pointsize=13, width=
> hist(kmer, ylab="Frequency(numbe
+      xlab=paste("Number of occu
+      breaks=max(kmer))
> dev.off()
null device
          1
> median(kmer)
[1] 22
> length(kmer)
[1] 9976
>
```

(k-merの種類は問わずに)31回出現したk-merが97個あったということ

出現回数のmedianは22

可能なk-merの種類数は4^21個。実際の種類数は9,976個でゲノムサイズ(=10,000)と酷似



40Xの分布。シークエンスエラーがない場合に相当。**実行ごとに結果が多少異なります**が、どこかで計算をさぼらないといけない現実的な理由があるためかも(バグかも)…。

8. サンプルデータ37のFASTA形式ファイル(sample37_ngs.fasta)の場合:

100X coverageであることが分かっているデータです。理由は、10,000塩基長のリファレンス配列から100塩基長の10,000個ランダム抽出したものだからです。k-merのkの値は、リード配列長以下に設定します。この場合は100塩基以下ですので、条件を満たす21を指定しています。hist関数実行時に分割数を指定すべくbreaksオプションも変更しています。

• 解析 | 基礎 | k-mer | ゲノムサイズ推定(基礎) | qrqc

```
in_f <- "sample37_ngs.fasta"
out_f <- "hoge8.png"
param_k <- 21
param_fig <- c(400, 380)

#必要なパッケージをロード
library(qrqc)

#入力ファイルの読み込み
hoge <- readSeqFile(in_f, type
hoge

#本番(k-mer出現頻度解析)
kmer <- table(hoge@kmer$kmer)
kmer
length(kmer)
table(kmer)

#ファイルに保存(ヒストグラム)
png(out_f, pointsize=13, width
hist(kmer, ylab="Frequency(num
     xlab=paste("Number of occ
```

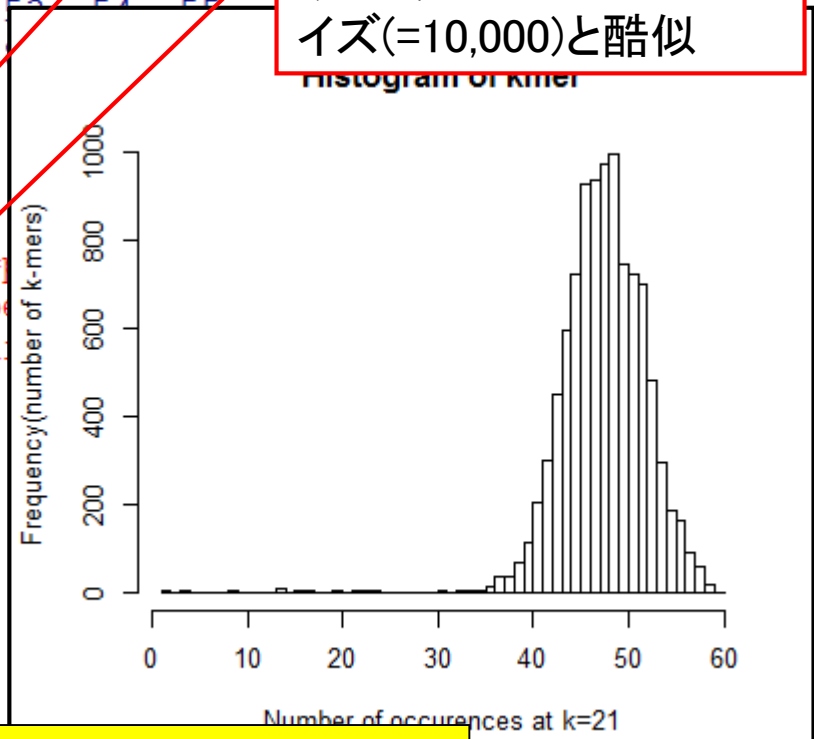
R Console

```
  1   2   3   4   5   6   7   8   9  10  11
  2   4   3   4   2   3   2   2   4   2   3
 12  13  14  15  16  17  18  19  20  21  22
  3   2   9   2   5   4   2   3   5   3   4
 23  24  25  26  27  28  29  30  31  32  33
  4   5   3   3   2   2   3   3   4   3   4
 34  35  36  37  38  39  40  41  42  43  44
  4   5  14  35  38  70 114 206 302 451 596
 45  46  47  48  49  50  51  52
725 927 937 975 995 748 724 701 4
 56  57  58  59  60
165  93  59  19   1
>
> #ファイルに保存(ヒストグラム)
> png(out_f, pointsize=13, width=
> hist(kmer, ylab="Frequency(numbe
+      xlab=paste("Number of occu
+      breaks=max(kmer))
> dev.off()
null device
          1
> median(kmer)
[1] 48
> length(kmer)
[1] 9978
```

(k-merの種類は問わずに)59回出現したk-merが19個あったということ

出現回数のmedianは48

可能なk-merの種類数は4^21個。実際の種類数は9,978個でゲノムサイズ(=10,000)と酷似



100Xの分布。シークエンスエラーがない場合に相当。coverageの増加(40X → 100X)に伴い、出現頻度分布x軸方向で右側にシフトしていることがわかる

9. サンプルデータ37のFASTA形式ファイル(sample37_ngs.fasta)の場合:

100X coverageであることが分かっているデータです。理由は、10,000塩基長のリファレンス配列から100塩基長のリ・解析 | 基礎 | k-mer | ゲノムサイズ推定(基礎) | qrqc
10,000個ランダム抽出したものだからです。k-merのkの値は、リード配列長以下に設定します。この場合は100塩基以下です
ので、条件を満たす31を指定しています。hist関数実行時に分割数を指定すべくbreaksオプションも変更しています。

```
in_f <- "sample37_ngs.fasta"
out_f <- "hoge9.png"
param_k <- 31
param_fig <- c(400, 380)

#必要なパッケージをロード
library(qrqc)

#入力ファイルの読み込み
hoge <- readSeqFile(in_f, type
hoge

#本番(k-mer出現頻度解析)
kmer <- table(hoge@kmer$kmer)
kmer
length(kmer)
table(kmer)

#ファイルに保存(ヒストグラム)
png(out_f, pointsize=13, width
hist(kmer, ylab="Frequency(num
     xlab=paste("Number of occ
```

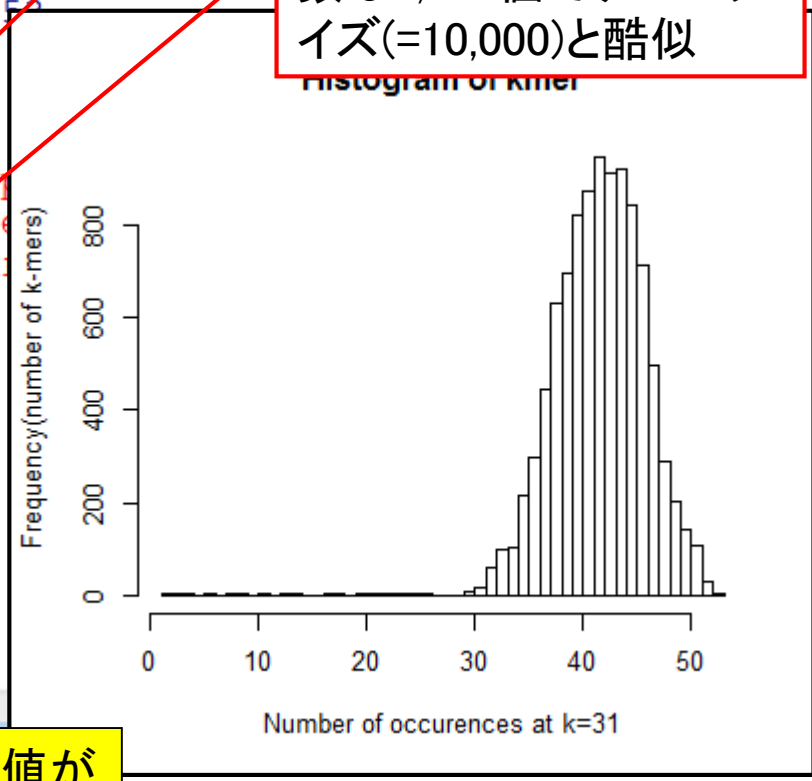
R Console

```
  1   2   3   4   5   6   7   8   9  10  11
  2   4   3   4   2   3   2   3   4   2   4
 12  13  14  15  16  17  18  19  20  21  22
  2   6   3   2   2   5   3   2   3   5   3
 23  24  25  26  27  28  29  30  31  32  33
  4   4   5   5   2   2   2  10  18  60 101
 34  35  36  37  38  39  40  41  42  43  44
105 217 300 444 630 693 818 872 944 912 919
 45  46  47  48  49  50  51  52  53
840 711 498 290 204 145 108  33
>
> #ファイルに保存(ヒストグラム)
> png(out_f, pointsize=13, width=
> hist(kmer, ylab="Frequency(numbe
+      xlab=paste("Number of occu
+      breaks=max(kmer))
> dev.off()
null device
          1
> median(kmer)
[1] 42
> length(kmer)
[1] 9968
>
```

(k-merの種類は問わずに)26回出現したk-merが5個あったということ

出現回数のmedianは42

可能なk-merの種類数は4^31個。実際の種類数は9,968個でゲノムサイズ(=10,000)と酷似



100Xの分布。シークエンスエラーがない場合に相当。k-merの値が変わっても(k=21 → 31)、k-merの種類数はゲノムサイズとほぼ同じ

10. サンプルデータ37のFASTA形式ファイル(sample37_ngs.fasta)の場合:

100X coverageであることが分かっているデータです。理由は、10,000塩基長のリファレンス配列から100塩基長の• 解析 | 基礎 | k-mer | ゲノムサイズ推定(基礎) | qrqc
個ランダム抽出したものだからです。k-merのkの値は、リード配列長以下に設定します。この場合は100塩基以下ですので、
を満たす11を指定しています。hist関数実行時に分割数を指定すべくbreaksオプションも変更しています。

```
in_f <- "sample37_ngs.fasta"
out_f <- "hoge10.png"
param_k <- 11
param_fig <- c(400, 380)

#必要なパッケージをロード
library(qrqc)

#入力ファイルの読み込み
hoge <- readSeqFile(in_f, type
hoge

#本番(k-mer出現頻度解析)
kmer <- table(hoge@kmer$kmer)
kmer
length(kmer)
table(kmer)

#ファイルに保存(ヒストグラム)
png(out_f, pointsize=13, width
hist(kmer, ylab="Frequency(num
     xlab=paste("Number of occ
```

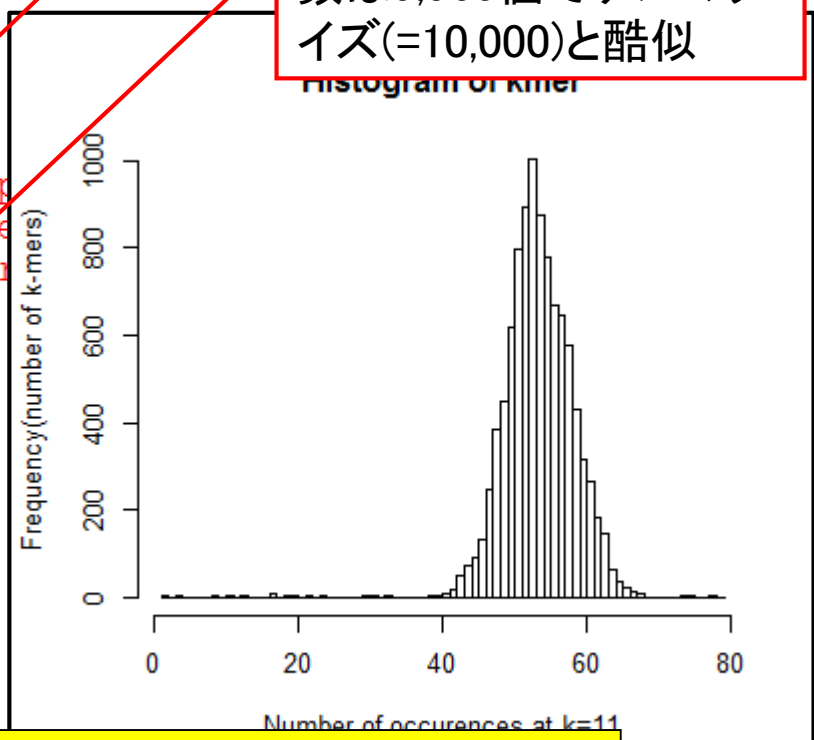
R Console

