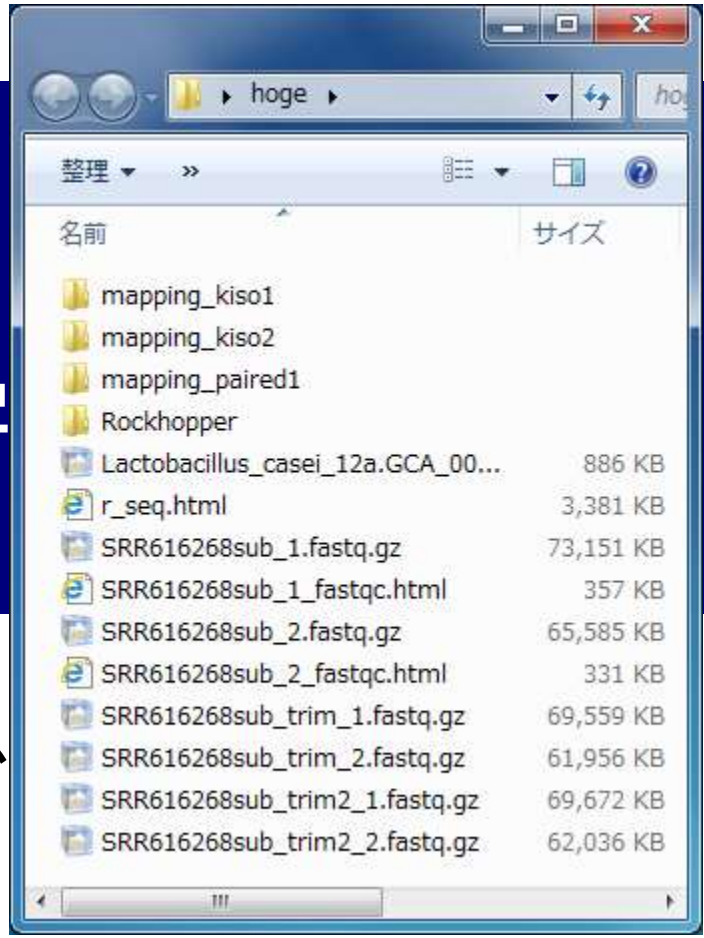
2015.07.01版

USBメモリ中のhogeフォルダをデスクトップにコピーしておいてください。

前回（6/23）のhogeフォルダがデスクトップに残っているかもしれないのでご注意ください。



農学生命情報科学特論I 第3回

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

講義予定

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-Seqに基づくトランスクリプトーム解析を題材とした講義を行います。

- 第1回（2015年6月16日）
  - データベース、データ取得、ファイル形式、Quality Control
  - 教科書の1.3節周辺
- 第2回（2015年6月23日）
  - Quality Control、k-mer解析、トリミング（アダプター配列除去）
- 第3回（2015年6月30日）
  - フィルタリング、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（2015年7月7日）
  - クラスタリング、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

Contents

- 先週の課題やエラーについて
  - 課題1の解説
  - アダプター配列除去（QuasR）実行時のエラーは門田のミス
  - Paired-endデータの取り扱い
- フィルタリング(filtering)
  - 基本形：リード数が同じpaired-endの場合（QuasRパッケージ）
  - 発展形：リード数が異なる場合（ShortReadパッケージ）
- アセンブル(assemble)
  - ゲノム用とトランスクリプトーム用
  - Rockhopper2（バクテリアのトランスクリプトーム用）
- マッピング（mapping）
  - 基礎、オプション、仮想データ説明
  - マッピング本番、出力ファイル形式、オプションと結果の関係
  - カウント情報取得
- 実データ解析
  - QuasRでマッピング → トリミング → 高精度なアセンブルとマッピングへ

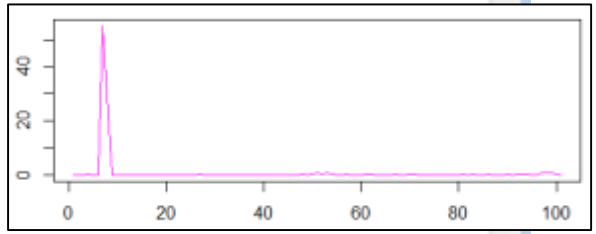
k-mer解析

「上級1」のコードをテンプレートにして、Position 7で特異的に出現しているAAGAGCAをRで確認してみよう。Head関数出力結果は最初の6個（つまりposition 1-6）までしか表示させていないので、ここではn=8としてposition 1-8まで表示させるようにしている。たしかにposition 7で期待値(Exp = 90.09)の55.15倍の出現回数になっていることが分かる。

```
#以下はおまけ(forループを用いて美しく...上級1)
param_len_ngs <- 107                    #リ
param_obj <- "CGGGCCT"                  #調
Obs <- NULL                             #お
hoge <- param_len_ngs - nchar(param_obj) +
for(i in 1:hoge){
    Obs <- c(Obs, table(subseq(fasta,
}
head(Obs)
mean(Obs)
Exp <- mean(Obs)
head(Obs/Exp)
plot(Obs/Exp, type="l", col="red")
```

R Console

```
> param_len_ngs <- 107                          #リード長を$
> param_obj <- "AAGAGCA"                        #調べたいk-m$
> Obs <- NULL                                   #おまじない
> hoge <- param_len_ngs - nchar(param_obj) + 1#positi$
> for(i in 1:hoge){                              #ループを回$
+     Obs <- c(Obs, table(subseq(fasta, start=i, widt$
+ }
> head(Obs, n=8)
AAGAGCA AAGAGCA AAGAGCA AAGAGCA AAGAGCA AAGAGCA
      3       6      11      27      11      21
AAGAGCA AAGAGCA
   4968    2096
> mean(Obs)
[1] 90.08911
> Exp <- mean(Obs)
> head(Obs/Exp, n=8)
    AAGAGCA     AAGAGCA     AAGAGCA     AAGAGCA
 0.03330036  0.06660073  0.12210133  0.29970326
    AAGAGCA     AAGAGCA     AAGAGCA     AAGAGCA
 0.12210133  0.23310254 55.14540059 23.26585339
> plot(Obs/Exp, type="l", col="magenta")
>
```