```
  37   38   39   40   41   42   43   44   45
   2    2    6    5    8   18   50   74   91
  46   47   48   49   50   51   52   53   54
 135  246  383  451  619  799  895 1002  876
  55   56   57   58   59   60   61   62   63
 780  671  647  577  429  316  264  182  147
  64   65   66   67   68   73   74   75   77
  63   39   25   15    8    2    6    5    1
  78   79
   4    2
>
> #ファイルに保存(ヒストグラム)
> png(out_f, pointsize=13, width=p
> hist(kmer, ylab="Frequency(numbe
+      xlab=paste("Number of occu
+      breaks=max(kmer))
> dev.off()
null device
          1
> median(kmer)
ACCGGAGGAGC
         54
> length(kmer)
[1] 9965
>
```

(k-merの種類は問わずに)67回出現したk-merが15個あったということ

出現回数のmedianは54

可能なk-merの種類数は4^11個。実際の種類数は9,965個でゲノムサイズ(=10,000)と酷似



100Xの分布。シークエンスエラーがない場合に相当。k-merの値が変わっても(k=31 → 11)、k-merの種類数はゲノムサイズとほぼ同じだが、実際にはエラーを多く含むので非現実的。

k-mer解析は様々な場面で利用されます

① 大まかなゲノムサイズ推定（とカバレッジ）

- 現実のNGSデータはシークエンスエラーやリピート配列を含むため正規分布っぽくはならないが主要なピークの位置情報をもとにゲノムサイズを推定可能（らしい）
- リード長とリード数から得られたNGSデータ中の総塩基数情報はわかっているので、得られたゲノムサイズで割ることでカバレッジ(coverage)が分かる

k = 41, 51, 61

k = 41, 51, 61

k = 41, 51, 61

S. aureus
ゲノムサイズ：2.8 Mb
Coverage： 167X
リード数：5,000,000
リード長：101 bp

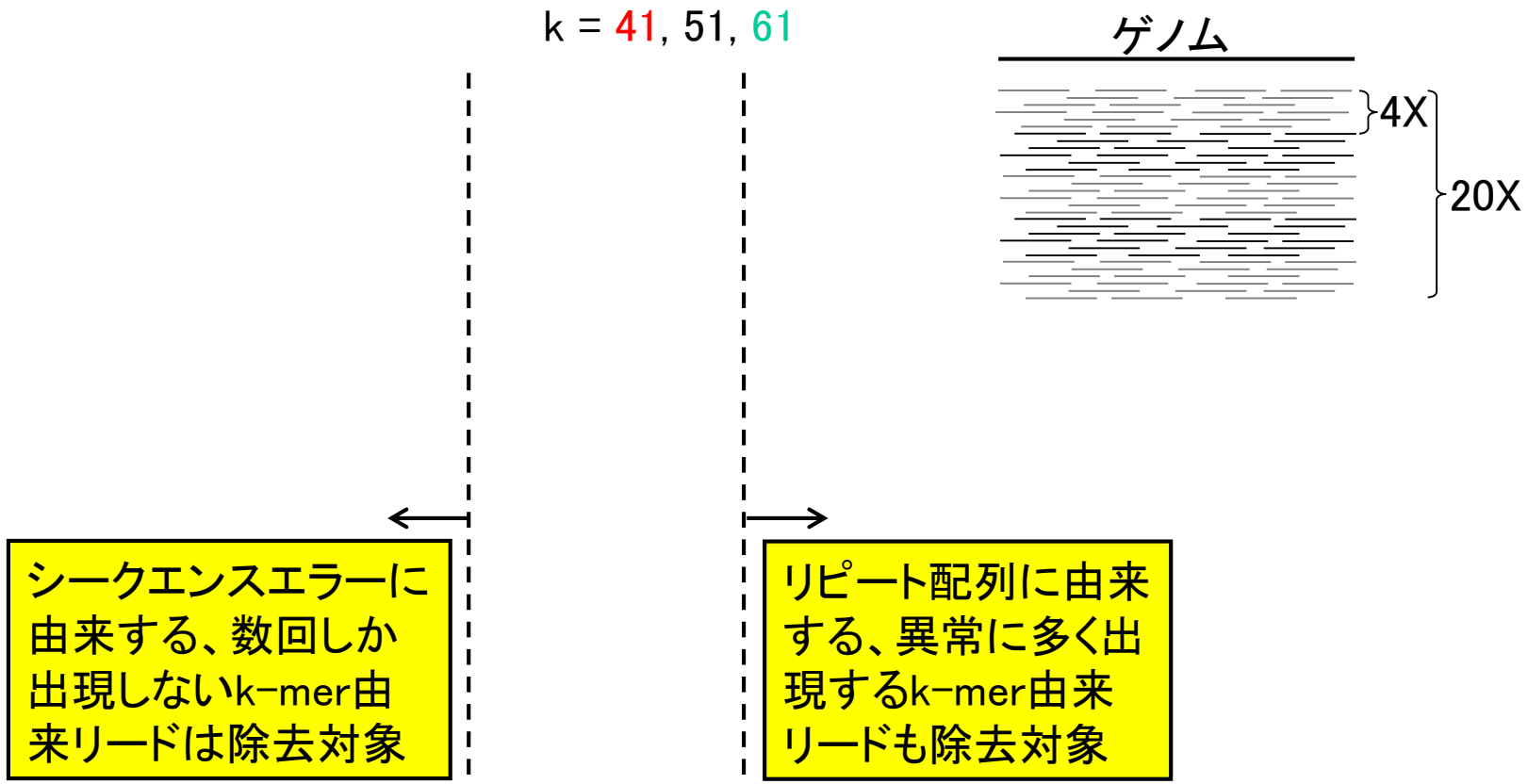
H. sapiens chr 14
ゲノムサイズ：88 Mb
Coverage： 70X
リード数：62,000,000
リード長：101 bp

B. impatiens
ゲノムサイズ：250 Mb
Coverage： 247X
リード数：497,000,000
リード長：124 bp

k-mer解析は様々な場面で利用されます

② 前処理（フィルタリング）

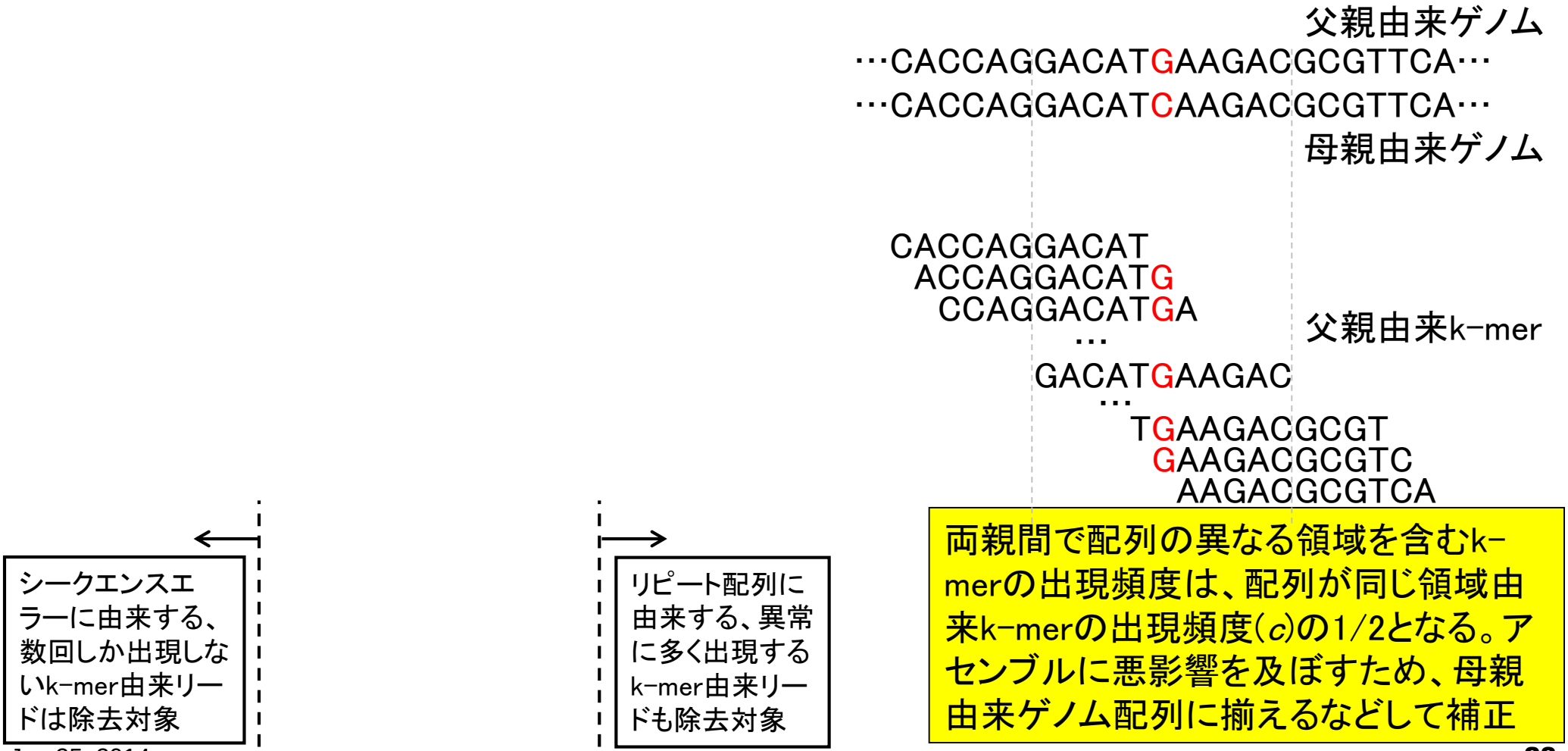
- シークエンスエラーやリピート配列由来リードの除去



k-mer解析は様々な場面で利用されます

③ 前処理（エラー補正）

- ヘテロ接合度の高い2倍体ゲノム(highly heterozygous diploid genomes)由来リードの補正



ゲノムアセンブルの手順

1. 前処理(pre-processing filtering)
   - クオリティの低いリードやコンタミを除去するステップ。塩基置換(substitution)やインデル(indels; insertion/deletion)を含むリードの除去や補正(error correction)。
   - 4つのアプローチ：k-mer, suffix tree/array, multiple sequence alignment, hybrid
2. **グラフ構築(graph construction)**
   - **前処理後のリードを用いて、リード間のオーバーラップ(overlap)を頼りにつなげていくステップ。シークエンスエラー(sequencing error)と多型(polymorphism)の違いを見るべく、グラフ構築時にエラー補正を行うものもある。**
   - **4つのアプローチ：OLC, de Bruijn graph (k-mer), greedy, hybrid**
3. グラフ簡易化(graph simplification)
   - グラフ構築後に、複雑化したグラフをシンプルにしていくステップ。連続したノード(nodes; 頂点) やバブルのマージ作業に相当。
4. 後処理(post-processing)
   - コンティグ(contigs)やスカッフォールド(scaffolds)を得るステップ。ミスアセンブリの同定も含む。

大きく分けて4つの手順からなる

- 前処理 | トリミング | 指定した末端塩基数だけ除去 (last modified 2013/06/15)
- アセンブル | について (last modified 2014/06/16) NEW
- アセンブル | ゲノム用
- アセンブル | トランスク
- アセンブル | ゲノム既
- マッピング | 備忘録 |
- マッピング | 備忘録 |
- マッピング | 備忘録 |
- マッピング | 備忘録 |
- マッピング | 備忘録 |
- マッピング | 基礎 (last
- マッピング | single-end
- マッピング | single-end

• アセンブル | について

2. グラフ構築(graph construction process):

ここで行うのは、前処理後のリードを用いてリード間のオーバーラップ(overlap)を頼りにつなげていく作業です。シークエンスエラー(sequencing error)と多型(polymorphism)の違いを見るべく、グラフ構築時にエラー補正を行うものもあります。おそらく全てのアセンブルプログラムはグラフ理論(Graph theory)を用いています。一筆書きの問題をグラフ化して解くような学問です。それらはさらに4つのアプローチに大別できます: Overlap (OLC), de Bruign (k-mer), Greedy, Hybrid。

2-1. Overlap-Layout-Consensus (OLC)アプローチ。アセンブル問題をハミルトンパス(Hamiltonian path)問題として解く、ABI3730 sequencerの頃から存在する伝統的な方法。overlap, layout, and consensusの3つのステップからなるためOLCと略される。454など比較的長い配列(数百塩基程度)のアセンブルを目的としたものが多いようですが、最近ではSGAやReadjoinerなどショートリードにもうまく対応したものが出てきているようです。

- Celera Assembler(OLC graph): Myers et al., Science, 2000
- ARACHNE(OLC graph): Batzoglou et al., Genome Res, 2002
- PCAP(OLC graph): Huang et al., Genome Res., 2003
- Newbler(OLC graph): Margulies et al., Nature, 2005
- Edena(OLC graph): Hernandez et al., Genome Res., 2008
- CABOG(MSA correction; OLC graph): Miller et al., Bioinformatics, 2008
- SHORTY(OLC graph; scaffolder): Hossain et al., BMC Bioinformatics, 2009
- Forge(OLC graph): Diguistini et al., Genome Biol., 2009
- SGA(k-mer correction; OLC graph; scaffolder): Simpson et al., Genome Res., 2012
- Readjoiner(k-mer correction; OLC graph): Gonnella et al., BMC Bioinformatics, 2012
- Fermi(OLC graph): Li H., Bioinformatics, 2012

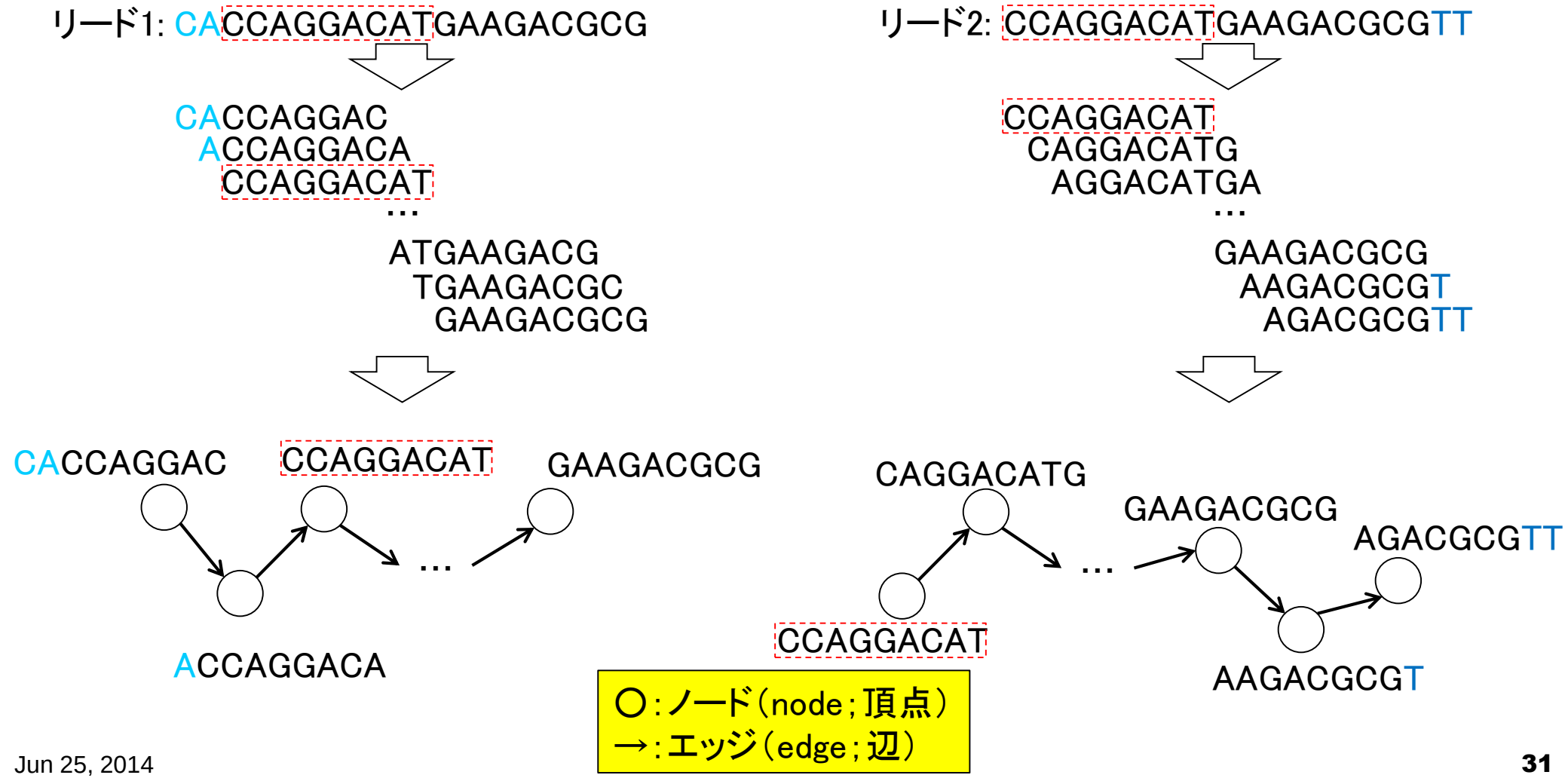
2-2. de Bruign (k-mer)アプローチ。これは、de Bruijn graph (DBG)中のオイラーパス探索(DBG approach; Eulerian approach; Euler approach; Eulerian path approach)に基づくもので、グラフは頂点(ノード; nodes or vertices; 一つ一つの配列に相当)と辺(エッジ; edges or arcs)で表されますが、リードを1塩基づつずらして全ての可能なk-mer (all possible fixed k length strings; k個の連続塩基のことでkは任意の正の整数)を生成し各k-merをノードとした有向グラフ(k-merグラフ)を作成します。全リードに対して同様の作業を行い、完全一致ノードをマージして得られるグラフがDBGです。そしてこのグラフは各エッジを一度だけ通るオイラーパス(Eulerian path)をもつことが分かっているので、あとは既知のオイラーパス問題専用アルゴリズムを適用するというアプローチです。どのk-merの値を用いればいいかはなかなか難しいようですが、KmerGenie (Chikh et al., Bioinformatics, 2014)という最適なkの値を見積もってくれるプログラムもあるようです。

- Velvet(k-mer graph; scaffolder): Zerbino and Birney, Genome Res., 2008
- EULER-SR(k-mer correction; k-mer graph; scaffolder): Chaisson et al., Genome Res., 2009
- ABySS(k-mer graph): Simpson et al., Genome Res., 2009
- ALLPATHS-LG(k-mer correction; k-mer graph; scaffolder): Maccallum et al., Genome Biol., 2009
- SOAPdenovo(k-mer correction; k-mer graph; scaffolder): Li et al., Genome Res., 2010
- SparseAssembler(k-mer graph): Ye et al., BMC Bioinformatics, 2012
- Platanus(k-mer correction; k-mer graph; scaffolder): Kajitani et al., Genome Res., 2014

様々な戦略がありますが、k-merを利用する方法の基本を紹介します

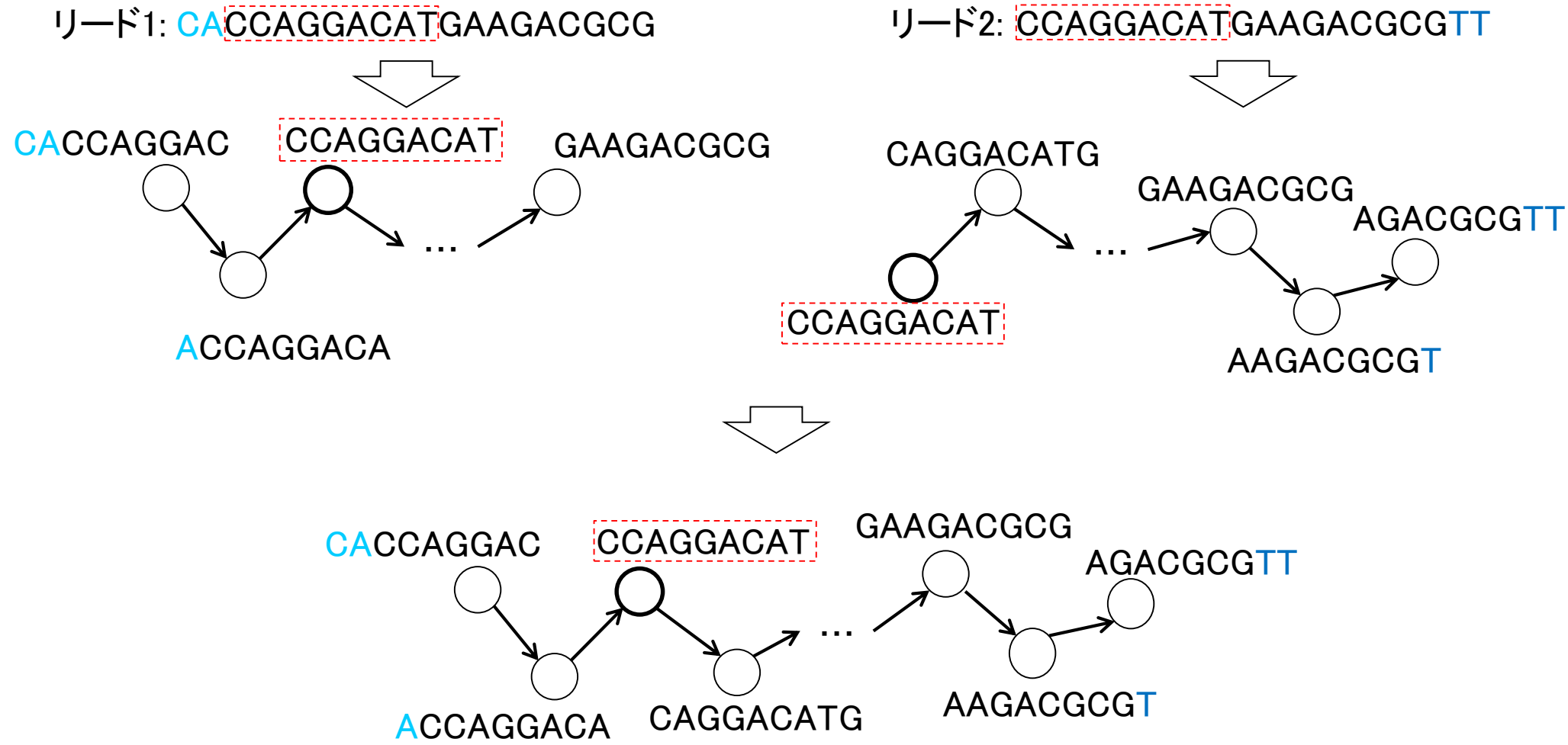
2.グラフ構築(graph construction)

- k-merアプローチ(de Bruijnグラフ)
  - リードを全ての可能なk-merに分割し、有向グラフを作成(k=9の例)



2.グラフ構築(graph construction)

- k-merアプローチ(de Bruijnグラフ)
  - 同一ノードをマージしてde Bruijnグラフを作成(k=9)



ゲノムアセンブルの手順

1. 前処理(pre-processing filtering)
   - クオリティの低いリードやコンタミを除去するステップ。塩基置換(substitution)やインデル(indels; insertion/deletion)を含むリードの除去や補正(error correction)。
   - 4つのアプローチ：k-mer, suffix tree/array, multiple sequence alignment, hybrid
2. グラフ構築(graph construction)
   - 前処理後のリードを用いて、リード間のオーバーラップ(overlap)を頼りにつなげていくステップ。シークエンスエラー(sequencing error)と多型(polymorphism)の違いを見るべく、グラフ構築時にエラー補正を行うものもある。
   - 4つのアプローチ：OLC, k-mer (de Bruijn graph), greedy, hybrid
3. **グラフ簡易化(graph simplification)**
   - **グラフ構築後に、複雑化したグラフをシンプルにしていくステップ。連続したノード(nodes; 頂点) やバブルのマージ作業に相当。**
4. 後処理
   - コンテ
     同定も含む。

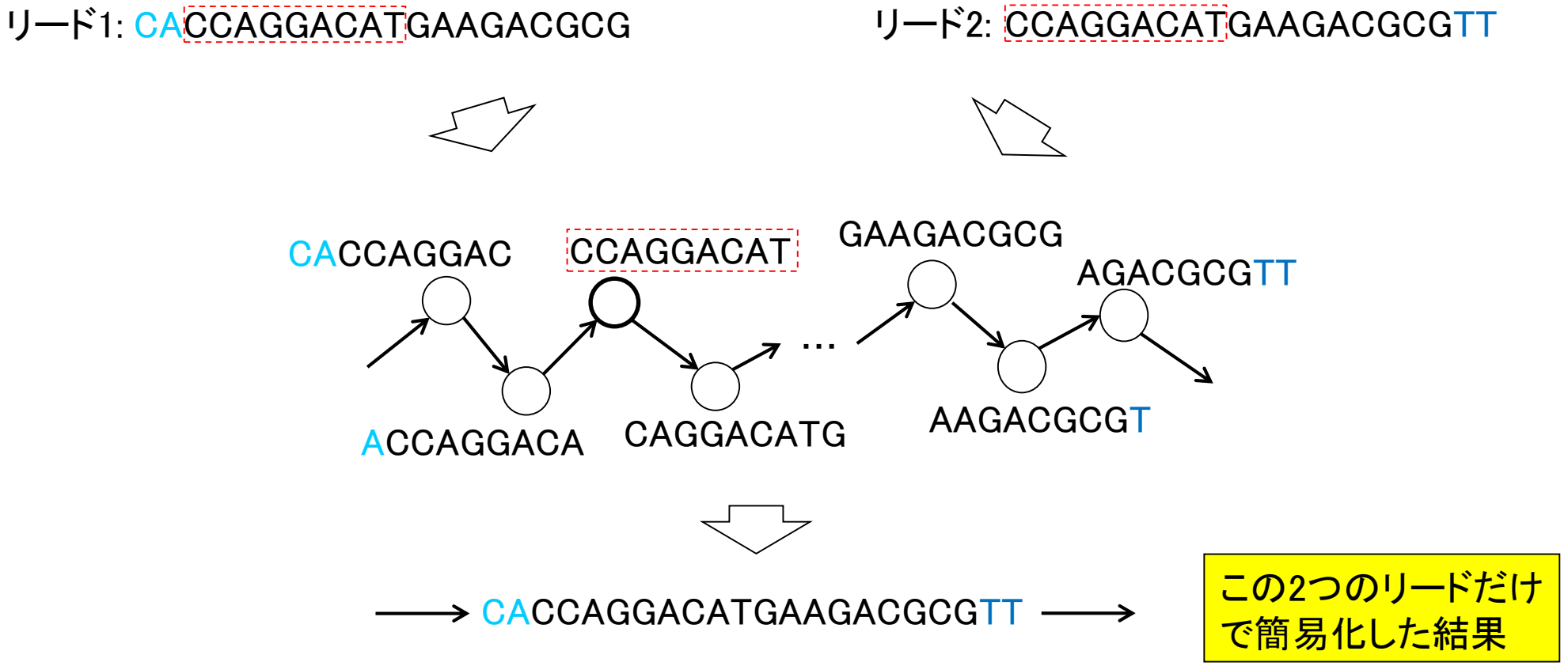
3. グラフ簡易化(graph simplification process):

ここで行うのは、グラフ構築後に複雑化したグラフをシンプルにしていく作業です。連続したノード(consecutive node) やバブル (bubble)のマージ作業に相当します。Dead endの除去やリピート領域で形成されるX-cutを2つのパスに切り分ける作業なども含みます。

大きく分けて4つの手順からなる

3. グラフ簡易化(graph simplification)

- 連続したノード(nodes; 頂点) やバブルのマージ



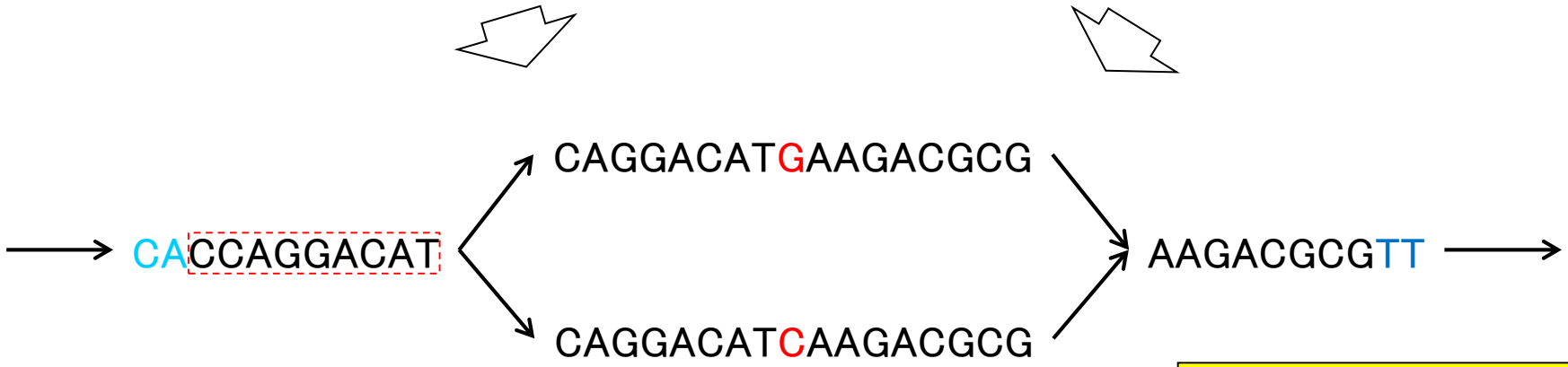
3. グラフ簡易化(graph simplification)

- 連続したノード(nodes; 頂点) やバブルのマージ

父親由来ゲノム
…CACCAGGACATGAAGACGCGTTCA…
…CACCAGGACATCAAGACGCGTTCA…
母親由来ゲノム

父親由来ゲノム
リード1: CACCAGGACATGAAGACGCG

リード2: CCAGGACATCAAGACGCGTT



SNPなど塩基に違いがあればバブル構造になります

ゲノムアセンブルの手順

1. 前処理(pre-processing filtering)
   - クオリ
     ンデル
   - 4つの
2. グラフ構
   - 前処理
     いくス
     を見る
   - 4つの
3. グラフ簡
   - グラフ
     (nodes; 頂点) やペアのマージ作業に相当。
4. **後処理(post-processing)**
   - **コンティグ(contigs)やスカッフォールド(scaffolds)を得るステップ。ミスアセンブリの同定も含む。**

4. 後処理(post-processing filtering):

ここで行うのは、簡易化後のグラフを一筆書き(渡り歩く、とか横切る、というイメージでよい;これがトラバース)してコンティグを得る作業です。paired-endリード情報を用いて、コンティグ同士を連結させたスーパーコンティグ(super-contigs)またはスカッフォールド(scaffolds)構築や、ミスアセンブリの同定も含みます。2.のアセンブラの中にscaffolderを書いてあるのはscaffolding moduleを持っているものたちです(El-Metwally et al., PLoS Comput Biol., 2013)。記載もれもあるとは思います。以下に示すのは、scaffoldingのみを単独で行うプログラムたちです。

- Bambus: Pop et al., Genome Res., 2004
- SOPRA: Dayarian et al., BMC Bioinformatics, 2010
- SSPACE: Boetzer et al., Bioinformatics, 2011
- Opera: Gao et al., J Comput Biol., 2011
- MIP Scaffolder: Salmela et al., Bioinformatics, 2011
- GRASS: Gritsenko et al., Bioinformatics, 2012
- SCARPA: Donmez et al., Bioinformatics, 2013
- L_RNA_scaffolder: Xue et al., BMC Genomics, 2013
- 手法比較論文: Hunt et al., Genome Biol., 2014

調べると沢山見つかります

ゲノムアセンブル（Linux）

- 前処理 | トリミング | 指定した末
- アセンブル | について (
- アセンブル | ゲノム用 (
- アセンブル | トランスクリプトー
- アセンブル | ゲノム 既知で転写
- マッピング | 備忘録 | について