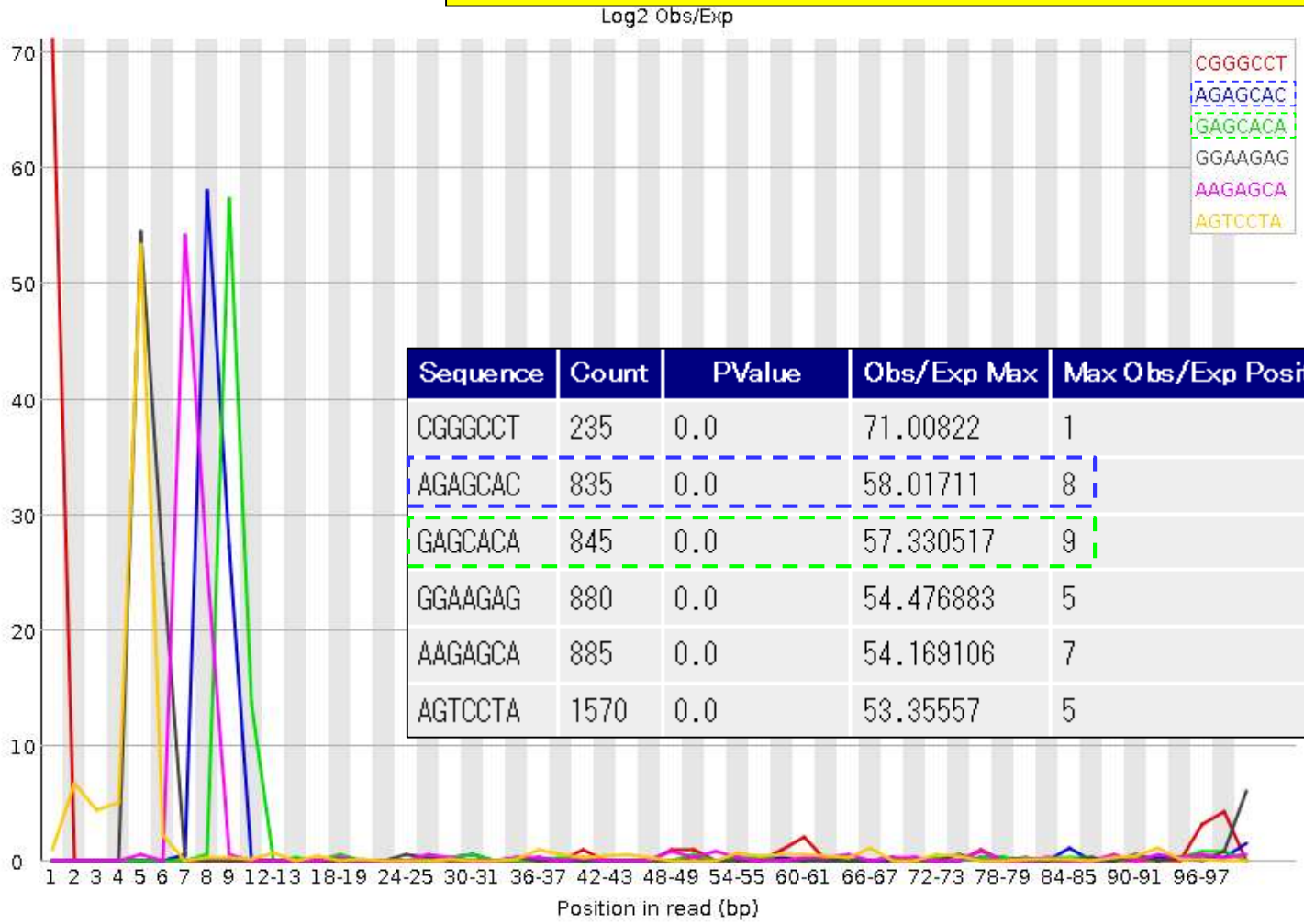


課題1

「上級1」のコードをテンプレートにして、①AGAGCAC、②GAGCACA、③長さを含めて任意、のk-mer解析を行い、得られた結果を示せ。

Summary

- Basic Statistics
- Per base sequence quality
- Per tile sequence quality
- Per sequence quality scores
- Per base sequence content
- Per sequence GC content
- Per base N content
- Sequence Length Distribution
- Sequence Duplication Levels
- Overrepresented sequences
- Adapter Content
- Kmer Content ①



Sequence	Count	PValue	Obs/Exp Max	Max Obs/Exp Position
CGGGCCT	235	0.0	71.00822	1
AGAGCAC	835	0.0	58.01711	8
GAGCACA	845	0.0	57.330517	9
GGAAGAG	880	0.0	54.476883	5
AAGAGCA	885	0.0	54.169106	7
AGTCCTA	1570	0.0	53.35557	5

課題1の解説

```
in_f <- "SRR616268sub_1.fastq.gz"        #入力ファイル名を指定してin_
↓
#必要なパッケージをロード↓
library(Biostrings)                      #パッケージの読み込み↓
↓
#入力ファイルの読み込み↓
fasta <- readDNAStringSet(in_f, format="fa
fasta                                    #確
↓
#以下はおまけ(forループを用いて美しく...上
param_len_ngs <- 107↓
param_obj <- "AGAGCAC"↓
Obs <- NULL↓
hoge <- param_len_ngs - nchar(param_obj) +
for(i in 1:hoge){↓
    Obs <- c(Obs, table(subseq(fasta, star
}↓
head(Obs, n=10)↓
mean(Obs)↓
Exp <- mean(Obs)↓
head(Obs/Exp, n=10)↓
plot(Obs/Exp, type="l", col="blue")↓
```

①AGAGCACのk-mer解析時にエラーが出たと思います。この原因は、AGAGCACが1つも存在しないポジションがあった場合に、そこがNAとなるからです。mean関数実行時に要素中にNAが1つでもあると計算できないという罠があったのです。

R Console

```
> head(Obs, n=10)
AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC
      2       3       8      19      14       4      30
AGAGCAC AGAGCAC AGAGCAC
   4969    2090       4
> mean(Obs)
[1] NA
> Exp <- mean(Obs)
> head(Obs/Exp, n=10)
AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC
     NA      NA      NA      NA      NA      NA      NA
AGAGCAC AGAGCAC AGAGCAC
     NA      NA      NA
> plot(Obs/Exp, type="l", col="blue")
Error in plot.window(...) :  有限な 'ylim' の値が必要で$
 追加情報: Warning messages:
1: In min(x) :  min の引数に有限な値がありません: Inf $
2: In max(x) :  max の引数に有限な値がありません: -Inf $
>
```

課題1の解説

```
in_f <- "SRR616268sub_1.fastq.gz"      #入力ファイル名を指定してin_fに格納
↓
#必要なパッケージをロード
library(Biostrings)                    #パッケージの読み込み
↓
#入力ファイルの読み込み
fasta <- readDNAStringSet(in_f, format="fa
fasta                                  #確
↓
#以下はおまけ(forループを用いて美しく...上
param_len_ngs <- 107
param_obj <- "AGAGCAC"
Obs <- NULL
hoge <- param_len_ngs - nchar(param_obj) +
for(i in 1:hoge){
    Obs <- c(Obs, table(subseq(fasta, star
}
head(Obs, n=10)
mean(Obs)
Exp <- mean(Obs)
head(Obs/Exp, n=10)
plot(Obs/Exp, type="l", col="blue")
```

Obsと打ち込んで、何が起こっているのかを悟る。画面に見えているだけでも、AGAGCACが存在しないpositionが少なくとも2ヶ所あることがわかる。

R Console

```
> Obs
AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC
      2       3       8      19      14       4      30
AGAGCAC AGAGCAC AGAGCAC    <NA> AGAGCAC AGAGCAC AGAGCAC
   4969    2090       4      NA       4      12       9
AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC
     18       5       4       6       5       5      13
AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC
      4       9       6       4       2       6      16
AGAGCAC AGAGCAC AGAGCAC AGAGCAC    <NA> AGAGCAC AGAGCAC
      7      19       2       9      NA       4       5
AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC
      4       7       2       4      15      10       4
AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC
      5       5       2       1       6       3       2
AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC
      8       8      17       5      14      24       8
AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC
      6       8       8      10      10       7      13
```

課題1の解説

```
in_f <- "SRR616268sub_1.fastq.gz"        #入力ファイル名を指定して
↓
#必要なパッケージをロード↓
library(Biostrings)                      #パッケージの読み込み↓
↓
#入力ファイルの読み込み↓
fasta <- readDNAStringSet(in_f, format="fa
fasta                                    #確
↓
#以下はおまけ(forループを用いて美しく...上
param_len_ngs <- 107↓
param_obj <- "AGAGCAC"↓
Obs <- NULL↓
hoge <- param_len_ngs - nchar(param_obj) +
for(i in 1:hoge){↓
    Obs <- c(Obs, table(subseq(fasta, star
}↓
head(Obs, n=10)↓
mean(Obs, na.rm=TRUE)↓
Exp <- mean(Obs, na.rm=TRUE)↓
head(Obs/Exp, n=10)↓
plot(Obs/Exp, type="l", col="blue")↓
```

最もシンプルな対処法は、mean関数中のna.rmオプションをTRUE（デフォルトはFALSE)にするやり方。NAの要素を読み飛ばすオプションなので、平均値の計算が若干不正確になるが、AGAGCACがposition 8で**相対的に**高頻度に出現しているということを調べるという点では全く問題ない。

R Console

```
> head(Obs, n=10)
AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC
      2       3       8      19      14       4      30
AGAGCAC AGAGCAC AGAGCAC
   4969    2090       4
> mean(Obs, na.rm=TRUE)
[1] 83.08081
> Exp <- mean(Obs, na.rm=TRUE)
> head(Obs/Exp, n=10)
    AGAGCAC     AGAGCAC     AGAGCAC     AGAGCAC
 0.02407295  0.03610942  0.09629179  0.22869301
    AGAGCAC     AGAGCAC     AGAGCAC     AGAGCAC
 0.16851064  0.04814590  0.36109422 59.80924012
    AGAGCAC     AGAGCAC
25.15623100  0.04814590
> plot(Obs/Exp, type="l", col="blue")
>
```

茂木 朋貴氏 他数名提供情報

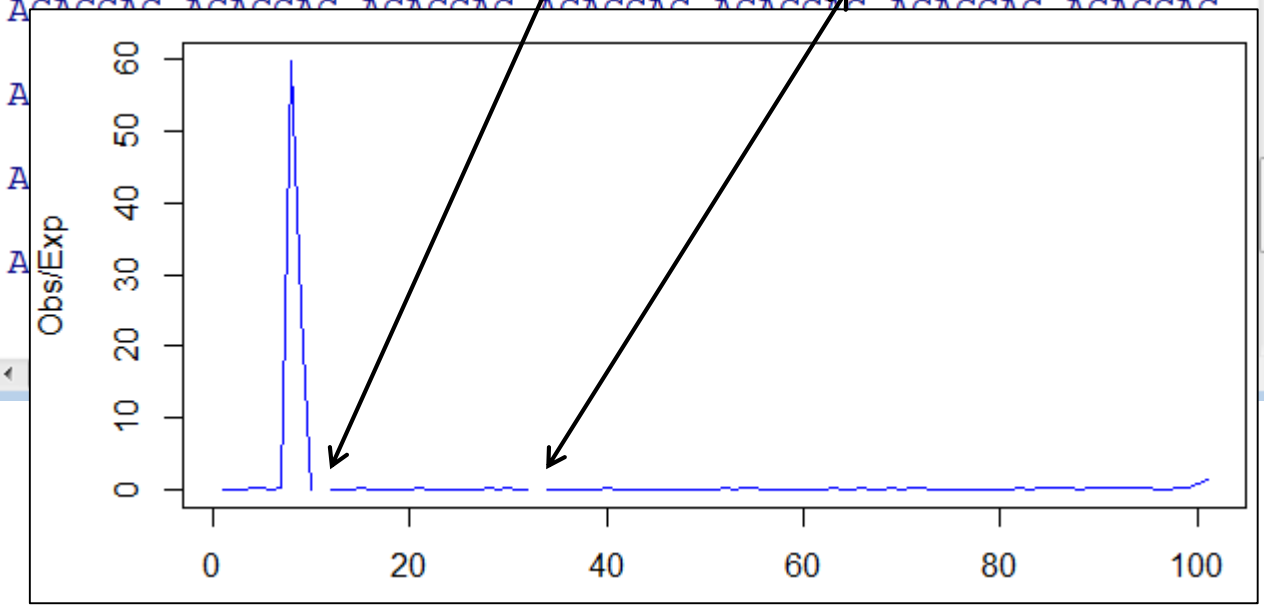
課題1の解説

平均値（この場合83.08081）さえ計算できれば、Obs/Expを計算可能。つまり、プロファイルを描画することができる。NAとなっている場所が2ヶ所あることがわかる。

```
in_f <- "SRR616268sub_1.fastq.gz"        #入力ファイル名を指定してin
↓
#必要なパッケージをロード↓
library(Biostrings)                      #パ
↓
#入力ファイルの読み込み↓
fasta <- readDNAStringSet(in_f, format="fa
fasta                                    #確
↓
#以下はおまけ(forループを用いて美しく...上
param_len_ngs <- 107↓
param_obj <- "AGAGCAC"↓
Obs <- NULL↓
hoge <- param_len_ngs - nchar(param_obj) +
for(i in 1:hoge){↓
    Obs <- c(Obs, table(subseq(fasta, star
}↓
head(Obs, n=10)↓
mean(Obs)↓
Exp <- mean(Obs)↓
head(Obs/Exp, n=10)↓
plot(Obs/Exp, type="l", col="blue")↓
```

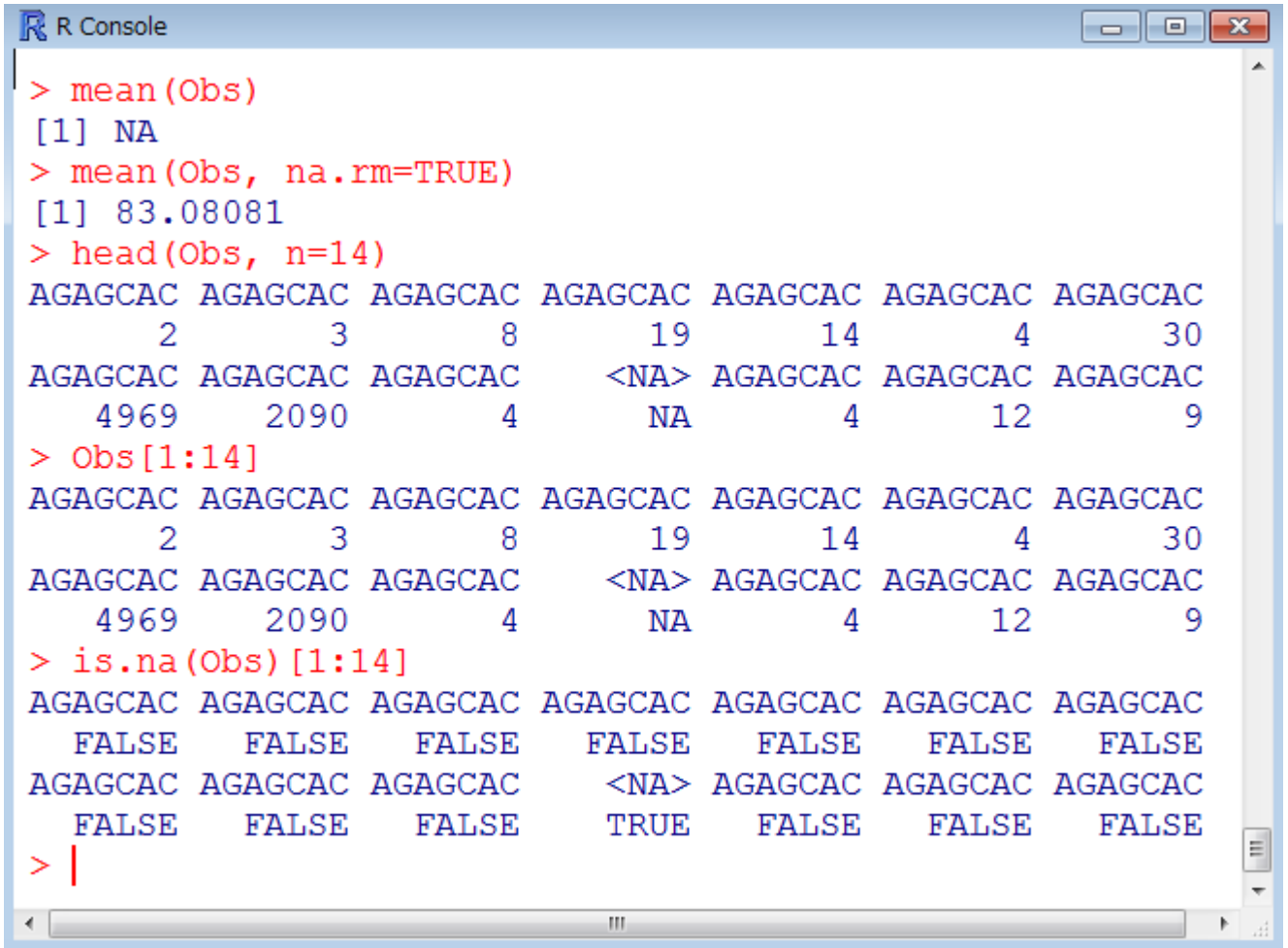
R Console

```
> Obs
AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC
      2       3       8      19      14       4      30
AGAGCAC AGAGCAC AGAGCAC    <NA> AGAGCAC AGAGCAC AGAGCAC
   4969    2090       4      NA       4      12       9
AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC
     18       5       4       6       5       5      13
AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC
      4       9       6       4       2       6      16
AGAGCAC AGAGCAC AGAGCAC AGAGCAC    <NA> AGAGCAC AGAGCAC
      7      19       2       9      NA       4       5
```



課題1の解説

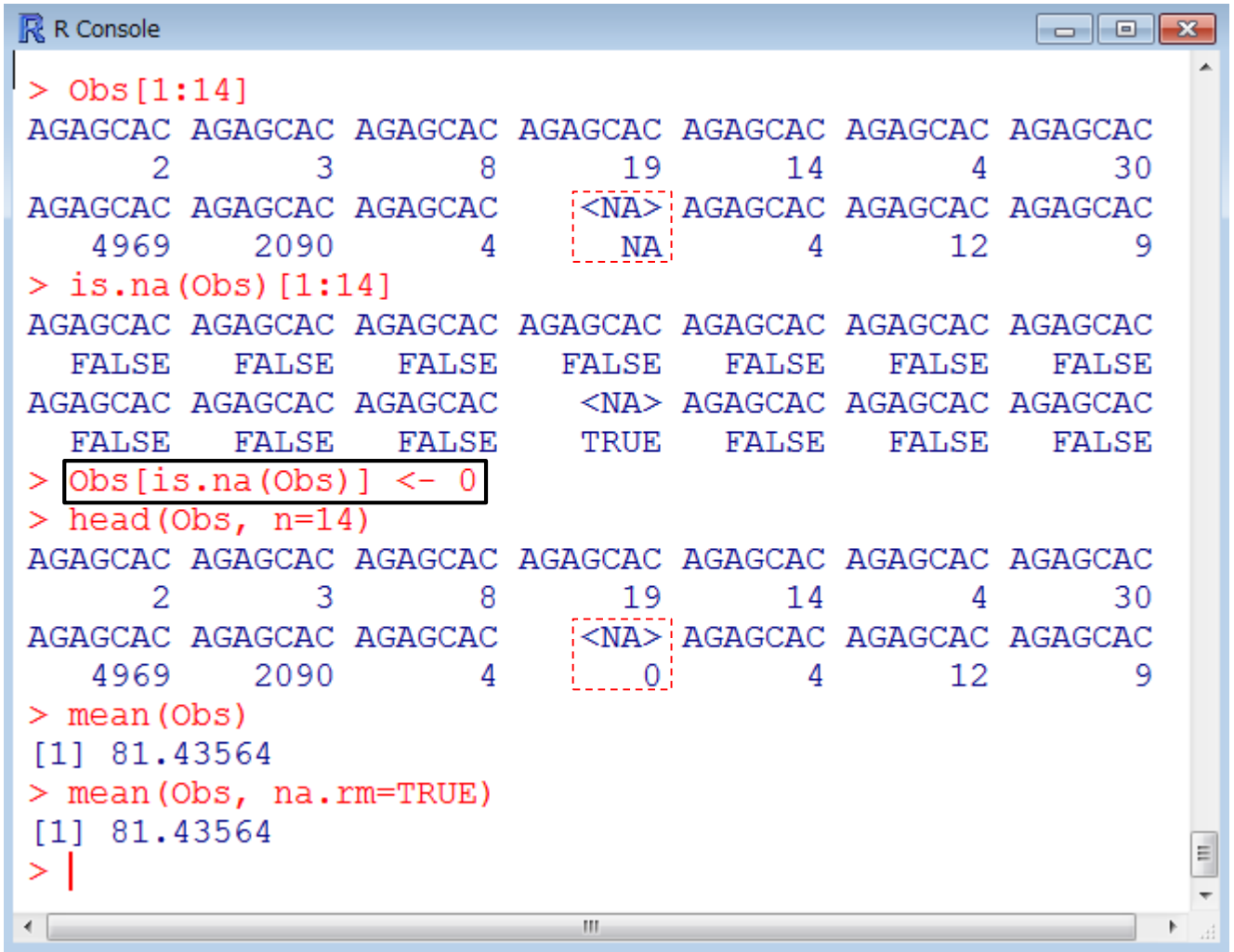
もっとよりよい方法は、①NAとなった要素に0を入れること。is.na関数は、要素がNAのときにTRUE、そうでないときにFALSEを返す。