- マッピング | 備忘録 | basic alig
- マッピング | 備忘録 | splice-aw
- マッピング | 備忘録 | Bisulfite
- マッピング | 備忘録 | (ESTレベ
- マッピング | 基礎(last modified
- マッピング | single-end | ゲノム
- マッピング | single-end | ゲノム

アセンブル | ゲノム用 NEW

2014年6月に調べた結果をリストアップします：

プログラム：

- Phusion: Mullikin and Ning, Genome Res., 2003
- PCAP: Huang et al., Genome Res., 2003
- Newbler: Margulies et al., Nature, 2005
- SSAKE: Warren et al., Bioinformatics, 2007
- Minimus: Sommer et al., BMC Bioinformatics, 2007
- VCAKE: Jeck et al., Bioinformatics, 2007
- SHARCGS: Dohm et al., Genome Res., 2007
- Edena: Hernandez et al., Genome Res., 2008
- Velvet: Zerbino and Birney, Genome Res., 2008
- CABOG: Miller et al., Bioinformatics, 2008
- EULER-SR: Chaisson et al., Genome Res., 2009
- SHORTY: Hossain et al., BMC Bioinformatics,
- QSRA: Bryant et al., BMC Bioinformatics, 2009
- ABySS: Simpson et al., Genome Res., 2009
- Taipan: Schmidt et al., Bioinformatics, 2009
- Forge: Diguistini et al., Genome Biol., 2009
- ALLPATHS-LG: Maccallum et al., Genome Bio
- SOAPdenovo: Li et al., Genome Res., 2010
- SGA: Simpson et al., Genome Res., 2012
- Mapsembler: Peterlongo and Chikhi, BMC Bioinformatics, 2012
- IDBA-UD: Peng et al., Bioinformatics, 2012
- SPAdes: Bankevich et al., J Comput Biol., 2012
- SparseAssembler: Ye et al., BMC Bioinformatics, 2012
- Readjoiner: Gonnella et al., BMC Bioinformatics, 2012
- Fermi: Li H., Bioinformatics, 2012
- TIGER: Wu et al., BMC Bioinformatics, 2012
- CloudBrush: Chang et al., BMC Genomics, 2012
- RACA: Kim et al., PNAS, 2013
- BayesHammer: Nikolenko et al., BMC Genomics, 2013
- SPA: Yang and Yooseph, Nucleic Acids Res., 2013
- SOAPdenovo2: Luo et al., Gigascience, 2012
- JR-Assembler: Chu et al., Proc Natl Acad Sci U S A., 2013
- Platanus: Kajitani et al., Genome Res., 2014

比較的ロングリードの454データ用

Review、ガイドライン、パイプライン系:

- Review: Miller et al., Genomics, 2010
- CG pipeline(パイプライン; 454用): Kislyuk et al., Bioinformatics, 2010
- A5 pipeline(パイプライン): Tritt et al., PLoS One, 2012
- Review: El-Metwally et al., PLoS Comput Biol., 2013
- iMetAMOS(パイプライン): Koren et al., BMC Bioinformatics, 2014

非モデル生物やヘテロ接合度の高い生物種用

微生物など小～中規模ゲノム配列決定用

ゲノムアセンブル（Linux以外）

- 書籍 | トランスクリプトーム解析 | 4.3.2 データの正規化(応用編) (last modified 2014/0
- 書籍 | トランスクリプトーム解析 | 4.3.3 2群間比較 (last modified 2014/04/28)
- 書籍 | トランスクリプトーム解析 | 4.3.4 他の実験デザイン(3群間) (last modified 2014/
- 書籍 | 日本乳酸菌学会誌 | 第1回イントロダクション (last modified 2014/06/18) NEW
- イントロ | 一般 | ランダムに行を抽出 (last modified 2014/10/10)
- イントロ | 一般 | 任意の文字列
- イントロ | 一般 | 任意のキーワ
- イントロ | 一般 | ランダムな塩基
- イントロ | 一般 | 任意の長さの
- イントロ | 一般 | 任意の位置の

ウェブツール:

- DDBJ Read Annotation Pipeline: Nagasaki et al., DNA Res., 2013
- 統合TVのDDBJ Read Annotation Pipeline関連番組の一例

- Galaxy: Goecks et al., Genome Biol., 2010
- P-Galaxy: Nagasaki et al., DNA Res., 2013
- DBCLS Galaxy

- Expression Atlas: Petryszak et al., Nucleic Acids Res., 2014
- ArrayExpress: Rustici et al., Nucleic Acids Res., 2013
- 統合TVのGene Expression Atlas関連番組の一例

- Cufflinks: Trapnell et al., Nat Biotechnol., 2010
- GENSCAN: Burge et al., J Mol Biol., 1997
- 解析目的別留意点やRPKMの説明はシリーズ Useful R 第7巻トラン

- DESeq: Anders and Huber, Genome Biol, 2010
- GSE53960 (320個のFASTQファイル): Yu et al., Nat Commun., 2014
- ReCount: Frazee et al., BMC Bioinformatics, 2011

- RefEx
- ライフサイエンス統合データベースセンター (DBCLS)

ゲノムアセンブリ以外にもマッピングなど一通りの解析が可能らしい

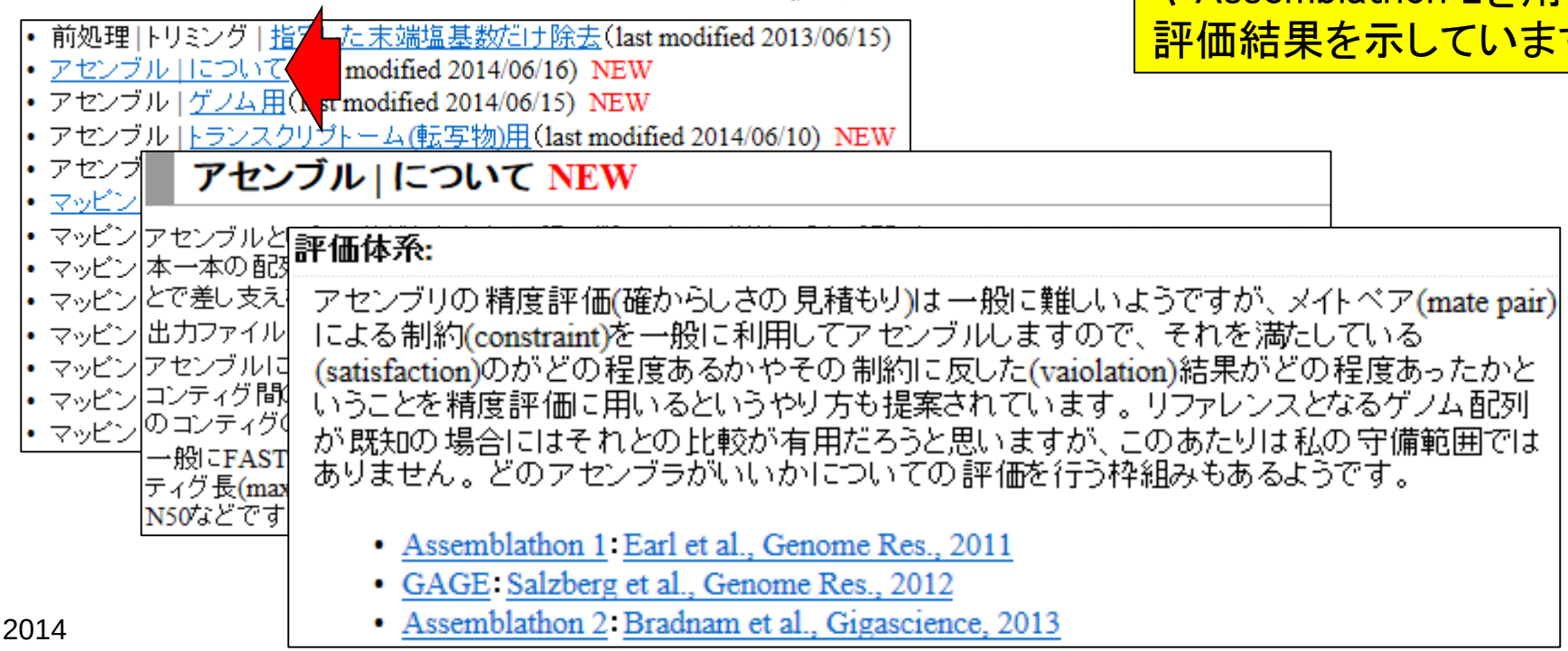
実験系のヒトはわりと使っている人が多いらしい。アセンブルはVelvetだけか?!

Rパッケージはありません

アセンブルの評価関連

- 様々な試み
  - Assemblathon 2 (http://assemblathon.org/)
  - GAGE (http://gage.cbcb.umd.edu/)
    - 自分のゲノムプロジェクトでどの程度のcoverageが必要か？
    - アセンブリ結果がどんな感じになるかの見通し
    - どのソフトウェア（とパラメータ）を使うべきか

最近のアセンブラは大抵GAGEやAssemblathon 2を用いた性能評価結果を示しています



トランスクリプトームアセンブル

- 前処理 | トリミング | 指定した末端塩基数だけ除去 (last modified 2013/06/15)
- アセンブル | について (last modified 2014/06/16) NEW
- アセンブル | ゲノム用 (last modified 2014/06/1… NEW
- アセンブル | トランスクリプトーム(転写物)用 …modified 2014/06/10) NEW
- アセンブル | ゲノム既知で転写物構造推定用 …st modified 2014/05/19) NEW
- マッピング | 備忘録 | について (last modified 2013/10/25)
- マッピング | 備忘録 | basic aligner (last…
- マッピング | 備忘録 | splice-aware align…
- マッピング | 備忘録 | Bisulfite sequenc…
- マッピング | 備忘録 | (ESTレベルの長…
- マッピング | 基礎 (last modified 2013/0…
- マッピング | single-end | ゲノム | basic a…
- マッピング | single-end | ゲノム | basic a…

アセンブル | トランスクリプトーム(転写物)用 NEW

2014年6月に調べた結果をリストアップします:

プログラム:

- Multiple-k: Surget-Groba and Montoya-Burgos, Genome Res., 2010
- Trans-ABySS: Robertson et al., Nature Methods, 2010
- Rnnotator: Martin et al., BMC Genomics, 2010
- Trinity: Grabherr et al., Nature Biotechnol, 2011
- SOAPdenovo-trans: (Luo et al., Gigascience, 2012)
- Oases: (Schulz et al., Bioinformatics, 2012)
- EBARDenovo: (Chu et al., Bioinformatics, 2013)
- IDBA-tran: (Peng et al., Bioinformatics, 2013)

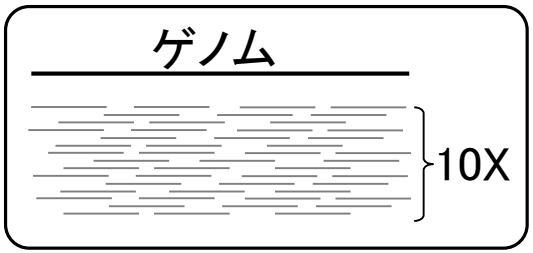
Review、ガイドライン、パイプライン系:

- Review: Martin and Wang, Nat. Rev. Genet., 2011
- ガイドライン: Haznedaroglu et al., BMC Bioinformatics, 2012
- ガイドライン: Yang and Smith, BMC Genomics, 2013
- ガイドライン: Feldmesser et al., BMC Genomics, 2014
- パイプライン: Melicher et al., BMC Genomics, 2014

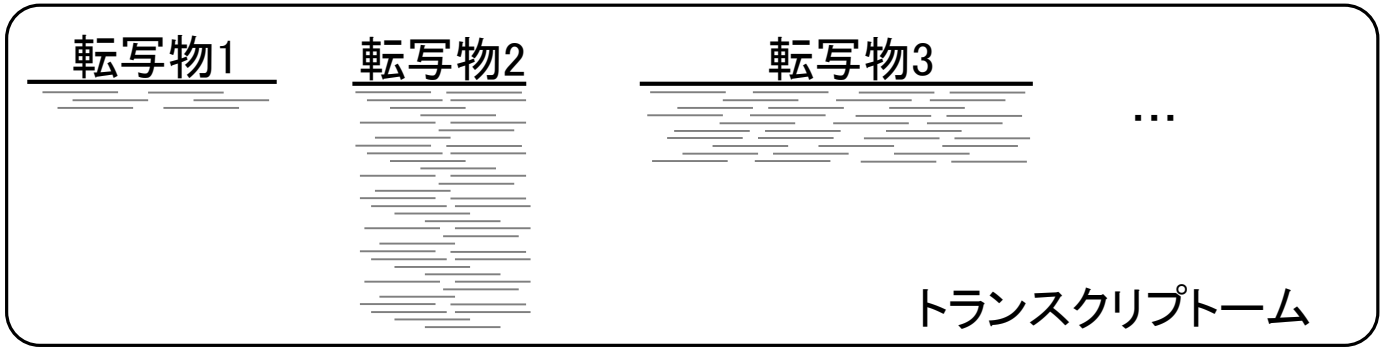
一番よく使われているのはTrinityのようです

ゲノム用とトランスクリプトーム用の違い

- Sequencing depth (coverage)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可能
    - 多くのアセンブラはcoverage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。

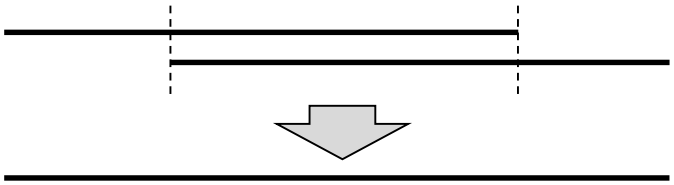


PacBioが普及すればトランスクリプトーム用はもはや必要なし?!



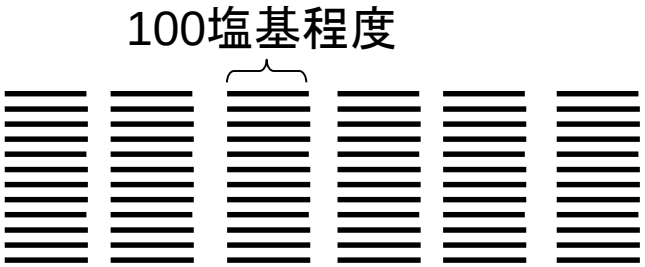
アセンブルの直観的な理解

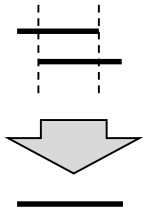
- 旧世代シーケンサー（ABI3730など）：～1,000塩基



一致領域(overlap)大
→ 信頼性高い

- NGS (short-read; Illumina)：～数百塩基





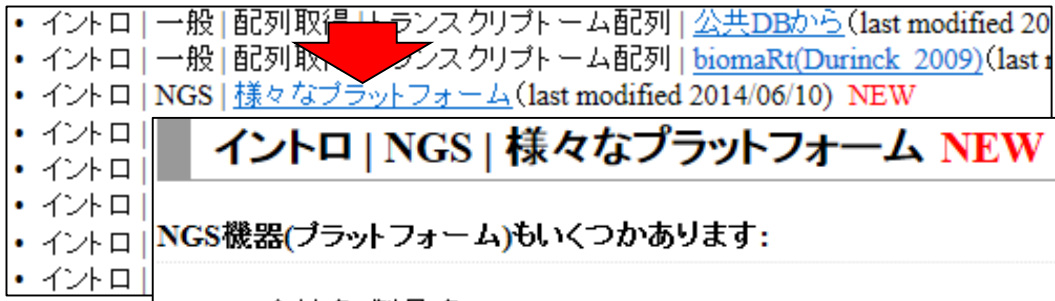
一致領域(overlap)小
→ 信頼性低い

- NGS (long-read; PacBio)：～数千塩基

エラーは多いが転写物配列レベルではアセンブルはほぼ不要なレベル

（今この瞬間を含む）未来予想図

- ゲノム配列決定
  - 小～中規模：Illumina MiSeq、PacBio
  - 大規模：Illumina HiSeq、PacBio
- トランスクリプトーム配列決定
  - PacBio (+ Illumina)
- 発現解析
  - Illumina HiSeq



イントロ | NGS | 様々なプラットフォーム NEW

NGS機器(プラットフォーム)もいくつかあります：

- 会社名：製品名
- Illumina: MiSeq, NextSeq 500, HiSeq 2500, HiSeq X Ten, ...
- Roche: GS FLX+ System, GS Junior+ System
- Life Technologies: SOLiD
- Life Technologies: Ion PGM System
- Pacific Biosciences: PacBio RS II System
- Dover社など「POLONATOR G.007」
- ...

PacBioを用いたトランスクリプトーム配列決定論文は既に存在する

Pacific Biosciences (PacBio)について：

PacBio RS II Systemは最長で20,000bp以上（平均は4,500 bp）読めるようですが配列のqualityが若干(85%程度)劣るようです。しかしエラーの入る場所がランダムなようで多数決ルール(majority rule)でエラー補正がかなりうまくいくらしいです。このロングリードでトランスクリプトーム配列決定（新規アイソフォームの発見）をヒト (Sharon et al., 2013)やニワトリ(Thomas et al., 2014)で行った論文などが出始めています。

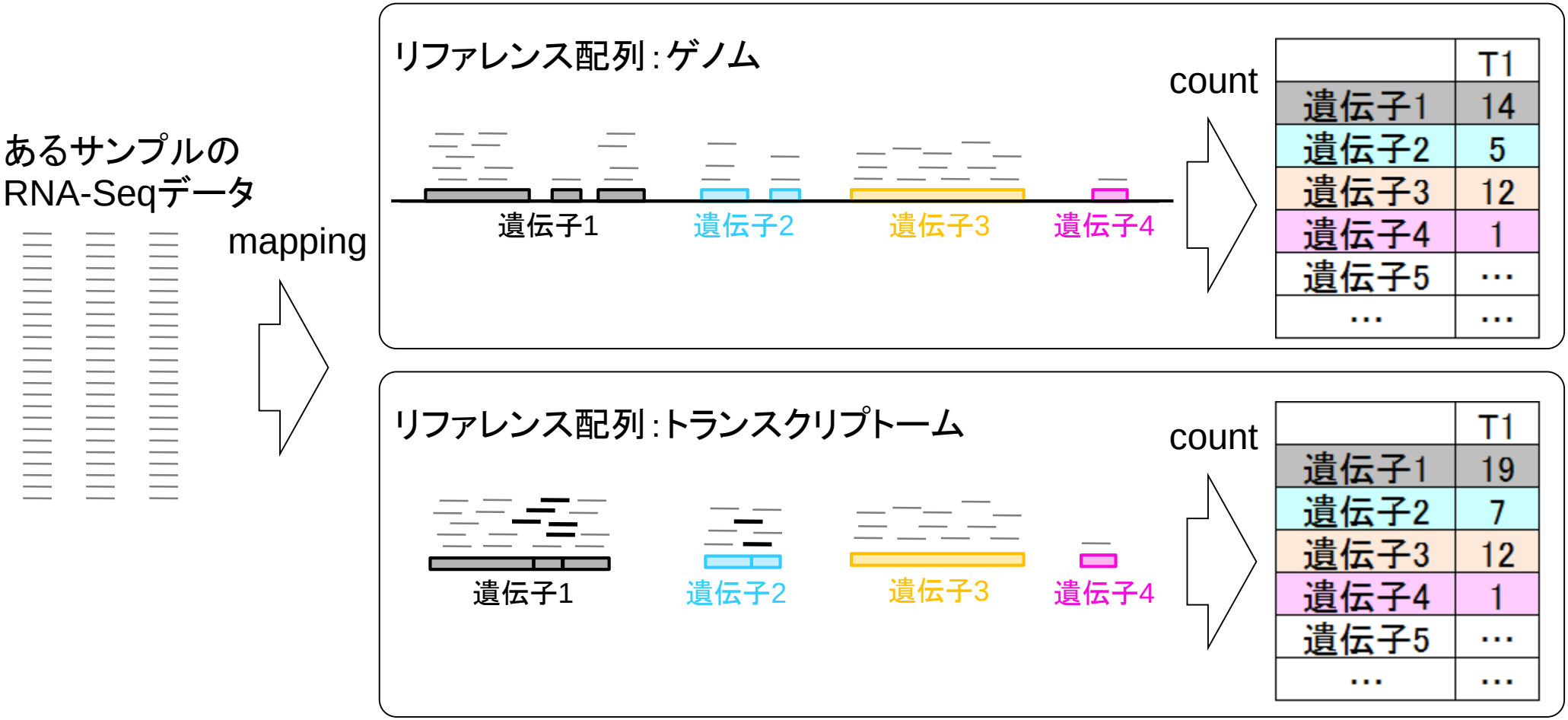
Smart-seq2の実験手順(Picelli et al., Nat Protoc., 2014)なども出ているようですね。葉緑体ゲノムでのアセンブリ性能評価(Ferrarini et al., BMC Genomics, 2013)もなされています。

Contents（第3回）

- アセンブル(Assembly)
  - 2つのアプローチ(two approaches)
    - Comparative approach (reference-based assembly; resequencing)：同一生物種または近縁種のゲノム配列を利用
    - *de novo* approach：過去に配列決定されたものの中に近縁種がない場合
  - アルゴリズム（計算手順）
    - k-mer解析
  - ゲノム用、トランスクリプトーム用、雑感
- **マッピング（QuasRパッケージを利用）**
  - シミュレーションデータを用いたマッピングの基礎
  - リアルデータのマッピング（カイコsmall RNA-seqデータ）
  - 課題
- カウント情報取得

マッピングの基本的なイメージ

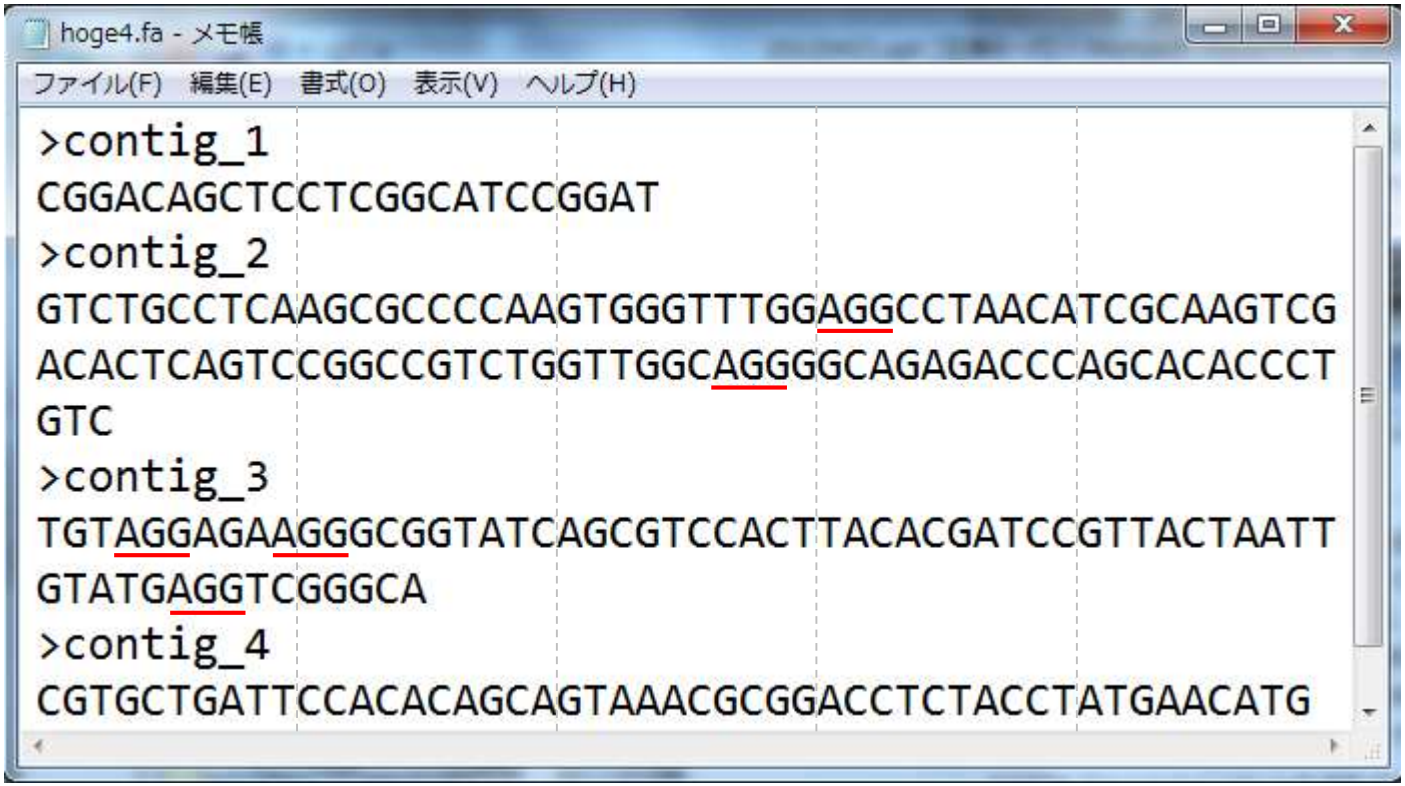
- 基本的なマッピングプログラム（basic aligner; bowtieなど）を用いた場合



マップされたリードをカウントしたデータ（カウントデータ）がその後の数値解析の基礎情報

マッピング ＝ 大量高速文字列検索

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



出力ファイル

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

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

マッピング（準備）

- マップされる側のリファレンス配列：ref_genome.fa

- 過去のお知らせ (last modified 2014/06/13) NEW
- Rのインストール/起動 (last modified 2014/05/14)
- サンプルデータ (last modified 2014/06/17) NEW
- 書籍 |トランスクリプトームについて (last modified 2014/05/12)

サンプルデータ NEW

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