野間口達洋氏 他数名提供情報

課題1の解説

0代入前後で出現頻度の平均値が83.08081から81.43564に少し下がっているのがわかります。



課題1の解説

こんな感じになります。2015年6月23日の課題を該当部分のみ差し替えたいヒトは、回収箱に入れてかまいません。課題用紙は後ろにあります。先週提出分と突き合わせて評価しますので、考察部分など差し替える必要がない部分は空欄で構いません。

```
in_f <- "SRR616268sub_1.fastq.gz"        #入力ファイル名を指定して
↓
#必要なパッケージをロード↓
library(Biostrings)                      #パ
↓
#入力ファイルの読み込み↓
fasta <- readDNAStringSet(in_f, format="fa
fasta                                    #確
↓
#以下はおまけ(forループを用いて美しく...上
param_len_ngs <- 107↓
param_obj <- "AGAGCAC"↓
Obs <- NULL↓
hoge <- param_len_ngs - nchar(param_obj) +
for(i in 1:hoge){↓
    Obs <- c(Obs, table(subseq(fasta, star
}↓
Obs[is.na(Obs)] <- 0↓
head(Obs, n=10)↓
mean(Obs, na.rm=TRUE)↓
Exp <- mean(Obs, na.rm=TRUE)↓
head(Obs/Exp, n=10)↓
plot(Obs/Exp, type="l", col="blue")↓
```

①

R Console

```
> Obs[is.na(Obs)] <- 0
> head(Obs, n=10)
AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC AGAGCAC
      2       3       8      19      14       4      30
AGAGCAC AGAGCAC AGAGCAC
   4969    2090       4
> mean(Obs, na.rm=TRUE)
[1] 81.43564
> Exp <- mean(Obs, na.rm=TRUE)
> head(Obs/Exp, n=10)
    AGAGCAC     AGAGCAC     AGAGCAC     AGAGCAC
 0.02455927  0.03683891  0.09823708  0.23331307
    AGAGCAC     AGAGCAC     AGAGCAC     AGAGCAC
 0.17191489  0.04911854  0.36838906 61.01750760
    AGAGCAC     AGAGCAC
25.66443769  0.04911854
> plot(Obs/Exp, type="l", col="blue")
>
```

バージョンアップ

ひっそりと修正しています。レポート中で問題点の指摘や解決法を示して頂いた方々に感謝m(_ _)m

- イントロ | NGS | 読み込み | FASTA形式 | 基本情報を取得 (last modified 2014/08/18)
- イントロ | NGS | 読み込み | FASTA形式 | description行の記述を整形 (last modified 2014/04/05)
- イントロ | NGS | 読み込み | FASTQ形式 | 基礎 (last modified 2015/06/24) NEW
- イントロ | NGS | 読み込み | FASTQ形式 | 応用 (last modified 2015/06/18) NEW

①

イントロ | NGS | 読み込み | FASTQ形式 | 基礎 NEW

Sanger FASTQ形式ファイルを読み込むやり方を示します。「基礎」では、FASTQファイルの中身を全て読み込む手順を示します。入力、出力形式は、ともに非圧縮(.fastq)、gzip圧縮(.fastq.gz)ファイルが可能です。

②

8. FASTQ形式ファイル(SRR616268sub_1.fastq.gz)の場合:

乳酸菌RNA-seqデータSRR616268の最初の100万リード分(約73MB)です。長さは全て107 bpです。NAを含む場合への各種対応策を2015年6月24日に追加しました(茂木朋貴氏、野間口達洋氏、他多くの受講生提供情報)。

```
#以下はおまけ(forループを用いて美しく...上級1)
param_len_ngs <- 107                    #リード長を指定
param_obj <- "CGGGCCT"                  #調べたいk-merを指定
Obs <- NULL                             #おまじない
hoge <- param_len_ngs - nchar(param_obj) + 1#positionの右端の値を計算してhogeに格
for(i in 1:hoge){                       #ループを回す
    Obs <- c(Obs, table(subseq(fasta, start=i, width=nchar(param_obj)))[param_
}
Obs[is.na(Obs)] <- 0                    #NAの位置に0を代入
head(Obs)                               #最初の6個の要素を表示
mean(Obs, na.rm=TRUE)                   #平均値
Exp <- mean(Obs, na.rm=TRUE)            #平均値をExpとして取り扱う
head(Obs/Exp)
plot(Obs/Exp, type="l", col="red")

#以下はおまけ(forループを用いて美しく...上級2)
param_obj <- "CGGGCCT"                  #調べたいk-merを指定
Obs <- NULL                             #おまじない
hoge <- width(fasta)[1] - nchar(param_obj) + 1#positionの右端の値を計算してhogeに
for(i in 1:hoge){                       #ループを回す
    Obs <- c(Obs, table(subseq(fasta, start=i, width=nchar(param_obj)))[param_
}
```

Contents

- 先週の課題やエラーについて
  - 課題1の解説
  - アダプター配列除去（QuasR）実行時のエラーは門田のミス
  - Paired-endデータの取り扱い
- フィルタリング(filtering)
  - 基本形：リード数が同じpaired-endの場合（QuasRパッケージ）
  - 発展形：リード数が異なる場合（ShortReadパッケージ）
- アセンブル(assemble)
  - ゲノム用とトランスクリプトーム用
  - Rockhopper2（バクテリアのトランスクリプトーム用）
- マッピング（mapping）
  - 基礎、オプション、仮想データ説明
  - マッピング本番、出力ファイル形式、オプションと結果の関係
  - カウント情報取得
- 実データ解析
  - QuasRでマッピング → トリミング → 高精度なアセンブルとマッピングへ

アダプター配列除去

param_nrecのところをいくら変えてもエラーが出続ける問題に多くの受講生が遭遇しました。理由はほぼ間違いなく、単純に私がpreprocessReads関数中に明示的にparam_nrecを与えていなかったからです。結果として、それが適切に反映されずにデフォルトの1,000,000がいつまでも使われていたということでしょう。

- 前処理 | フィルタリング | GFF/GTF形式ファイル (last modified 2013/10/10)
- 前処理 | フィルタリング | 組合せ | ACGTのみ & 指定した長さの範囲の配列 (last
- 前処理 | トリミング | ポリA配列除去 | ShortRead(Morgan_2009) (last modif
- 前処理 | トリミング | アダプター配列除去(基礎) | QuasR(Gaidatzis_2015) ①
- 前処理 | トリミング | アダプター配列除去(基礎) | girafe(Toedling_2010) (la
- 前処理 | トリミング |
- 前処理 | トリミング |
- 前処理 | トリミング |
- 前処理 | トリミング |
- 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
- マッピング | について
- マッピング | basic alig
- マッピング | splice-aw
- マッピング | Bisulfite
- マッピング | (ESTレベ
- マッピング | 基礎 (las
- マッピング | single-en

前処理 | トリミング | アダプター配列除去(基礎) |

QuasRパッケージを用いたアダプター配列除去の基本形を示します。param_nrecオプションは、一度に処理するリード数を指定しているところですが、基本的に無視で構いません。デフォルトの1000000のときに、メモリ不足でフリーズしたので、デフォルトの
です。それでも
「ファイル」-「ディ

1. gzip圧縮状態のF

small RNA-seqデー
中の記述から GSE
(Zhu_2013)の7を実
は"TGGAATTCTC
以外はデフォルトで
り、右側のアダプタ
いる理由です。約5

```
in_f <- "SRR60
out_f <- "hoge
param_adapter
param_nrec <-

#必要なパッケー
library(QuasR)

#本番(アダプター
res <- preproc
```