```
out_f <- "ref_genome.fa"                #出力ファイル名を指定してout_fに格納
param_len_ref <- c(48, 160, 100, 123)   #配列長を指定
narabi <- c("A","C","G","T")            #以下の数値指定時にACGTの並びを間違えないようにするために表示(内部的に
param_composition <- c(28, 22, 26, 24) #(A,C,G,Tの並びで)各塩基の存在比率を指定
param_desc <- "chr"                     #FASTA形式ファイルのdescription行に記述する内容
param4 <- 3                             #コピーを作成したい配列番号を指定
param5 <- c(2, 7)                       #コピー先配列の塩基置換したい位置を指定

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

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

#本番(配列生成)
set.seed(1000)                          #おまじない(同じ乱数になるようにするため)
ACGTset <- rep(narabi, param_composition)#narabi中の塩基がparam_compositionで指定した数だけ存在する文字列へ
```

コピペで作成

マッピング（準備）

- マップされる側のリファレンス配列：ref_genome.fa

18. ランダムな塩基配列から生成したリファレンスゲノム配列データ(ref_genome.fa)。48, 160, 100, 123, 100 bpの配列長をもつ、計5つの塩基配列を生成しています。description行は"contig"とい
している。set.seed関数を利用し、chr3の配列と同
す。そのため、実際に指定するのは最初の4つ分の

```
out_f <- "ref_genome.fa"
param_len_ref <- c(48, 160, 100, 123)
narabi <- c("A","C","G","T")
param_composition <- c(28, 22, 26, 24)
param_desc <- "chr"
param4 <- 3
param5 <- c(2, 7)

#必要なパッケージをロード
library(Biostrings)

#塩基置換関数の作成
enkichikan <- function(fa, p) {
  t <- substring(fa, p, p)
  t_c <- chartr("CGAT", "GCTA", t)
  substring(fa, p, p) <- t_c
  return(fa)
}
```

chr3とchr5の違いは、2番目と7番目の塩基のみ。
マッピングプログラムbowtie利用時に、”-m”オプションの違いの把握が可能。

ref_genome.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

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

マッピング（準備）

- マップする側のRNA-seqデータ：sample_RNAseq1.fa

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

サンプルデータ NEW

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

19. 上記リファレンスゲノム配列データ(ref_genome.fa)に対してbasic alignerでマッピング
seqデータ(sample_RNAseq1.fa)とそのgzip圧縮ファイル(sample_RNAseq1.fa.gz)。
リファレンス配列を読み込んで、list_sub3.txtで与えた部分配列を抽出したものです
がわかっているので、basic alignerで許容するミスマッチ数を変えてマップされる or
DNAStringSetオブジェクトを入力として塩基置換を行うDNAString_chartr関数を用い
の塩基にミスマッチを入れています。

```
in_f1 <- "ref_genome.fa"                #入力ファイル名(multi-fa
in_f2 <- "list_sub3.txt"                #入力ファイル名(リストフ
out_f <- "sample_RNAseq1.fa"            #出力ファイル名を指定して
param <- 4                              #塩基置換したい位置を指定

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

#塩基置換関数の作成
DNAString_chartr <- function(fa, p) {   #関数名や引数の作成
  str_list <- as.character(fa)          #文字列に変更
  t <- substring(str_list, p, p)        #置換したい位置の塩基を取
  t_c <- chartr("CGAT", "GCTA", t)      #置換後の塩基を作成
  substring(str_list, p, p) <- t_c      #置換
  fa_r <- DNAStringSet(str_list)        #DNAStringSetオブジェク
  names(fa_r) <- names(fa)              #description部分の情報を
  return(fa_r)                          #置換後のデータを返す
}
```

sample_RNAseq1.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

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

コピペで作成

マッピング（準備）

- マップする側のRNA-seqデータ：sample_RNAseq1.fa

ref_genome.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

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

sample_RNAseq1.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

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

許容するミスマッチ数による違いや、マップされるべき場所が完全に把握できるように、リードのdescription行に記述されている

*QuasR*パッケージを用いてマッピング

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

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

- マッピング | 基礎(last modified 2013/06/19)
- マッピング | single-end | ゲノム | basic aligner(基礎) | QuasR(Lerch_XXX)(last modified 2013/
- マッピング | single-end | ゲノム | basic aligner(応用) | QuasR(Lerch_XXX)(last modified 2013/10/25)
- マッピング | single-end | ゲノム | splice-aware aligner | QuasR(Lerch_XXX)(last modified 2013/10/25)
- マップ後 | について(last modified 2013/06/19)
- マップ後 | 出
- マップ後 | 出
- マップ後 | 出
- マップ後 | 出
- マップ後 | 出
- マップ後 | カウ
- マップ後 | カウ

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

教科書p86-89

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

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

1.サンプルデータ18,19のRNA-seqデータ(sample_RNAseq1.fa)のref_genome.faへのマッピングの場合(mapping_single_genome1.txt):

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

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

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

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

#ファイルに保存(QCレポート用のpdfファイル作成)
out_f <- sub(".bam", "_QC.pdf", out@alignments[,1])#Quqlity Controlレポートのpdfファイル名を作成した結果
qQCReport(out, pdfFilename=out_f)              #QCレポート結果をファイルに保存
out_f                                          #ファイル名を表示してるだけです

#ファイルに保存(BED形式ファイル)
```

複数のRNA-seqサンプルを実行できるようにリストファイルとして与える

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

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

QuasRパッケージを用いてsingle-end RNA-seqデータのリファレンスゲノム配列へのマッピングを行うやり方を示します。basic alignerの一つであるBowtie (Langmead et al., Genome Biol., 2009)を実装した Rbowtieパッケージを内部的に使っています。

Bowtie自体は、複数個所にマップされるリードの取り扱い(uniquely mapped reads or multi-mapped r
容するミスマッチ数を指定する"-v"などの様々なオプションを利用可能ですが、「基礎」のところでは
マッピングのオプションをいくつか変更して挙動を確認したり、複数のRNA-seqファイルを一度にマッ
尚、出力ファイルは、"*.bam", "*_QC.pdf", "*.bed"の3つです。それ以外のファイルは基本無視で
「ファイル」-「ディレクト

1.サンプルデータ18,19

オプションを"-m 1 --b
("chr3_11_45"と"chr3
致ではない(4番目の

```
in_f1 <- "mappi
in_f2 <- "ref_ge
param_mapping <-

#必要なパッケージ
library(QuasR)
library(GenomicF

#本番(マッピング)
time_s <- proc.t
out <- qAlign(in
time_e <- proc.t
time_e - time_s
out
alignmentStats(o

#ファイルに保存(Q
out_f <- sub(".b
qQCReport(out, p
out_f

#ファイルに保存(B
```

R Console

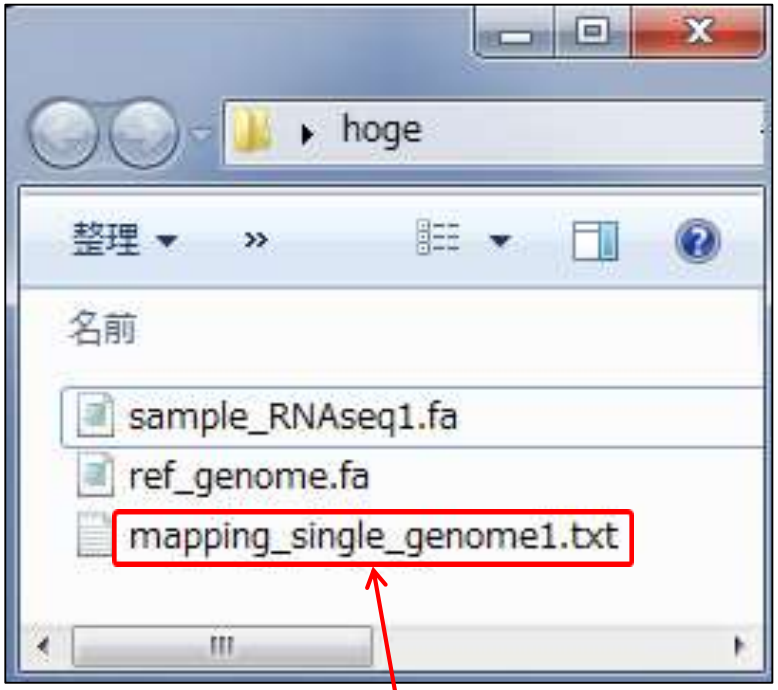
```
> alignmentStats(out)
             seqlength mapped unmapped
namae:genome       531      5        3
>
> #ファイルに保存(QCレポート用のpdfファイル作成)
> out_f <- sub(".bam", "_QC.pdf", out@alignments[,1])#Quqlity Cont$
> qQCReport(out, pdfFilename=out_f)       #QCレポート結果をファイル$
collecting quality control data
creating QC plots
> out_f                                   #ファイル名を表示してるだ$
[1] "C:/Users/kadota/Desktop\\sample_RNAseq1_205c310435be_QC.pdf"
>
> #ファイルに保存(BED形式ファイル)
> tmpfname <- out@alignments[,1]          #ファイル名(in_f1の1列目$
> for(i in 1:length(tmpfname)){           #サンプル数(ファイル数)分$
+   hoge <- readGAlignments(tmpfname[i]) #BAM形式ファイルを読み込$
+   hoge <- as.data.frame(hoge)           #データフレーム形式に変換
+   tmp <- hoge[, c("seqnames","start","end")]#必要な列の情報のみ$
+   out_f <- sub(".bam", ".bed", tmpfname[i])#BED形式ファイル名を$
+   out_f                                 #ファイル名を表示してるだ$
+   write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names$
+ }
>
```

入力ファイル中の8リードのうち、マップされたのが5リード、マップされなかったのが3リード。R console画面でなく、QCレポートPDFファイル中にも記述あり。

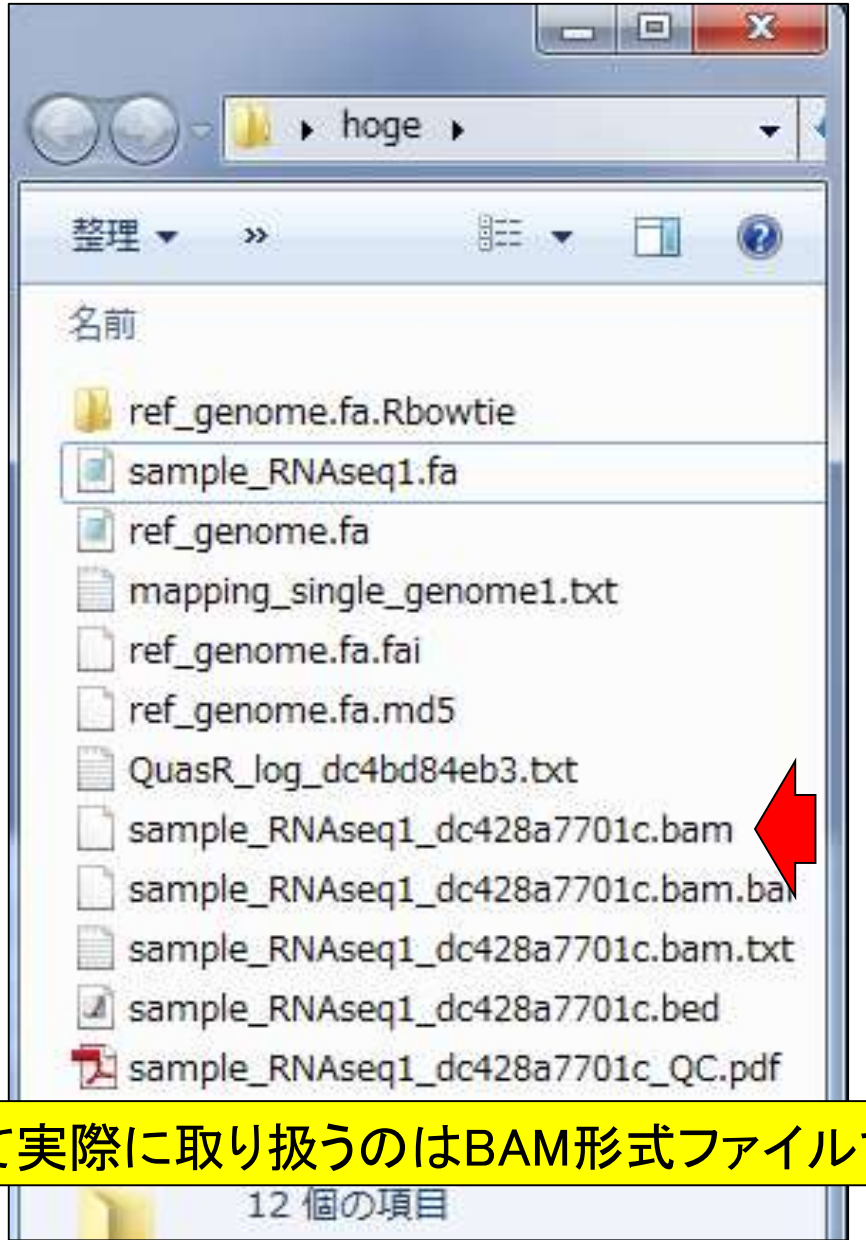
教科書p86-89

QuasRパッケージを用いてマッピング

教科書p86-89



実行後



FileName	SampleName
sample_RNAseq1.fa	namae

出力ファイルとして実際に取り扱うのはBAM形式ファイルです

教科書p86-89

マッピング結果の出力ファイル形式

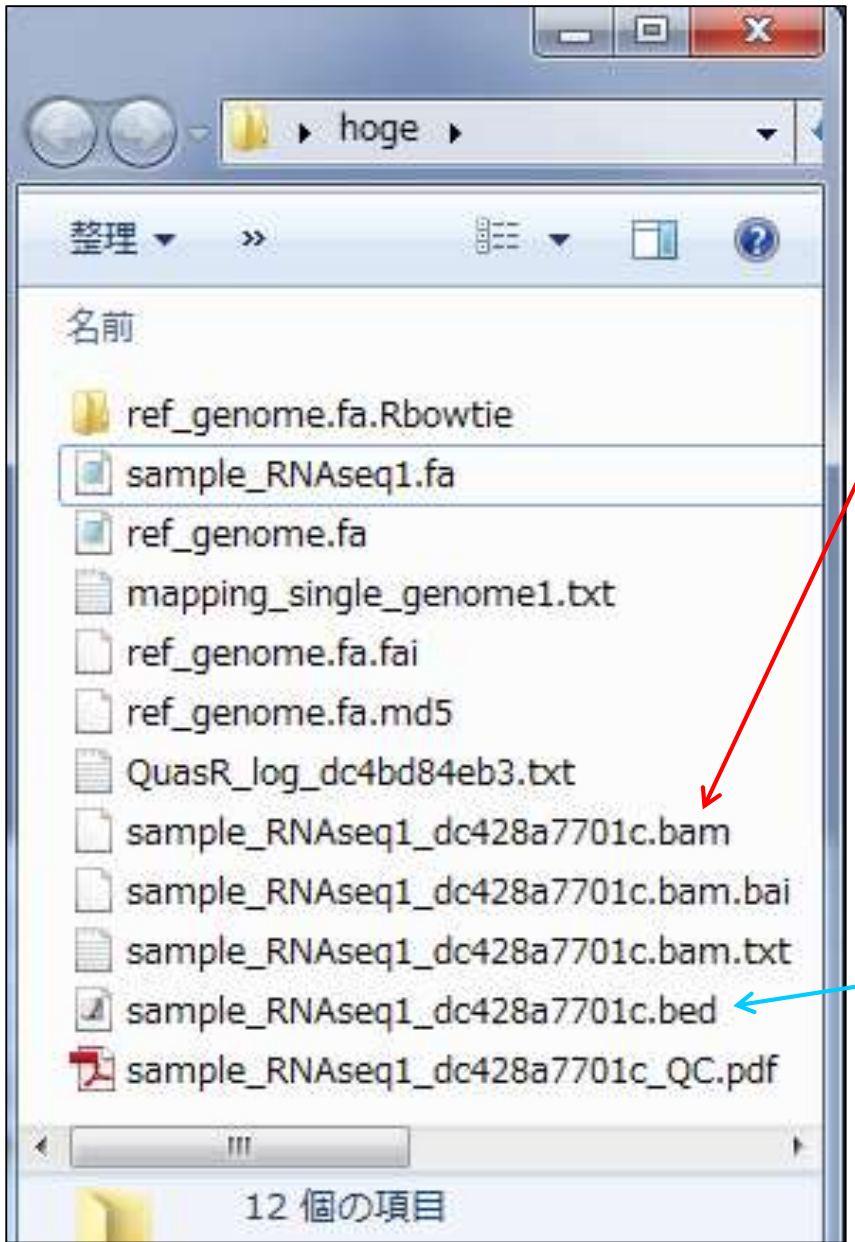
- ゲノム上のどの位置にどのリードがマッピングされたか（トランスクリプトームの場合どの転写物配列上のどの位置にどのリードがマッピングされたか）を表すファイル形式は複数あります。
  - SAM (Sequence Alignment/Map) format
    - SAMtools (Li et al., *Bioinformatics*, **25**: 2078–2079, 2009)
  - **BAM** (Binary Alignment/Map) format
    - SAMtools (Li et al., *Bioinformatics*, **25**: 2078–2079,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

…

実用上はBAM形式、視覚上はBED形式

マッピング結果の出力ファイル形式

教科書p86-89



BAM形式ファイル

```
 ‹ ·····ÿ ·BC ·P }ÑNÀ0 Ë #á  î·µë0¦WAFt:nø7MEØ: °
unø,¾¯g $CM¼hÞÞß9▯{GÓÎW á«=   Hf4 øJ7E,yw =hQ@u5? öUï Y /Ñ9
#
C?Ý0³AOw▯-y. 7'á§)Ô Oû eóW5OÆ¤1sgÎ|:ùÀ,dÏ^ ûÚ  ±
pÙÙQ ¾çÐif 5QÓaõVÒ¥‹7+-V¼ 97°Ø8¥f£xEÊ÷
ÿÇë8/w|  £¿`u Ôsi®/y-Ô²¥¯ A=z õRÕ▯¿fT®êc¾Hâ {äyÉBânSÜ²?à Ù
  ªÚïu/··uø ZcÏÖ ·¶Æ ï- 4oË à   ·· ‹ ·····ÿ ·BC ·  Ò=oÃ
  à³Õ) R  cI<0ÆuHÕ-¶o ;W] Òiký
bè¿Nhì( c;‡ Ò=z9jhj öt÷F£p.ÂÞµõ▯·ý®S¦f¥`  $Ò& $×
È Ï˜e▯]ä%Tø)×¯Jø        ]££Ü->Ôd
Ê!:ì'ä8·ñhÝ¢³ÔQ °6jyP-³¾PÎ(a»¢ÆýQ³STD¤öÏí é  §
õÓÝhÜõÒ¢¦ ôë ¦·  I _éBÓ^óÇ!bUFÃ eV¯p¥P6(Yp¶
g='Ýj&W èÆO-<Ô ù  d,·· ‹ ·····ÿ ·BC · · ·········
```

BED形式ファイル

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

BEDの最小限の情報は、リードIDを含まない

マッピングオプションと結果の解釈

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

ref_genome.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

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

TGGG
GAGGAG
GGGCTG
GGTTCA

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

sample_RNAseq1.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

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

マップされなかったのは、
計8リード中3リード

マッピングオプションと結果の解釈

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

ref_genome.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

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

sample_RNAseq1.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

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

完全一致でも複数個所にマップされるために落とされた2リード

マッピングオプションと結果の解釈

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

ref_genome.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

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

sample_RNAseq1.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

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

1塩基ミスマッチのため落とされたリード

実データのマッピングを行う

教科書p89-90

- 前処理 | トリミング | アダプター配列除去(基礎) | girafe(Toedling_2010)(last modified 2014/06/11) NEW
- 前処理 | トリミング | アダプター配列除去(基礎) | ShortRead(Morgan_2009)(last modified 2014/06/11) NEW
- 前処理 | トリミング | アダプター配列除去(応用) | QuasR(Lerch_20XX)(last modified 2014/06/11) NEW

前処理 | トリミング | アダプター配列除去(基礎) | ShortRead(Morgan_2009) NEW

4. FASTQ形式ファイル(SRR609266.fastq.gz)の場合:

small RNA-seqデータ(400Mb弱、11928428リード)です。圧縮ファイルもreadDNAStringSet関数で通常手順で読み込めます。原著論文(Nie et al., BMC Genomics, 2013)中の記述からGSE41841を頼りに、SRP016842にたどりつき、イントロ | NGS | 配列取得 | FASTQ or SRALite | SRAdb(Zhu_2013)の7を実行して得られたものが入力ファイルです。原著論文中では、アダプター配列やクオリティの低いリードを除去したのち、ゲノムにマッピングしたと書いてあります。アダプター配列情報はどこにも書かれていませんでしたが、Table S2中のアダプター配列除去後の最も短いリードが18 nt (例: "GCAGTCGTGGCCGAGCGG")であり、「この18 nt」と「この配列を含む生リード配列の差分」がアダプター配列ということになります。詳細な情報は書かれていませんでしたが、おそらくアダプター配列は"TGGAATTCTCGGGTGCCAAGGAACTCCAGTC..."という感じだろうと推測して、許容するミスマッチ数が1という条件でアダプタ配列除去を行っています。最後のwriteFastq関数実行時にcompress=Tとしてgzip圧縮FASTQ形式で保存しています。

```
in_f <- "SRR609266.fastq.gz"                #入力ファイル名を指定してin_fに格納(RNA-seqファイル)
out_f <- "hoge4.fastq.gz"                   #出力ファイル名を指定してout_fに格納
param_adapter <- "TGGAATTCTCGGGTGCCAAGGAACTCCAGTC"#アダプター配列を指定
param_mismatch <- 1                         #許容するミスマッチ数を指定

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

#入力ファイルの読み込み
fastq <- readFastq(in_f)                    #in

#本番
hoge1 <- trimLRPatterns(Rpattern=param_adapter,#アダプター配列除去を行った結果をhoge1に格納
            subject=sread(fastq),           #アダプター配列除去を行った結果をhoge1に格納
            max.Rmismatch=rep(param_mismatch, nchar(param_adapter)))#アダプター配列除去を行った結果をh
hoge2 <- BStringSet(quality(quality(fastq)), start=1, end=width(hoge1))#quality(fastq)オブジェクトを
fastq <- ShortReadQ(hoge1, hoge2, id(fastq))#ReadFastQ関数を用いてReadFastQというクラスオブジェクトを-
sread(fastq)                                #配列情報を表示
```

目的:カイコゲノム配列にsmall RNA-seqリードをマップ。アダプター配列除去前後でのマップ率の違いを考察(←これが課題)。hoge – SRP016842フォルダ中に2つともあります。

実データのマッピングを行う

教科書p89-90

複数のRNA-seqサンプルを実行できるようにリストファイルとして与える

- マッピング | single-end | ゲノム | basic aligner(基礎) | QuasR(Lerch_XXX)(last modified 2013/10/25)
- マッピング | single-end | ゲノム | basic aligner(応用) | QuasR(Lerch_XXX)(last modified 2013/10/25)
- マッピング | single-end | ゲノム | splice-aware aligner | QuasR(Lerch_XXX)(last modified 2013/10/25)

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

QuasRパッケージを用いてsingle-end RNA-seqデータのリファレンスゲノム配列へのマッピングを行うやり方を示します。basic alignerの一つ

6. 2つのgzip圧縮FASTQ形式ファイル(SRR609266.fastq.gzとhoge4.fastq.gz)のカイコゲノム(integretedseq.fa)へのマッピングの場合(mapping_single_genome8.txt):

small RNA-seqデータ(400Mb弱; 11928428リード; Nie et al., BMC Genomics, 2013)です。イントロ | NGS | 配列取得 | FASTQ or SRALite | SRAdb(Zhu 2013)の7を実行して得られたものがSRR609266.fastq.gzです。また、前処理 | トリミング | アダプター配列除去(基礎) | ShortRead(Morgan 2009)の4を実行して得られたものがhoge4.fastq.gzです。カイコゲノム配列は、農業生物資源研究所(NIAS)が提供しているカイコゲノム配列のウェブページからIntegrated sequences (integretedseq.txt.gz) をダウンロードし、解凍します。解凍後のファイル名は"integretedseq.txt"となりますが、拡張子を".txt"から".fa"に変更して、"integretedseq.fa"としたものを使用しています。30分強かかります。