②

4. gzip圧縮状態のFASTQ形式ファイル(SRR609266.fastq.gz)の場合:

small RNA-seqデータ(ファイルサイズは400Mb弱、11928428リード)です。このファイルに対するFastQC実行結果として「RNA PCR Primer, Index 1」(RPI1)が含まれているとレポートされた49 bpをアダプター配列として入力しています。ここでは、アダプター配列以外はデフォルトで実行しています。アダプター配列の位置は5'側(左側)ではなく3'側(右側)にあるという前提であり、右側のアダプター配列しかトリムしないやり方です。それが、preprocessReads関数実行時にRpatternのみ記載している理由です。約5分。

```
in_f <- "SRR609266.fastq.gz"          #入力ファイル名を指定してin_fに格納
out_f <- "hoge4.fastq.gz"             #出力ファイル名を指定してout_fに格納
param_adapter <- "TGGAATTCTCGGGTGCCAAGGAACTCCAGTCACATCACGATCTCGTATG"#アダプター配
param_nrec <- 500000                  #一度に処理するリード数を指定

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

#本番(アダプター配列除去)
res <- preprocessReads(filename=in_f, #アダプター配列除去を実行
            outputFilename=out_f,     #アダプター配列除去を実行
            Rpattern=param_adapter)   #アダプター配列除去を実行
res                                   #確認してるだけです
file.size(in_f)                       #入力ファイルのサイズを表示
file.size(out_f)                      #出力ファイルのサイズを表示
```

中村浩正氏 提供情報

アダプター配列除去

休み時間にでも動作確認してみてください。きっとうまくいくことでしょう。

- 前処理 | フィルタリング | GFF/GTF形式ファイル (last modified 2013/10/10)
- 前処理 | フィルタリング | 組合せ | ACGTのみ & 指定した長さの範囲の配列 (last
- 前処理 | トリミング | ポリA配列除去 | ShortRead(Morgan_2009) (last modif
- 前処理 | トリミング | アダプター配列除去(基礎) | QuasR(Gaidatzis_2015) ①
- 前処理 | トリミング | アダプター配列除去(基礎) | girafe(Toedling_2010) (la
- 前処理 | トリミング |
- 前処理 | トリミング |
- 前処理 | トリミング |
- 前処理 | トリミング |
- 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
- マッピング | について
- マッピング | basic alig
- マッピング | splice-aw
- マッピング | Bisulfite
- マッピング | (ESTレベ
- マッピング | 基礎 (las
- マッピング | single-en

前処理 | トリミング | アダプター配列除去(基礎) | QuasR(Gaidatzis_2015) NEW

QuasRパッケージを用いたアダプター配列除去の基本形を示します。param_nrecオプションは、一度に処理するリード数を指定しているところですが、基本的に無視で構いません。デフォルトの1000000のときに、メモリ不足でフリーズしたので、デフォルトの
です。それでも
「ファイル」-「ディレク

②

1. gzip圧縮状態のF

small RNA-seqデー
中の記述から GSE
(Zhu_2013)の7を実
は"TGGAATTCTC
以外はデフォルトで
り、右側のアダプタ
いる理由です。約5

```
in_f <- "SRR60
out_f <- "hoge
param_adapter
param_nrec <-

#必要なパッケー
library(QuasR)

#本番(アダプタ
res <- preproc
```

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

small RNA-seqデータ(ファイルサイズは400Mb弱、11928428リード)です。このファイルに対するFastQC実行結果として「RNA PCR Primer, Index 1」(RPI1)が含まれているとレポートされた49 bpをアダプター配列として入力しています。ここでは、アダプター配列以外はデフォルトで実行しています。アダプター配列の位置は5'側(左側)ではなく3'側(右側)にあるという前提であり、右側のアダプター配列しかトリムしないやり方です。それが、preprocessReads関数実行時にRpatternのみ記載している理由です。約5分。

```
in_f <- "SRR609266.fastq.gz"            #入力ファイル名を指定してin_fに格納
out_f <- "hoge4.fastq.gz"               #出力ファイル名を指定してout_fに格納
param_adapter <- "TGGAATTCTCGGGTGCCAAGGAACTCCAGTCACATCACGATCTCGTATG"#アダプター配
param_nrec <- 500000                    #一度に処理するリード数を指定

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

#本番(アダプター配列除去)
res <- preprocessReads(filename=in_f,   #in_fを入力とする
            outputFilename=out_f,       #out_fを出力とする
            Rpattern=param_adapter,     #リードの右側(Right)のアダプターを除去
            nrec=param_nrec)            #一度に処理するリード数
res                                     #確認してるだけです
file.size(in_f)                         #入力ファイルのサイズを表示
file.size(out_f)                        #出力ファイルのサイズを表示
```

2015年6月23日の講議資料

Paired-endの取扱い

- 前処理 | トリミング | ポリA配列除去 | ShortRead(Morgan_2009) (last modified 2014/06/11)
- 前処理 | トリミング | アダプター配列除去(基礎) | QuasR(Gaidatzis_2015) (last modified 2015/06/22)推奨
- 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)
- 前処理 | トリミング | アダプター配列除去(基礎) | ShortRead(Morgan_2009) (last modified 2014/06/21)
- 前処理 | トリミング | アダプター配列除去(応用) | QuasR(Gaidatzis_2015) (last modified 2015/06/23)推奨
- 前処理 | トリミング | アダプター配列除去(応用) | ShortRead(Morgan_2009) (last modified 2014/06/18)
- 前処理 | トリミング | 指定した末端塩基数は除去 (last modified 2013/06/11)
- アセンブル | につい
- アセンブル | ゲノム
- アセンブル | トラン
- マッピング | につい
- マッピング | basic a
- マッピング | splice-
- マッピング | Bisulfi
- マッピング | (ESTレ
- マッピング | 基礎 (
- マッピング | single-
- マッピング | single-

① ②

前処理 | トリミング | アダプター配列除去(応用) | QuasR(Gaidatzis_2015) NEW

QuasRパッケージを
生してR Guiが終了
6月21日にparam_nre
構いません。。デフォル
す。処理時間は長くなり
「ファイル」-「ディレクトリ

1. gzip圧縮状態のFAST

small RNA-seqデータ(
2013)中の記述から GS
SRAdb(Zhu_2013)のア
の低いリードを除去した
ませんでしたが、Table
(例:"GCAGTCGTGG
列ということになります
は"TGGAATTCTCGG
チまで許容して(推定)
が18nt以上のものをフ
あるという前提であり、
RpatternとmaxRmisma

```
in_f <- "SRR6092
out_f <- "hoge1.
param_adapter <-
```

2. gzip圧縮FASTQ形式ファイル(SRR616268sub_1.fastq.gzとSRR616268sub_2.fastq.gz)の場合:

乳酸菌RNA-seqデータSRR616268の最初の100万リード分です。paired-endデータです。SRR616268sub_1.fastq.gzは、約75MB、全リード107 bpです。SRR616268sub_2.fastq.gzは、約67MB、全リード93 bpです。FastQC実行結果として「TruSeq Adapter, Index 3」が含まれているとレポートされました。これでググると塩基配列情報は"GATCGGAAGAGCACACGTCTGAACTCCAGTCACTTAGGCATCTCGTATGCCGTCTTCTGCTTG"と書いてあったので、これを入力として与えます。アダプター配列の位置は5'側(左側)にあるという前提であり、左側のアダプター配列しかトリムしないやり方です。それが、preprocessReads関数実行時にLpatternのみ記載している理由です。残念ながら、QuasR (ver. 1.8.2)では「Removing adapters from paired-end samples is not yet supported」と出ます。2015年6月23日に開発者に実装を心待ちにしているとメールをしておきました。

```
in_f1 <- "SRR616268sub_1.fastq.gz"
in_f2 <- "SRR616268sub_2.fastq.gz"
out_f1 <- "hoge1_1.fastq.gz"
out_f2 <- "hoge1_2.fastq.gz"
param_adapter <- "GATCGGAAGAGCACACG
param_nrec <- 500000

#必要なパッケージをロード
library(QuasR)

#本番(前処理)
res <- preprocessReads(filename=in_
                       filenameMate=in_f2,
                       outputFilename=out_f1,
                       outputFilenameMate=out_f2,#前処理を実行
                       Lpattern=param_adapter)    #前処理を実行
res                                               #確認してるだけです
```

R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files(pattern="sub_..fastq")
[1] "SRR616268sub_1.fastq.gz"
[2] "SRR616268sub_1_fastqc.html"
[3] "SRR616268sub_2.fastq.gz"
[4] "SRR616268sub_2_fastqc.html"
>
```

Paired-endの取扱い

QuasR (ver. 1.8.2)では、まだpaired-endデータのアダプター配列除去には対応できていないようだといいましたが、その後すぐに開発者が回避策(workaround)を教えてくれました。

2. gzip圧縮FASTQ形式ファイル(SRR616268sub_1.fastq.gzとSRR616268sub_2.fastq.gz)の場

乳酸菌RNA-seqデータSRR616268の最初の100万リード分です。paired-endデータです。SRR6は、約75MB、全リード107 bpです。SRR616268sub_2.fastq.gzは、約67MB、全リード93 bpです。して「TruSeq Adapter, Index 3」が含まれているとレポートされました。これでググると塩基配列情報は"GATCGGAAGAGCACACGTCTGAACTCCAGTCACTTAGGCATCTCGTATGCCGTCTTCTGCTTG"と書いてあったので、これを入力として与えます。アダプター配列の位置は5'側(左側)にあるという前提であり、左側のアダプター配列しかトリムしないやり方です。それが、preprocessReads関数実行時にLpatternのみ記載している理由です。残念ながら、QuasR (ver. 1.8.2)では「Removing adapters from paired-end samples is not yet supported」と出ます。2015年6月23日に開発者に実装を心待ちにしているとメールをしておきました。

```
in_f1 <- "SRR616268sub_1.fastq.gz"      #入力ファイル名を指定してin_f1に格納
in_f2 <- "SRR616268sub_2.fastq.gz"      #入力ファイル名を指定してin_f2に格納
out_f1 <- "hoge1_1.fastq.gz"            #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge1_2.fastq.gz"            #出力ファイル名を指定してout_f2に格納
param_adapter <- "GATCGGAAG
param_nrec <- 500000

#必要なパッケージをロード
library(QuasR)

#本番(前処理)
res <- preprocessReads(file
              filenameMate=i
              outputFilename
              outputFilename
              Lpattern=param
res
```

R Console

```
 要求されたパッケージ Rbowtie をロード中です
>
> #本番(前処理)
> res <- preprocessReads(filename=in_f1, #前処理を実行
+                filenameMate=in_f2,     #前処理を実行
+                outputFilename=out_f1,  #前処理を実行
+                outputFilenameMate=out_f2,#前処理を実行
+                Lpattern=param_adapter)  #前処理を実行
Error in preprocessReads(filename = in_f1, filenameMate = in_f2, $
  Removing adapters from paired-end samples is not yet supported $
> res                                    #確認してるだけです
 エラー:  オブジェクト 'res' がありません
>
```

Thanks to Dr. Stadler

回避策

3. gzip圧縮FASTQ形式ファイル(SRR616268sub_1.fastq.gz)の場合:

乳酸菌RNA-seqデータSRR616268の最初の100万リード分です。paired-endデータのforward側、約75MB、全リード107 bpです。FastQC実行結果(SRR616268sub_1_fastqc.html)として「TruSeq Adapter, Index 3」が含まれているとレポートされました。これでググると塩基配列情報は"GATCGGAAGAGCACACGTCTGAACTCCAGTCACTTAGGCATCTCGTATGCCGTCTTCTGCTTG"と書いてあったので、これを入力として与えます。アダプター配列の位置は5'側(左側)にあるという前提であり、左側のアダプター配列しかトリムしないやり方です。この例題では、アダプター配列除去のみしか行わず出力ファイル中のリード数が入力ファイルと同じになるようにしています。これを確実に実現するために、許容するNの数としてトテツモナイ大きさの1000をparam_nBasesで与え、アダプター配列除去後の許容する最低配列長として全てが残る0をparam_minLengthで与えています。それが、preprocessReads関数実行時にLpatternのみ記載している理由です。約1分。出力ファイルでもう一度FastQCを実行すると、「TruSeq Adapter, Index 3」がOverrepresented sequencesの項目から消えていることまでは確認しました。しかし、まだ「TruSeq Adapter, Index 2」が残っています。ファイルサイズが、74,906,576 bytesから71,542,907 bytesに若干減っていることがわかります。

```
in_f <- "SRR616268sub_1.fastq.gz"       #入力ファイル名を指定してin_fに格納
out_f <- "hoge3.fastq.gz"               #出力ファイル名を指定
param_adapter <- "GATCGGAAGAGCACACGTCTGAACTCCAGTCACTTAGGCA
param_nBases <- 1000                    #許容するNの数を指定
param_minLength <- 0                    #アダプター配列除去後
param_nrec <- 500000                    #一度に処理するリード

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

#本番(アダプター配列除去)
res <- preprocessReads(filename=in_f,   #in_fを入力とする
             outputFilename=out_f,      #out_fを出力とする
             Lpattern=param_adapter,    #リードの左側(Left)
             nBases=param_nBases,       #許容するNの数
             minLength=param_minLength,#アダプター配列除去後
             nrec=param_nrec)           #一度に処理するリード
res                                     #確認してるだけです
file.size(in_f)                         #入力ファイルのサイズ
file.size(out_f)                        #出力ファイルのサイズ
```

R Console

```
> res                                                    #$
                        SRR616268sub_1.fastq.gz
totalSequences                          1000000
matchTo5pAdapter                         456561
matchTo3pAdapter                              0
tooShort                                      0
tooManyN                                      0
lowComplexity                                 0
totalPassed                             1000000
> file.size(in_f)                                      #$
[1] 74906576
> file.size(out_f)                                     #$
[1] 71542907
>
```

基本戦略は「single-endとして別々にアダプター配列除去を行うが、出力ファイルのリード数が変わらないようにする」です。乳酸菌paired-end RNA-seqデータ(SRR626268)のアダプター配列除去を行う一連の手順の基本形は、例題3-5です。例題3では、Forward側のリードファイルを入力として、FastQCでレポートされた「TruSeq Adapter, Index 3」の除去を行っています。約1分

回避策

4. gzip圧縮FASTQ形式ファイル(hoge3.fastq.gz)の場合:

3.で得られたhoge3.fastq.gzファイル中には、まだ「TruSeq Adapter, Index 2」が残っています。「TruSeq Adapter, Index 2」の塩基配列情報は "GATCGGAAGAGCACACGTCTGAACTCCAGTCACCGATGTATCTCGTATGCCGTCTTCTGCTTG" と書いてあったので、これを入力として与えます。約1分。ファイルサイズが、71,542,907 bytesから71,343,605 bytesに若干減っていることがわかります。この出力ファイル(hoge4.fastq.gz)と同じものが SRR616268sub_trim2_1.fastq.gz (1,000,000リード、71,343,605 bytes)です。FastQC実行結果はSRR616268sub_trim2_1_fastqc.htmlです。

```
in_f <- "hoge3.fastq.gz"                    #入力ファイル名を指定してin_fに格納
out_f <- "hoge4.fastq.gz"                   #出力ファイル名を指定してout_fに格納
param_adapter <- "GATCGGAAGAGCACACGTCTGAACTCCAGTCACCGATGTATCTCGTATGCCGTCTTCTGCTTG"#アダ
param_nBases <- 1000                        #許容するNの数を指定
param_minLength <- 0                        #アダプター配列除去後の許容する最低配列長を指定
param_nrec <- 500000                        #一度に処理するリード数を指定

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

#本番(アダプター配列除去)
res <- preprocessReads(filename=in_f,       #in_fを入力とする
             outputFilename=out_f,          #out_fを出力とする
             Lpattern=param_adapter,        #リードの左側(Left)
             nBases=param_nBases,           #許容するNの数
             minLength=param_minLength,     #アダプター配列除去後
             nrec=param_nrec)               #一度に処理するリード
res                                         #確認してるだけです
file.size(in_f)                             #入力ファイルのサイズ
file.size(out_f)                            #出力ファイルのサイズ
```

R Console

```
> res                                                    #$
                    hoge3.fastq.gz
totalSequences             1000000
matchTo5pAdapter            170381
matchTo3pAdapter                 0
tooShort                         0
tooManyN                         0
lowComplexity                    0
totalPassed                1000000
> file.size(in_f)                                       #$
[1] 71542907
> file.size(out_f)                                      #$
[1] 71343605
>
```

例題4では、例題3の出力ファイルを入力として、もう1つレポートされていた「TruSeq Adapter, Index 2」の除去を行っています。FastQC実行環境にない場合でも、用いた実験プロトコルが分かれば、候補となるadapterやprimer配列の一部で文字列検索すればある程度わかります。もちろん、この場合はIndex配列部分を含めたりしたほうがいいとは思います。約1分

回避策

5. gzip圧縮FASTQ形式ファイル(SRR616268sub_2.fastq.gz)の場合:

乳酸菌RNA-seqデータSRR616268の最初の100万リード分です。paired-endデータのreverse側、約67MB、93 bpです。FastQC実行結果(SRR616268sub_2_fastqc.html)として「Illumina Single End PCR Primer 1」がいるとレポートされました。これでググると塩基配列情報は"AATGATACGGCGACCACCGAGATCTACACTCTTTCCCTACACGACGCTCTTCCGATCT"と書いて で、これを入力として与えます。アダプター配列の位置は5'側(左側)にあるという前提であり、左側のアダ 列しかトリムしないやり方です。それが、preprocessReads関数実行時にLpatternのみ記載している理由で 分。ファイルサイズが、67,158,462 bytesから63,524,791 bytesに若干減っていることがわかります。この出 ル(hoge5.fastq.gz)と同じものが SRR616268sub_trim2_2.fastq.gz(1,000,000リード、63,524,791 bytes)です。 行結果はSRR616268sub_trim2_2_fastqc.htmlです。