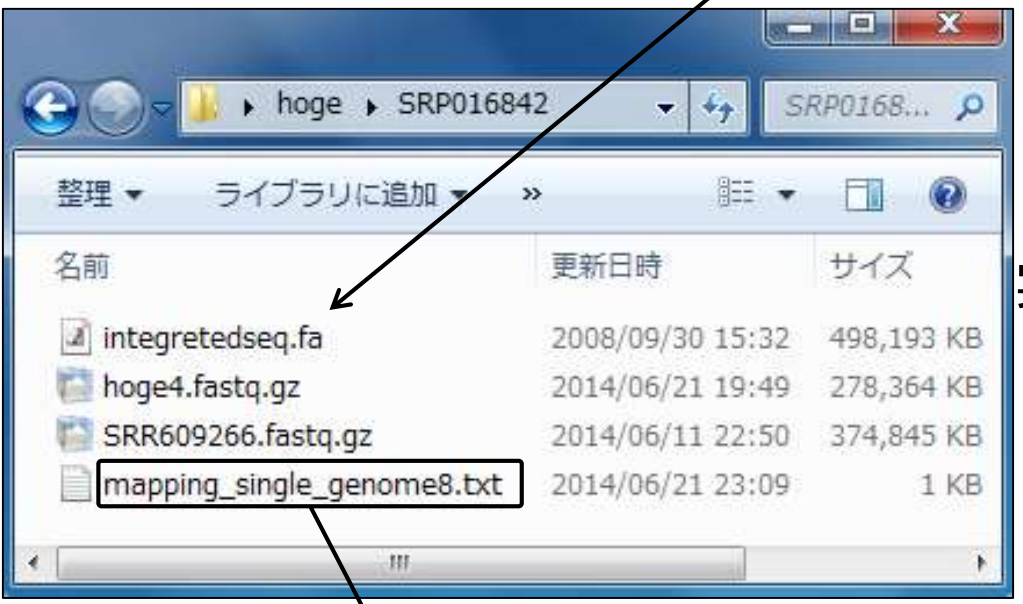
```
in_f1 <- "mapping_single_genome8.txt"  #入力ファイル名を指定してin_f1に格納(RNA-seqファイル)
in_f2 <- "integretedseq.fa"            #入力ファイル名を指定してin_f2に格納(リファレンス配列)
param_mapping <- "-m 1 --best --strata -v 2"#マッピング時のオプションを指定

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

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

許容するミスマッチ数は2個（”-v 2”）、1か所にマップされるリードのみ出力（”-m 1”）

実データのマッピングを行う

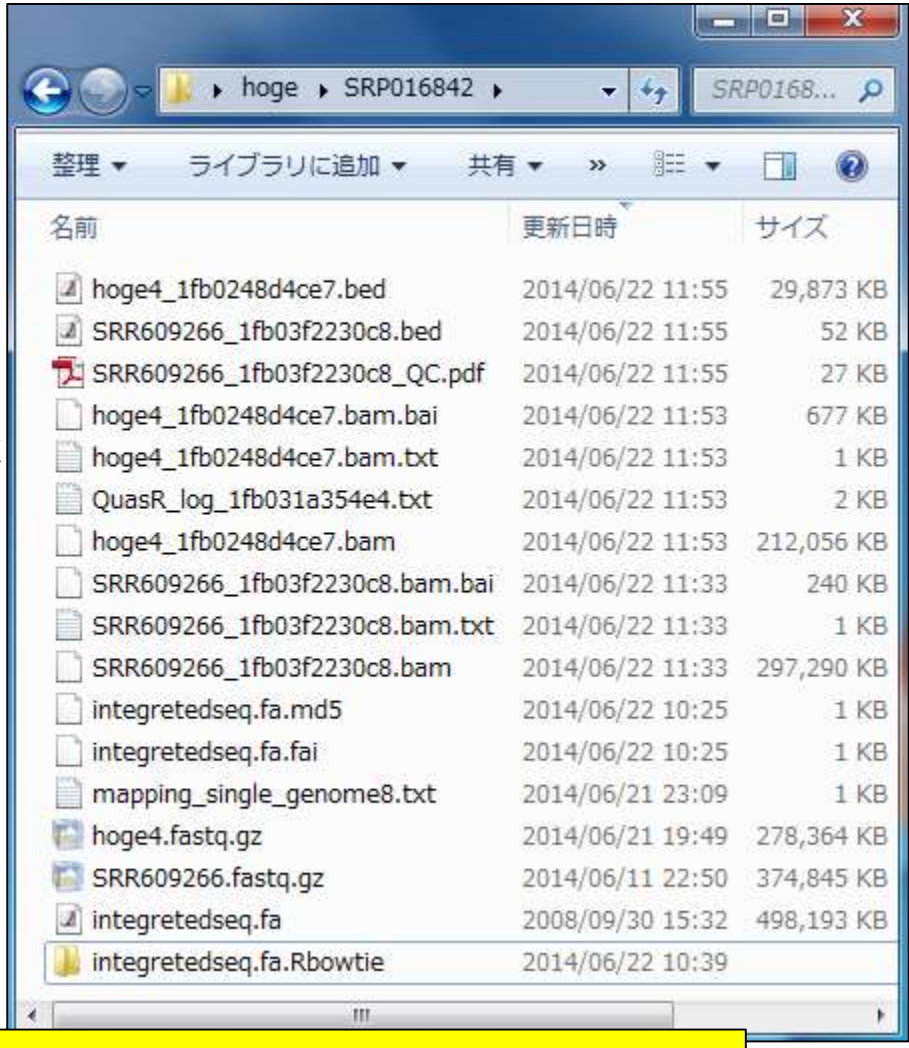
教科書p89-90

カイコゲノムファイル



FileName	SampleName
SRR609266.fastq.gz	pre_adapter_trim
hoge4.fastq.gz	post_adapter_trim

実行後



ファイルサイズ削減のため、配布したhoge – SRP016842
フォルダ中のファイル群はいくつか除いています

実データのマッピング結果

教科書p89-90

```
R Console

> time_e - time_s                          #計算時間を表示(一番右側の数$
   ユーザ    システム          経過
     37.62       13.58     5329.58
> out                                      #マッピングに用いたパラメー$
Project: qProject
Options    : maxHits          : 1
             paired           : no
             splicedAlignment: FALSE
             bisulfite        : no
             snpFile          : none
Aligner    : Rbowtie v1.4.0 (parameters: -m 1 --best --strata -v 2)
Genome     : C:/Users/kadota/Desktop/hoge/SRP016842/integretedseq.fa $

Reads      : 2 files, 2 samples (fastq format):
  1. SRR609266.fastq.gz  pre_adapter_trim  (phred33)
  2. hoge4.fastq.gz      post_adapter_trim (phred33)

Genome alignments: directory: same as reads
  1. SRR609266_1fb03f2230c8.bam
  2. hoge4_1fb0248d4ce7.bam

Aux. alignments: none
```

マッピングに要した時間は5329.58秒(約90分)

マッピングに用いたプログラムやオプション情報

入力と出力ファイル情報

実データのマッピング結果

教科書p89-90

R Console

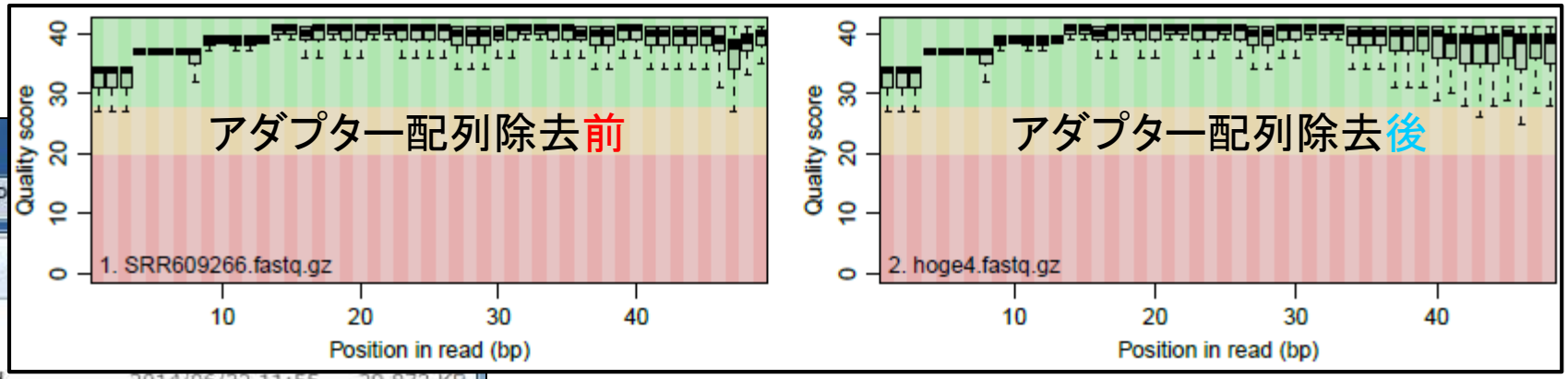
```
> alignmentStats(out)                          #マッピング結果(alignment statist$
                          seqlength  mapped unmapped
pre_adapter_trim:genome   502962917    2257 11926171
post_adapter_trim:genome 502962917 1308126 10620302
>
> #ファイルに保存(QCレポート用のpdfファイル作成)
> out_f <- sub(".bam", "_QC.pdf", out@alignments[,1])#Quqlity Controlレポ$
> qQCReport(out, pdfFilename=out_f)            #QCレポート結果をファイルに保存
collecting quality control data
creating QC plots
> out_f                                        #ファイル名を表示してるだけです
[1] "C:/Users/kadota/Desktop/hoge/SRP016842\\SRR609266_1fb03f2230c8_QC.pdf"
[2] "C:/Users/kadota/Desktop/hoge/SRP016842\\hoge4_1fb0248d4ce7_QC.pdf"
>
> #ファイルに保存(BED形式ファイル)
> tmpfname <- out@alignments[,1]               #ファイル名(in_f1の1列目に相当)を$
> for(i in 1:length(tmpfname)){                #サンプル数(ファイル数)分だけルー$
+   hoge <- readGAlignments(tmpfname[i]) #BAM形式ファイルを読み込んだ結果$
+   hoge <- as.data.frame(hoge)                #データフレーム形式に変換
+   tmp <- hoge[, c("seqnames","start","end")]#必要な列の情報のみ抽出した$
+   out_f <- sub(".bam", ".bed", tmpfname[i])#BED形式ファイル名を作成した$
+   out_f                                      #ファイル名を表示してるだけです
+   write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.$
+ }
```

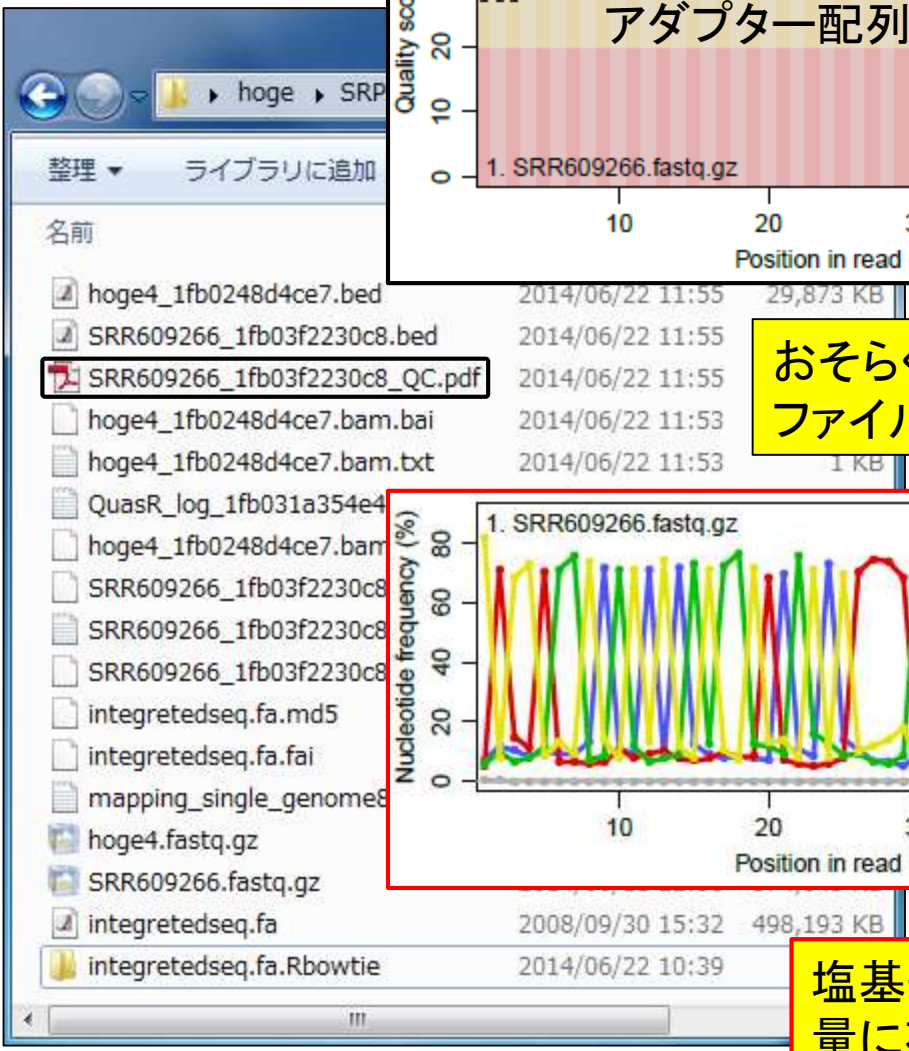
アダプター配列除去前後のマッピング結果

QCレポートファイルは実際には1つだけ作成される

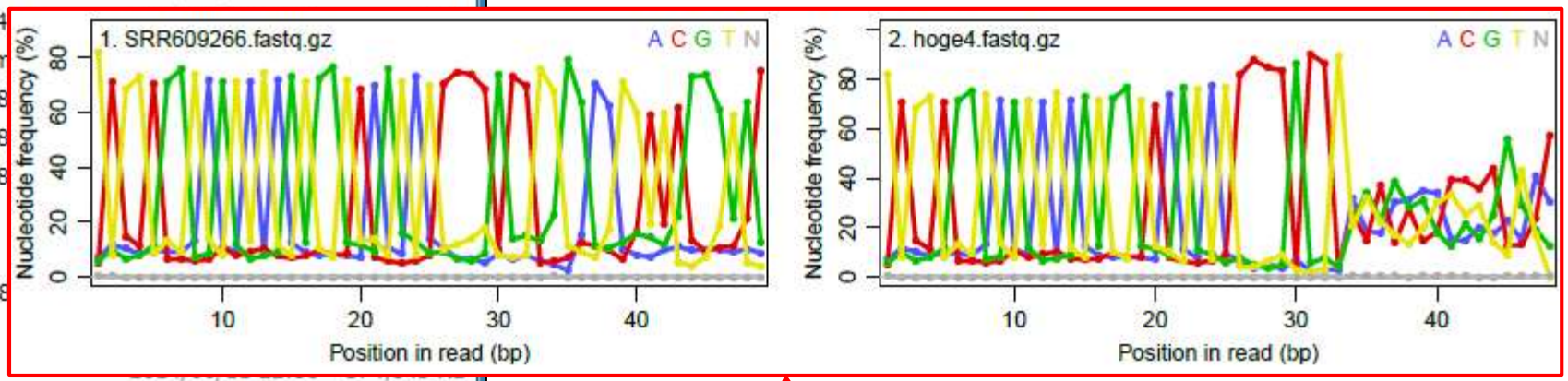
実データのマッピング結果

教科書p89-90

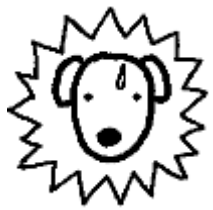


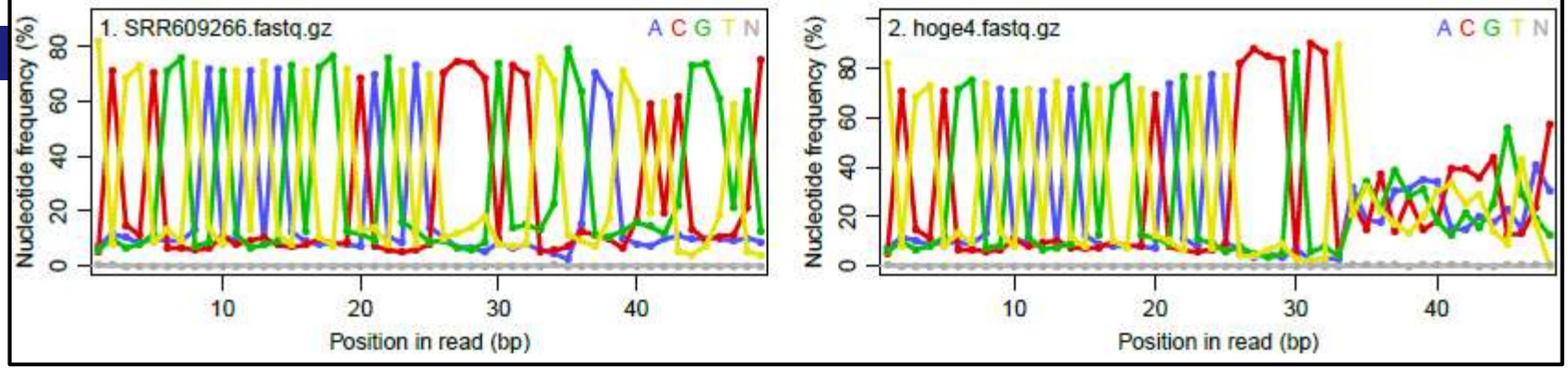


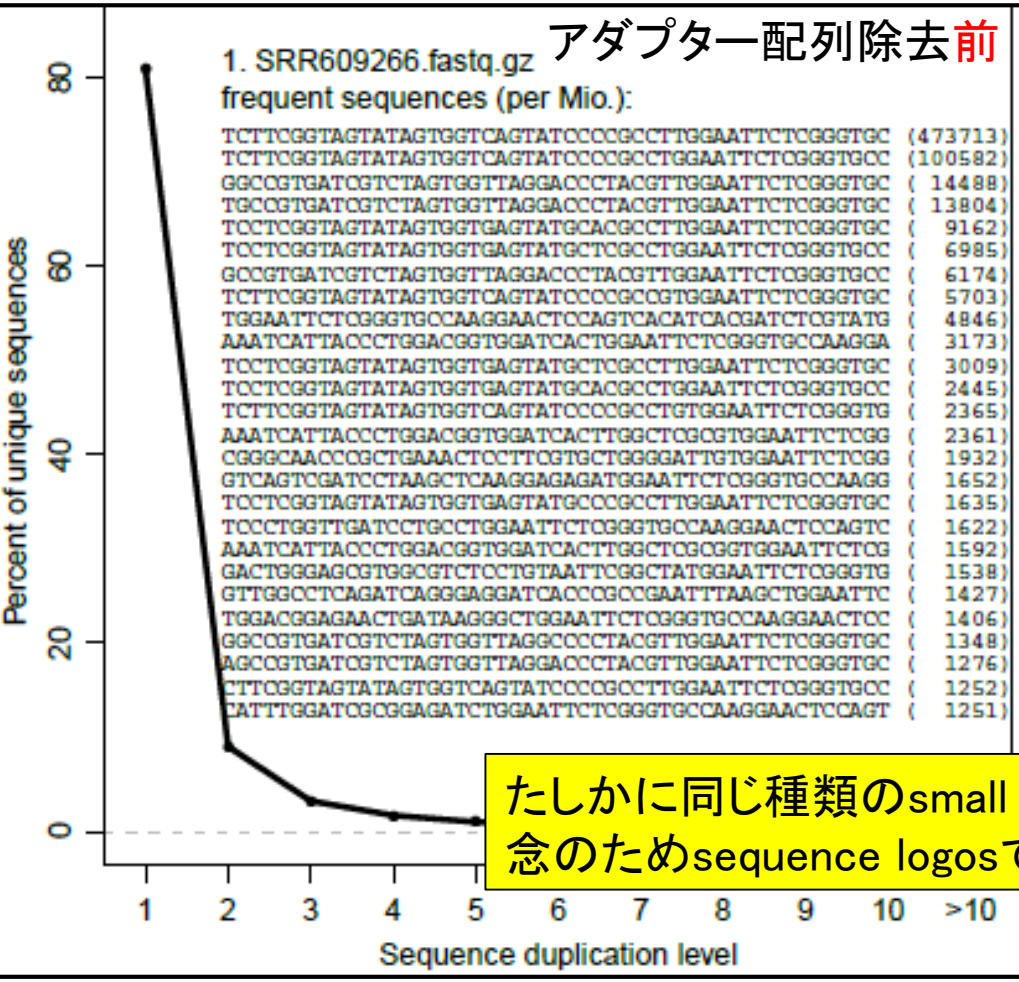
おそらくどのマッピングプログラムもこのようなサマリーレポートファイルを出力する。上：クオリティ分布、下：塩基組成



塩基組成があたかも同じ種類のものが大量に存在しているように見えるがバグか?!




1. SRR609266.fastq.gz
A C G T N
Nucleotide frequency (%)
0
20
40
60
80
10
20
30
40
Position in read (bp)
2. hoge4.fastq.gz
A C G T N
Nucleotide frequency (%)
0
20
40
60
80
10
20
30
40
Position in read (bp)


アダプター配列除去前
1. SRR609266.fastq.gz
frequent sequences (per Mio.):
TCTTCGGTAGTATAGTGGTCAGTATCCCCGCCTTGGAATTCTCGGGTGC (473713)
TCTTCGGTAGTATAGTGGTCAGTATCCCCGCCTGGAATTCTCGGGTGCC (100582)
GGCCGTGATCGTCTAGTGGTTAGGACCCTACGTTGGAATTCTCGGGTGC (14488)
TGCCGTGATCGTCTAGTGGTTAGGACCCTACGTTGGAATTCTCGGGTGC (13804)
TCCTCGGTAGTATAGTGGTGAGTATGCACGCCTTGGAATTCTCGGGTGC (9162)
TCCTCGGTAGTATAGTGGTGAGTATGCTCGCCTGGAATTCTCGGGTGCC (6985)
GCCGTGATCGTCTAGTGGTTAGGACCCTACGTTGGAATTCTCGGGTGCC (6174)
TCTTCGGTAGTATAGTGGTCAGTATCCCCGCCGTGGAATTCTCGGGTGC (5703)
TGGAATTCTCGGGTGCCAAGGAACTCCAGTCACATCACGATCTCGTATG (4846)
AAATCATTACCCTGGACGGTGGATCACTGGAATTCTCGGGTGCCAAGGA (3173)
TCCTCGGTAGTATAGTGGTGAGTATGCTCGCCTTGGAATTCTCGGGTGC (3009)
TCCTCGGTAGTATAGTGGTGAGTATGCACGCCTGGAATTCTCGGGTGCC (2445)
TCTTCGGTAGTATAGTGGTCAGTATCCCCGCCTGTGGAATTCTCGGGTG (2365)
AAATCATTACCCTGGACGGTGGATCACTTGGCTCGCGTGGAATTCTCGG (2361)
CGGGCAACCCGCTGAAACTCCTTCGTGCTGGGGATTGTGGAATTCTCGG (1932)
GTCAGTCGATCCTAAGCTCAAGGAGAGATGGAATTCTCGGGTGCCAAGG (1652)
TCCTCGGTAGTATAGTGGTGAGTATGCCCGCCTTGGAATTCTCGGGTGC (1635)
TCCCTGGTTGATCCTGCCTGGAATTCTCGGGTGCCAAGGAACTCCAGTC (1622)
AAATCATTACCCTGGACGGTGGATCACTTGGCTCGCGGTGGAATTCTCG (1592)
GACTGGGAGCGTGGCGTCTCCTGTAATTCGGCTATGGAATTCTCGGGTG (1538)
GTTGGCCTCAGATCAGGGAGGATCACCCGCCGAATTTAAGCTGGAATTC (1427)
TGGACGGAGAACTGATAAGGGCTGGAATTCTCGGGTGCCAAGGAACTCC (1406)
GGCCGTGATCGTCTAGTGGTTAGGCCCCTACGTTGGAATTCTCGGGTGC (1348)
AGCCGTGATCGTCTAGTGGTTAGGACCCTACGTTGGAATTCTCGGGTGC (1276)
CTTCGGTAGTATAGTGGTCAGTATCCCCGCCTTGGAATTCTCGGGTGCC (1252)
CATTTGGATCGCGGAGATCTGGAATTCTCGGGTGCCAAGGAACTCCAGT (1251)
Percent of unique sequences
0
20
40
60
80
1
2
3
4
5
6
7
8
9
10
>10
Sequence duplication level
アダプター配列除去後
2. hoge4.fastq.gz
frequent sequences (per Mio.):
TCTTCGGTAGTATAGTGGTCAGTATCCCCGCCT (477603)
TCTTCGGTAGTATAGTGGTCAGTATCCCCGCC (102080)
GGCCGTGATCGTCTAGTGGTTAGGACCCTACGT (14594)
TGCCGTGATCGTCTAGTGGTTAGGACCCTACGT (13912)
TCCTCGGTAGTATAGTGGTGAGTATGCACGCCT (9344)
TCCTCGGTAGTATAGTGGTGAGTATGCTCGCC (6968)
GCCGTGATCGTCTAGTGGTTAGGACCCTACGT (6370)
TCTTCGGTAGTATAGTGGTCAGTATCCCCGCCG (5864)
TGGAATTCTCGGGTGCCAAGGAACTCCAGTCACATCACGATCTCGTA (4857)
AAATCATTACCCTGGACGGTGGATCAC (3176)
TCCTCGGTAGTATAGTGGTGAGTATGCTCGCCT (3065)
TCTTCGGTAGTATAGTGGTCAGTATCCCCGCCTG (2475)
TCCTCGGTAGTATAGTGGTGAGTATGCACGCC (2460)
AAATCATTACCCTGGACGGTGGATCACTTGGCTCGCG (2354)
CGGGCAACCCGCTGAAACTCCTTCGTGCTGGGGATTG (1975)
TCCTCGGTAGTATAGTGGTGAGTATGCCCGCCT (1745)
TCCCTGGTTGATCCTGCC (1686)
GTCAGTCGATCCTAAGCTCAAGGAGAGA (1670)
AAATCATTACCCTGGACGGTGGATCACTTGGCTCGCGG (1645)
GACTGGGAGCGTGGCGTCTCCTGTAATTCGGCTA (1553)
GTTGGCCTCAGATCAGGGAGGATCACCCGCCGAATTTAAGC (1470)
TGGACGGAGAACTGATAAGGGC (1451)
CTTCGGTAGTATAGTGGTCAGTATCCCCGCCT (1367)
GGCCGTGATCGTCTAGTGGTTAGGCCCCTACGT (1355)
TGCCGTGATCGTCTAGTGGTTAGGCCCCTACGT (1278)
TGAGATCATTGTGAAAGCTAAT (1274)
CATTTGGATCGCGGAGATC (1268)
CCTACGT (1237)
ACCACTGATTGCAT (1144)
Percent of unique sequences
0
20
40
60
80
1
2
3
4
5
6
7
8
9
10
>10
Sequence duplication level
たしかに同じ種類のsmall RNA配列が沢山存在してそう。
念のためsequence logosで確認してみる。

- 解析 | 一般 | GC含量 (GC contents)(last modified 2014/05/01)
- 解析 | 一般 | Sequence logos(Schneider_1990)(last modified 2014/06/21) NEW
- 解析 | 一般 | 上流配列解析 | LDSS(Yamamoto ...07)(last modified 2012/07/17)
- 解析 | 一般 | 上流配列解析 | Relative Appearance Ratio(Yamamoto_2011)(last modified ...

• 解析 | 一般 | Sequence logos(Schneider_1990)

解析 | 一般 | Sequence logos(Schneider_1990) NEW

seqLogoパッケージを用いてsequence logos (Schneider and Stephens, 1990)を実行するやり方を示します。ここでは

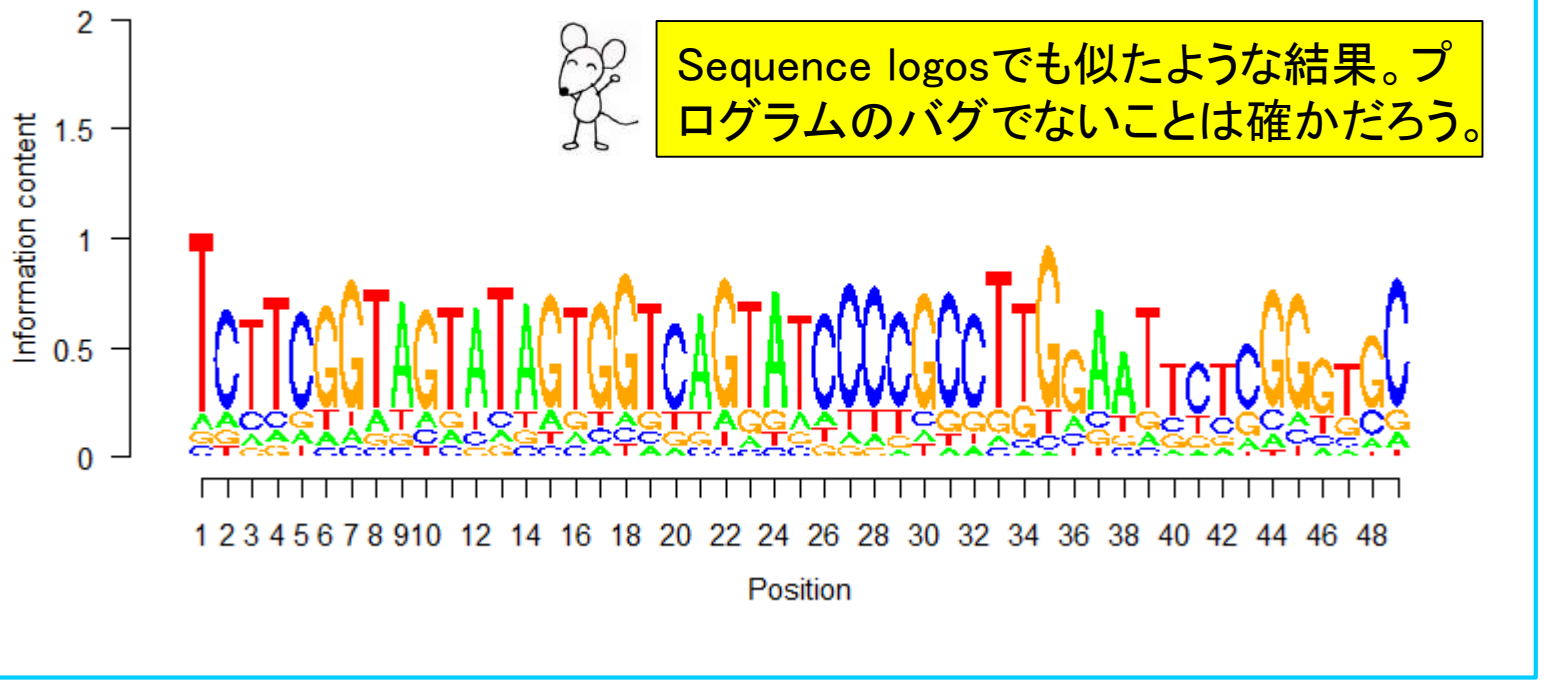
mu
があ
「ファイ

8. FASTQ形式ファイル(SRR609266.fastq.gz)の場合:

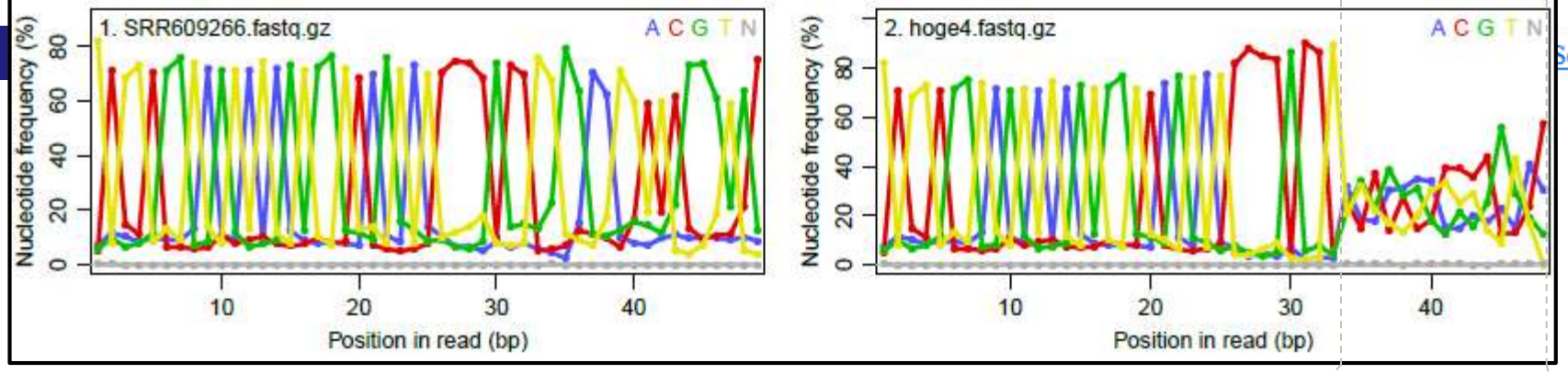
small RNA-seqデータ(400Mb弱、11,928,428リード)です。圧縮ファイルもreadDNAStringSet関数で通常手順で読み込めます。原著論文(Nie et al., BMC Genomics, 2013)中の記述からGSE41841を頼りに、SRP016842にたどりつき、イントロ | NGS | 配列取得 | FASTQ or SRALite | SRAdb(Zhu_2013)の7を実行して得られたものが入力ファイルです。

1. 入力

```
in_f <- "SRR609266.fastq.gz"          #入力ファイル名を指定してin_fに格納
out_f <- "hoge8.png"                  #出力ファイル名を指定してout_fに格納
param_fig <- c(800, 370)              #ファイル出力時の横幅と縦幅を指定(単位はピクセル)