```
in_f <- "SRR616268sub_2.fastq.gz"       #入力ファイル名を指定してin_fに格納
out_f <- "hoge5.fastq.gz"               #出力ファイル名を指定してout_fに格納
param_adapter <- "AATGATACGGCGACCACCGAGATCTACACTCTTTCCCTACACGACGCTCTTCCGATCT"#
param_nBases <- 1000                    #許容するNの数を指定
param_minLength <- 0                    #アダプター配列除去後
param_nrec <- 500000                    #一度に処理するリード

#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(Biostrings)                     #パッケージの読み込み

#前処理(reverse complementの作成)
hoge <- reverseComplement(DNAString(param_adapter))#逆相補

#本番(アダプター配列除去)
res <- preprocessReads(filename=in_f,   #in_fを入力とする
            outputFilename=out_f,       #out_fを出力とする
            Lpattern=as.character(hoge),#リードの左側(Left
            nBases=param_nBases,        #許容するNの数
            minLength=param_minLength,#アダプター配列除去後
            nrec=param_nrec)            #一度に処理するリード
res                                     #確認してるだけです
```

例題5では、Reverse側のリードファイルを入力として、FastQCでレポートされた「Illumina Single End PCR Primer 1」の除去を行っています。Primer配列の場合は、逆相補鎖(reverse complement)を作成してから与えないとうまく除去できなかったことから、このようにしています。FastQCを実行できる環境にあれば、「overrepresented sequences」項目のpossible sourceのところがNo Hitになったかどうかでうまくいったかどうかの判定ができます。約1分

R Console

```
> res                                   #$
                    SRR616268sub_2.fastq.gz
totalSequences                      1000000
matchTo5pAdapter                     273793
matchTo3pAdapter                          0
tooShort                                  0
tooManyN                                  0
lowComplexity                             0
totalPassed                         1000000
> file.size(in_f)                       #$
[1] 67158462
> file.size(out_f)                      #$
[1] 63524791
>
```

Contents

- 先週の課題やエラーについて
  - 課題1の解説
  - アダプター配列除去（QuasR）実行時のエラーは門田のミス
  - Paired-endデータの取り扱い
- フィルタリング(filtering)
  - 基本形：リード数が同じpaired-endの場合（QuasRパッケージ）
  - 発展形：リード数が異なる場合（ShortReadパッケージ）
- アセンブル(assemble)
  - ゲノム用とトランスクリプトーム用
  - Rockhopper2（バクテリアのトランスクリプトーム用）
- マッピング（mapping）
  - 基礎、オプション、仮想データ説明
  - マッピング本番、出力ファイル形式、オプションと結果の関係
  - カウント情報取得
- 実データ解析
  - QuasRでマッピング → トリミング → 高精度なアセンブルとマッピングへ

フィルタリング：基本形

リード数の揃ったpaired-endデータを入力として、許容するNの数と最短配列長でフィルタリングするやり方を示します。

- 前処理 | フィルタリング | Illuminaの pass filtering (last modified 2013/06/19)
- 前処理 | フィルタリング | GFF/GTF形式ファイル (last modified 2013/10/10)
- 前処理 | フィルタリング | 組合せ | ACGTのみ & 指定した長さの範囲の配列 (last modified 2014/06/11)
- 前処理 | フィルタリング | paired-end | 配列長とN数 | QuasR(Gaidatzis_2015) ① modified 2015/06/26) NE
- 前処理 | フィルタリング | paired-end | 共通リード抽出 | ShortRead(Morgan_200 (last modified 2015/06/26)
- 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
- マッピング | について (
- マッピング | basic aligne
- マッピング | splice-awar
- マッピング | Bisulfite se
- マッピング | (ESTレベル
- マッピング | 基礎 (last r
- マッピング | single-end
- マッピング | single-end
- マッピング | single-end
- マップ後 | について(las

前処理 | フィルタリング | paired-end | 配列長とN数 | QuasR(Gaidatzis_2015)

NEW

QuasRパッケージを用い ② ルタリング 例を示します。
「ファイル」-「ディレクトリの変

1. gzip圧縮FASTQ形式ファ

前処理 | トリミング | アダプ
paired-endデータです。SRR
SRR616268sub_trim2_2.fas
tooManyNが1,551リードあ
す。尚、998,431 + 107 + 1,
(union)は(1,000,000 - 998,4

```
in_f1 <- "SRR616268s
in_f2 <- "SRR616268s
out_f1 <- "hoge1_1.f
out_f2 <- "hoge1_2.f
param_nrec <- 500000

#必要なパッケージをロ
library(QuasR)

#本番(前処理)
res <- preprocessRea
                filenam
```

2. gzip圧縮FASTQ形式ファイル(SRR616268sub_trim2_1.fastq.gzとSRR616268sub_trim2_2.fastq.gz)の場合:

許容するN数(param_nBases)と最短配列長(param_minLength)を明示的に与えるやり方です。tooShortが119、tooManyNが1,762リードあったことがわかります。結果として、998,208リードからなるファイルが出力されています。例題6に比べて若干生き残るリード数が減るのは、param_nBases(デフォルトは2)とparam_minLength(デフォルトは14)での条件がデフォルトよりも若干厳しめ(1と18)だからです。

```
in_f1 <- "SRR616268sub_trim2_1.fastq.gz"#入力ファイル名を指定してin_f1に格納
in_f2 <- "SRR616268sub_trim2_2.fastq.gz"#入力ファイル名を指定してin_f2に格納
out_f1 <- "hoge2_1.fastq.gz"            #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge2_2.fastq.gz"            #出力ファイル名を指定してout_f2に格納
param_nBases <- 1                       #許容するNの数を指定
param_minLength <- 18                   #アダプター配列除去後の許容する最短配列長を指定
param_nrec <- 500000                    #一度に処理するリード数を指定

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

#本番(前処理)
res <- preprocessReads(filename=in_f1, #前処理を実行
            filenameMate=in_f2,        #前処理を実行
            outputFilename=out_f1,     #前処理を実行
            outputFilenameMate=out_f2,#前処理を実行
            nBases=param_nBases,       #許容するNの数
            minLength=param_minLength,#アダプター配列除去後の許容する最短配列長
            nrec=param_nrec)           #一度に処理するリード数
res                                    #確認してるだけです
```

フィルタリング：基本形

この例題で用いているファイルはhoge4.fastq.gzとhoge5.fastq.gzでも構いません。中身は同じです。

2. gzip圧縮FASTQ形式ファイル(SRR616268sub_trim2_1.fastq.gzとSRR616268sub_trim2_2.fastq.gz)の場合:

許容するN数(param_nBases)と最短配列長(param_minLength)を明示的に与えるやり方です。tooShortが119、tooManyNが1,762リードあったことがわかります。結果として、998,208リードからなるファイルが出力されています。例題6に比べて若干生き残るリード数が減るのは、param_nBases(デフォルトは2)と param_minLength(デフォルトは14)での条件がデフォルトよりも若干厳しめ (1と18)だからです。

```
in_f1 <- "SRR616268sub_trim2_1.fastq.gz"#入力ファイル名を指定してin_f1に格納
in_f2 <- "SRR616268sub_trim2_2.fastq.gz"#入力ファイル名を指定してin_f2に格納
out_f1 <- "hoge2_1.fastq.gz"            #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge2_2.fastq.gz"            #出力ファイル名を指定してout_f2に格納
param_nBases <- 1                       #許容するNの数を指定
param_minLength <- 18                   #アダプター配列除去後の許容する最短配列長を指定
param_nrec <- 500000                    #一度に処理するリード数を指定

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

#本番(前処理)
res <- preprocessReads(filename=in_f1, #前処理を実行
             filenameMate=in_f2,       #前処理を実行
             outputFilename=out_f1,    #前処理を実行
             outputFilenameMate=out_f2,#前処理を実行
             nBases=param_nBases,      #許容するNの数
             minLength=param_minLength,#アダプター配列除去後
             nrec=param_nrec)          #一度に処理するリード
res                                    #確認してるだけです
```

R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> file.info(list.files(pattern="fastq.gz"))
                                    size
hoge3.fastq.gz                  71542907
hoge4.fastq.gz                  71343605
hoge5.fastq.gz                  63524791
SRR609266.fastq.gz             383840833
SRR616268sub_1.fastq.gz         74906576
SRR616268sub_2.fastq.gz         67158462
SRR616268sub_trim_1.fastq.gz    71227695
SRR616268sub_trim_2.fastq.gz    63442904
SRR616268sub_trim2_1.fastq.gz   71343605
SRR616268sub_trim2_2.fastq.gz   63524791
```