#必要なパッケージをロード
library(B
library(s

#入力ファ
fasta <-

#本番(seq
hoge <- c
out <- ma

#ファイル
png(out_f
seqLogo(c
dev.off()
```

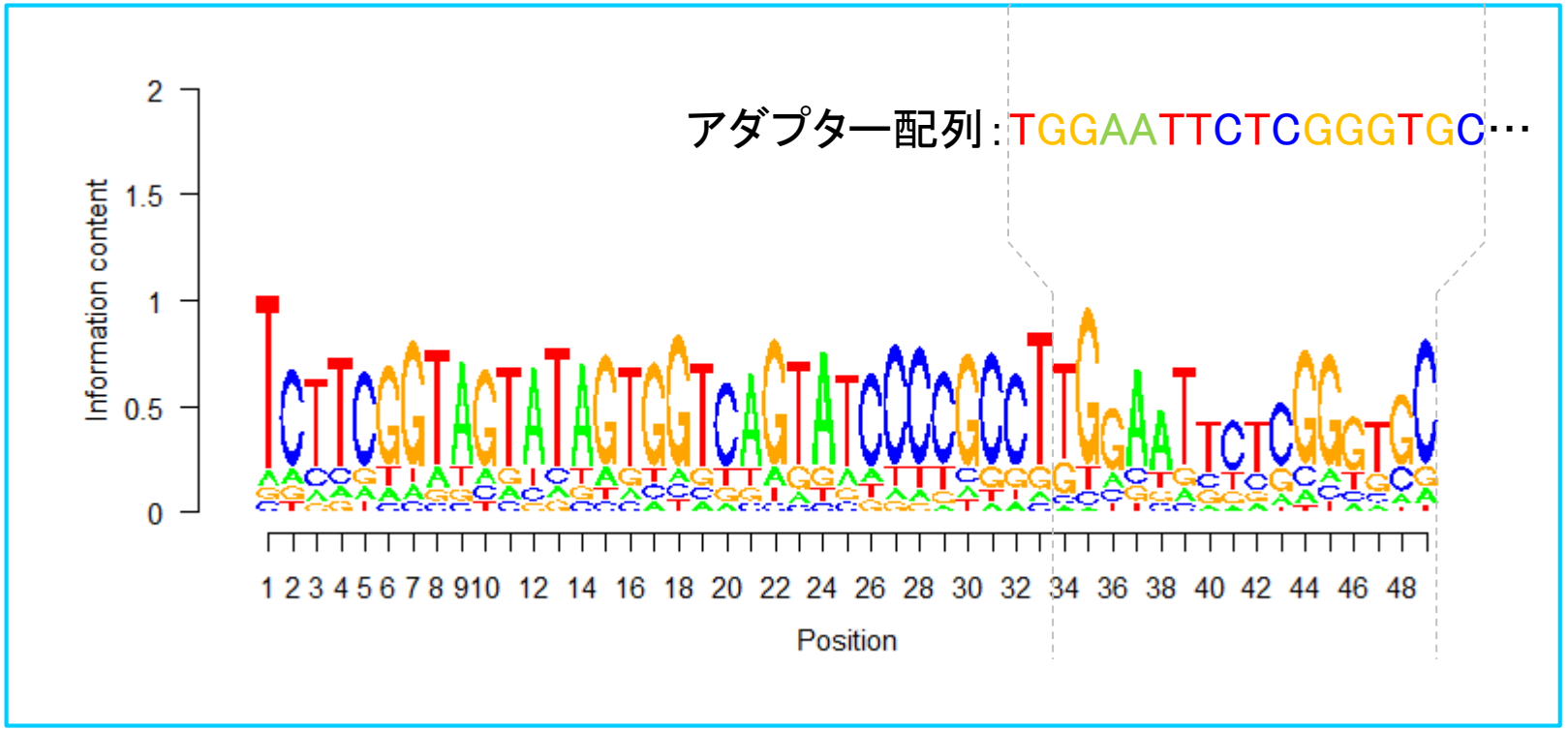


Sequence logosでも似たような結果。プログラムのバグでないことは確かだろう。



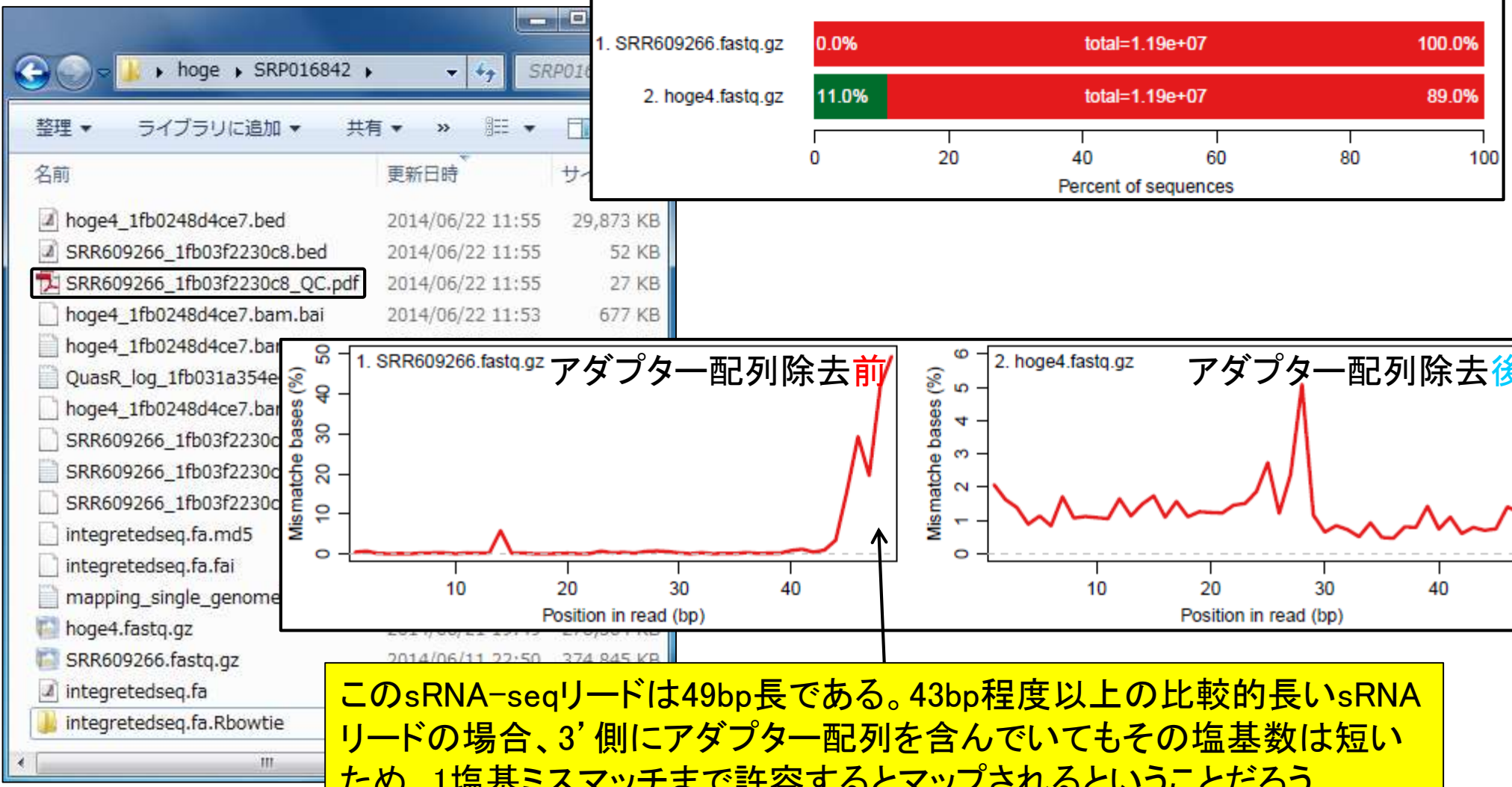
Schneider 1990)

正しくアダプター配列除去ができていることもわかる



実データのマッピング結果

教科書p89-90



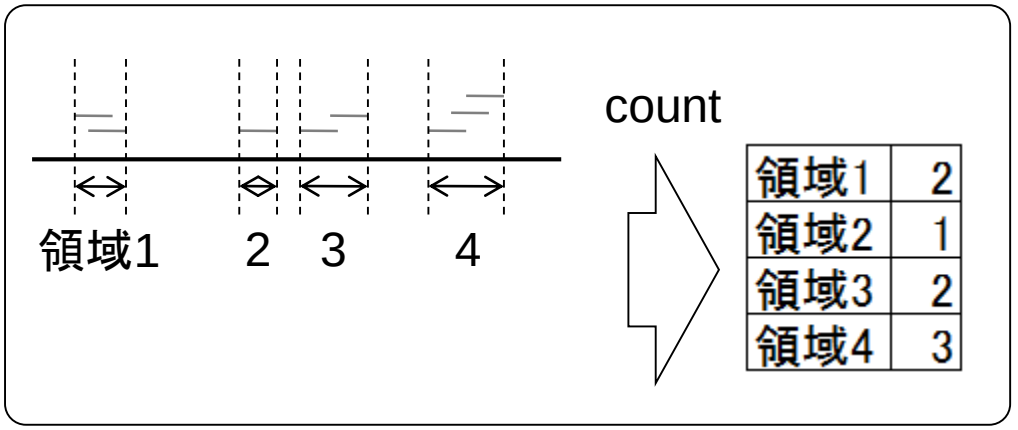
このsRNA-seqリードは49bp長である。43bp程度以上の比較的長いsRNAリードの場合、3’側にアダプター配列を含んでいてもその塩基数は短いため、1塩基ミスマッチまで許容するとマップされるということだろう。

課題

- アダプター配列除去前後のsmall RNA-seqデータを
  カイコゲノムにマップし、マップ率を比較する
  1. マッピング**前**の総リード数を述べよ
     - アダプター配列除去前のSRR609266.fastq.gz：
     - アダプター配列除去後のhoge4.fastq.gz：
  2. マッピング**後**の「**マップされたリード数**」を述べよ
     - アダプター配列除去前のSRR609266.fastq.gz：
     - アダプター配列除去後のhoge4.fastq.gz：
  3. 結果の考察。

マッピング結果からのカウント情報取得

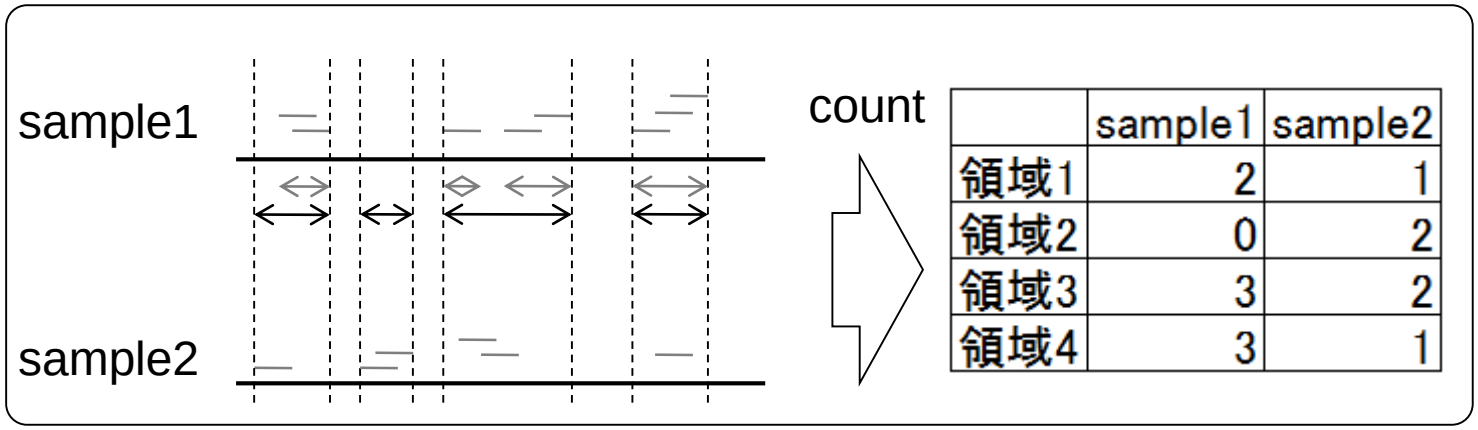
- アノテーション情報を利用する場合
  - UCSC Genes, Ensembl Genesなど様々なテーブル名を指定可能
  - gene, exon, promoter, junctionなど様々なレベルを指定可能
- アノテーション情報がない場合
  - マップされたリードの和集合領域を同定したのち、領域ごとのリード数をカウント
  - BEDtools (Quinlan et al., 2010)中のmergeBedプログラムを実行して和集合領域同定後、intersectBedプログラムを実行してリード数をカウントする作業に相当



基本的なイメージ

マッピング結果からのカウント情報取得

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



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

複数サンプルの場合には領域が変わりうる

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

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

教科書p90-95

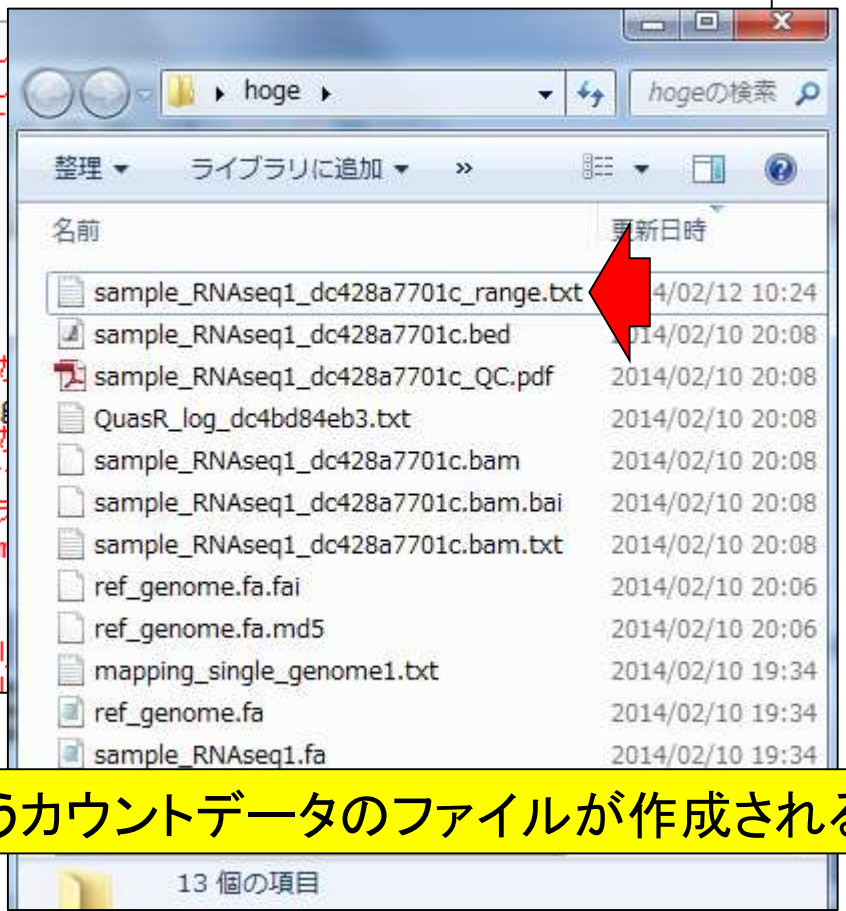
マップ後 | カウント情報取得 | ゲノム | アノテーション無 | QuasR(Lerch_XXX) NEW

QuasRパッケージを用いたsingle-end RNA-seqデータのリファレンスゲノム配列へのBowtieによるマッピングから、カウントデータ取得までの一連の流れを示します。アノテーション情報がない場合を想定しているので、GenomicAlignments パッケージを利用して、マップされたリードの和集合領域(union range)を得たのち、領域ごとにマップされたリード数をカウントしています。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

1.サンプルデータ18,19のRNA-seqデータ(sample_RNAseq1.fa)のref_genome.faへのマッピングの場合(mapping_single_genome1.txt):

オプションを"-m 1 --best --strata -v 0"とした例です。

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

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

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

#本番(マップされたリードの和集合領域同定)
tmpfname <- out@alignments[,1]         #ファイル名(in_f1の1列
tmpsname <- out@alignments[,2]         #サンプル名(in_f1の2列
```



*_range.txtというカウントデータのファイルが作成される

教科書p90-95

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

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

#前処理(マッピング)
time_s <- proc.time()                  #計算時間を計測するため
out <- qAlign(in_f1, in_f2, alignmentParameter=param_mapping)#マッピングを行うqAlign関数を実行した結果をoutに
time_e <- proc.time()                  #計算時間を計測するため
time_e - time_s                        #計算時間を表示(一番右側の数字。単位はsecond)
out                                    #マッピングに用いたパラメータや入力ファイルの情報などを表示
alignmentStats(out)                    #マッピング結果(alignment statistics)の表示。seqlength：リファレンス配

#本番(マップされたリードの和集合領域同定)
tmpfname <- out@alignments[,1]         #ファイル名(in_f1の1列目に相当)をtmpfnameとして取り扱いたいだけです
tmpsname <- out@alignments[,2]         #サンプル名(in_f1の2列目に相当)をtmpsnameとして取り扱いたいだけです
for(i in 1:length(tmpfname)){          #サンプル数(ファイル数)分だけループを回す
  if(i == 1){
    k <- readGAlignments(tmpfname[i])  #BAM形式ファイルを読み込んだ結果をkに格納(これはGAlignmentsオブジェクト
  } else{
    k <- c(k, readGAlignments(tmpfname[i]))#BAM形式ファイルを読み込んだ結果をkに格納(これはGAlignmentsオブジェ
  }
}
m <- reduce(granges(k))                #GRangesオブジェクトへの変換および

#本番(カウント情報取得)
tmp <- as.data.frame(m)                #出力ファイルのカウントデータ以外の
for(i in 1:length(tmpfname)){          #サンプル数(ファイル数)分だけループ
  tmpcount <- summarizeOverlaps(m, tmpfname[i])#GRangesオブジェクトmに対
  count <- assays(tmpcount)$counts     #SummarizedExperimentクラスオブジェクトのtmpcountからカウントデータ行列
  colnames(count) <- tmpsname[i]       #行列countの列名を変更
  tmp <- cbind(tmp, count)             #保存したい情報を連結してtmpに格納(和集合領域とカウント)
}

#ファイルに保存
out_f <- sub(".bam", "_range.txt", tmpfname[i])#変更したファイル名を作成した結果をout_fに格納
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.names=T)#tmpの中身を指定したファイル名
```

“*.bam”という文字列を”*_range.txt”という文字列に変更したものを出力ファイル名として自動的に生成している

マッピング結果からのカウント情報取得

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

QuasRパッケージを用いたsingle-end RNA-seqデータのリファレンスゲノム配列へのBowtieによるマッピングから、カ
の一連の流れを示します。アノテーション情報がない場合を想定しているので、GenomicAlignments パッケージを利
リードの和集合領域(union range)を得たのち、領域ごとにマップされたリード数をカウントしています。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

1.サンプルデータ18,19のRNA-seqデータ(sample_RNAseq1.fa)のref_genome.faへのマッピングの場合(mapping
オプションを"-m 1 --best --strata -v 0"とした例です。

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

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

*.bed

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

FileName	SampleName
sample_RNAseq1.fa	namae

*_range.txt

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

カウント数はこちら

マッピング結果からのカウント情報取得

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

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

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

FileName	SampleName
sample_RNAseq1.fa	sample1
sample_RNAseq2.fa	sample2

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

リストファイル中で指定したサンプル名がカウントデータ行列の列名となる

昔よく見かけたカウントデータ取得手段

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

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

上記情報はshort-readの頃の情報なので既に古いかも…。今はlong-readになっているのでsplice-aware alignerの一種のTophatなどから得られたカウント情報だろう

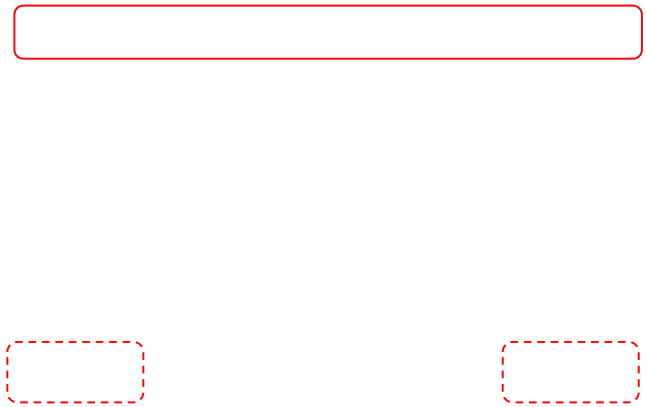
定量化：遺伝子レベル ⇔ isoformレベル

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

count情報を得る際に、どのexonの情報を用いるか?

遺伝子のカウント数の定義

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



様々な思想があり、当然その後の解析結果に影響を及ぼします

1.サンプルデータ18-20の複数のRNA-seqデータ(sample_RNAseq1.fa)をref_genome.faにマッピングする場合(mapping_single_genome1.txt): 教科書p90-95

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

```
for(i in 1:length(tmpfname)){          #サンプル数(ファイル数)分だけルー
  if(i == 1){
    k <- readGAlignments(tmpfname[i])  #BAM形式ファイルを読み込んだ結果を
  } else{
    k <- c(k, readGAlignments(tmpfname[i]))#BAM形式ファイルを読み込んだ結
  }
}
m <- reduce(granges(k))                #GRangesオブジェク

#本番(カウント情報取得)
tmp <- as.data.frame(m)                #出力ファイルのカウ
for(i in 1:length(tmpfname)){          #サンプル数(ファイ
  tmpcount <- summarizeOverlaps(m, tmpfname[i])#GRangesオ
  count <- assays(tmpcount)$counts     #SummarizedExperimentクラスオブジェクトのtmpcountからカウントデータ行列
  co
  tm
}

#フ
out_
writ
```

R Console

```
> ?summarizeOverlaps
starting httpd help server ... done
>
```

Unionがデフォルトらしいが、exon-union methodに相当する記述ではないようだ。他にもpaired-endやstrand情報など奥が深いのでご注意。

mode

mode can be one of the pre-defined count methods such as "Union", "IntersectionStrict", or "IntersectionNotEmpty" or it can be a user supplied count function. For a custom count funtion, the input arguments must match those of the pre-defined options and the function must return a vector of counts the same length as the annotation ('features' argument). See examples for details.

The pre-defined options are designed after the counting modes available in the HTSeq package by Simon Anders (see references).

- "Union" : (Default) Reads that overlap any portion of exactly one feature are counted. Reads that overlap multiple features are discarded. This is the most conservative of the 3 modes.
- "IntersectionStrict" : A read must fall completely "within" the feature to be counted. If a read overlaps multiple features but falls "within" only one, the read is counted for that feature. If the read is "within" multiple features, the read is discarded.
- "IntersectionNotEmpty" : A read must fall in a unique disjoint region of a feature to be counted. When a read overlaps multiple features, the features are partitioned into disjoint intervals. Regions that are shared between the features are discarded leaving only the unique disjoint regions. If the read overlaps one of these remaining regions, it is assigned to the feature the unique disjoint region came from.