フィルタリング：基本形

リード長が18塩基未満を捨て、1塩基だけNを許容するという閾値でフィルタリングをした結果、998,208リードが残ったということ。約2分

2. gzip圧縮FASTQ形式ファイル(SRR616268sub_trim2_1.fastq.gzとSRR616268sub_trim2_2.fastq.gz)の

許容するN数(param_nBases)と最短配列長(param_minLength)を明示的に与えるやり方です。tooShortが119、tooManyNが1,762リードあったことがわかります。結果として、998,208リードからなるファイルが出力されています。例題6に比べて若干生き残るリード数が減るのは、param_nBases(デフォルトは2)と param_minLength(デフォルトは14)での条件がデフォルトよりも若干厳しめ (1と18)だからです。

```
in_f1 <- "SRR616268sub_trim2_1.fastq.gz"#入力ファイル名を指定してin_f1に格納
in_f2 <- "SRR616268sub_trim2_2.fastq.gz"#入力ファイル名を指定してin_f2に格納
out_f1 <- "hoge2_1.fastq.gz"            #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge2_2.fastq.gz"            #出力ファイル名を指定してout_f2に格納
param_nBases <- 1                       #許容するNの数を指定
param_minLength <- 18                   #アダプター配列除去後の許容する最短配列長を指定
param_nrec <- 500000                    #一度に処理するリード数を指定

#必要なパッケージ
library(QuasR)

#本番(前処理)
res <- preproce
            fi
            ou
            ou
            nB
            mi
            nr
res
```

R Console

```
+                minLength=param_minLength,#アダプター配列除去後の許容する最短$
+                nrec=param_nrec)          #一度に処理するリード数
  filtering .../SRR616268sub_trim2_1.fastq.gz and
    SRR616268sub_trim2_2.fastq.gz
> res                                     #確認してるだけです
                  SRR616268sub_trim2_1.fastq.gz:SRR616268sub_trim2_2.fastq.gz
totalSequences                                                        1000000
matchTo5pAdapter                                                           NA
matchTo3pAdapter                                                           NA
tooShort                                                                  119
tooManyN                                                                 1762
lowComplexity                                                               0
totalPassed                                                            998208
>
```

フィルタリング：発展形

- 前処理 | トリミング | ポリA配列除去 | ShortRead(Morgan 2009) (last modified 2014/06/11)
- 前処理 | トリミング | アダプター配列除去(基礎) | QuasR(Gaidatzis 2015) (last modified 2015/06/24)推奨
- 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)
- 前処理 | トリミング | アダプター配列除去(基礎) | ShortRead(Morgan 2009) (last modified 2014/06/21)
- 前処理 | トリミング | アダプター配列除去(応用) | QuasR(Gaidatzis 2015) (last modified 2015/06/26)推奨
- 前処理 | トリミング | アダプター配列除去(応用) | ShortRead(Morgan 2009) (last modified 2014/06/18)
- 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)
- 前処理 | フィルタリング | PHREDスコアが低い塩基をNに置換 (last modified 2014/03/03)
- 前処理 | フィルタリング | PHREDスコアが低い配列(リード)を除去 (last modified 2014/08/27)
- 前処理 | フィルタリング | ACGTのみからなる配列を抽出 (last modified 2014/08/04)
- 前処理 | フィルタリング | ACGT以外のcharacter"-"をNに変換 (last modified 2013/06/18)
- 前処理 | フィルタリング | ACGT以外の文字数が閾値以下の配列を抽出 (last modified 2013/09/27)
- 前処理 | フィルタリング | 重複のない配列セットを作成 (last modified 2013/06/18)
- 前処理 | フィルタリング | 指定した長さ以上の配列を抽出 (last modified 2014/02/07)
- 前処理 | フィルタリング | 任意のリード(サブセット)を抽出 (last modified 2014/08/21)
- 前処理 | フィルタリング | 指定した長さの範囲の配列を抽出 (last modified 2015/02/26)
- 前処理 | フィルタリング | 任意のIDを含む配列を抽出 (last modified 2013/06/18)
- 前処理 | フィルタリング | Illuminaのpass filtering (last modified 2013/06/19)
- 前処理 | フィルタリング | GFF/GTF形式ファイル (last modified 2013/10/10)
- 前処理 | フィルタリング | 組合せ | ACGTのみ & 指定した長さの範囲の配列 (last modified 2014/06/11)
- 前処理 | フィルタリング | paired-end | 配列長とN数 | QuasR(Gaidatzis 2015) (last modified 2015/06/26)
- 前処理 | フィルタリング | paired-end | 共通リード抽出 | ShortRead(Morgan 2009) (last modified 2015/06/2

①

②

③

④

基本戦略は、paired-endをsingle-endとして独立に取り扱い、最後に共通リードを抽出。例えば①～③あたりの任意のフィルタリングをsingle-endとして独立に実行。最後に④でリード数の異なるpaired-endのファイルを入力として、共通リードを出力。ここでは、①の例題6-8を実行して得られたリード数の異なるpaired-endファイルを入力として、④を実行するやり方を示します。個人見解としては、④の入力ファイルの段階ではFASTQではなくFASTA形式でいいと思います。一連のフィルタリングでそこそこのクオリティが保証されているので、わざわざクオリティ情報を保持しておく必要はないだろうという思想

• 前処理 | トリミング | アダプター配列除去(基礎) | QuasR(Gaidatzis_2015) 参考

フィルタリング：発展形

Small RNA-seqデータのときはリードの右側にあるアダプター配列除去でしたが、乳酸菌RNA-seqデータのときは左側にアダプターがついているのでRpatternではなくLpatternになっている点に注意。

- 前処理 | トリミング | ポリA配列除去 | ShortRead(Morgan_2009) (last modified 2014/06/11)
- 前処理 | トリミング | アダプター配列除去(基礎) | QuasR(Gaidatzis_2015) (last modified 2015/06/24
- 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)
- 前処理 | トリミング | アダプター配列除去(基礎) | ShortRead(Morgan_2009) (last modified 2014/06/21)
- 前処理 | トリミング
- 前処理 | トリミング
- 前処理 | トリミング
- 前処理 | フィルタリ
- 前処理 | フィルタリ
- 前処理 | フィルタリ
- 前処理 | フィルタリ
- 前処理 | フィルタリ
- 前処理 | フィルタリ
- 前処理 | フィルタリ
- 前処理 | フィルタリ
- 前処理 | フィルタリ
- 前処理 | フィルタリ
- 前処理 | フィルタリ
- 前処理 | フィルタリ
- 前処理 | フィルタリ

① ②

前処理 | トリミング | アダプター配列除去(基礎) | QuasR(Gaidatzis_2015)

NEW

QuasRパッケージを
一度に処理するリ
ときに、メモリ不足で
長くなりますが、エラー
さらに小さくして対処し
村浩正 氏提供情報)。
「ファイル」-「ディレクト

1. gzip圧縮FASTQ形

small RNA-seqデー
Genomics, 2013)中の
FASTQ or SRA | SR
ら、おそらくアダプター
測できます。ここでは
(左側)ではなく3'側(
それが、preprocessR

```
in_f <- "SRR609
out_f <- "hoge1
param_adapter <
```

5. gzip圧縮FASTQ形式ファイル(SRR616268sub_1.fastq.gz)の場合:

乳酸菌RNA-seqデータSRR616268の最初の100万リード分です。paired-endデータのforward側、約75MB(74,906,576 bytes)、全リード107 bpです。FastQC実行結果(SRR616268sub_1_fastqc.html)として「TruSeq Adapter, Index 3」が含まれているとレポートされました。これでググると塩基配列情報は "GATCGGAAGAGCACACGTCTGAACTCCAGTCACTTAGGCATCTCGTATGCCGTCTTCTGCTTG" と書いてあったので、これを入力として与えます。アダプター配列の位置は5'側(左側)にあるという前提であり、左側のアダプター配列しかトリムしないやり方です。それが、preprocessReads関数実行時にLpatternのみ記載している理由です。約2分。出力ファイル(998,664リード、71,426,499 bytes)でもう一度FastQCを実行すると、「TruSeq Adapter, Index 3」がOverrepresented sequencesの項目から消えていることまでは確認しました。しかし、まだ「TruSeq Adapter, Index 2」が残っています。

```
in_f <- "SRR616268sub_1.fastq.gz"          #入力ファイル名を指定してin_fに格納
out_f <- "hoge5.fastq.gz"                  #出力ファイル名を指定してout_fに格納
param_adapter <- "GATCGGAAGAGCACACGTCTGAACTCCAGTCACTTAGGCATCTCGTATGCCGTCTTCTGCTTG"#アダプ
param_nrec <- 500000                       #一度に処理するリード数を指定

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

#本番(アダプター配列除去)
res <- preprocessReads(filename=in_f,      #in_fを入力とする
            outputFilename=out_f,          #out_fを出力とする
            Lpattern=param_adapter,        #リードの左側(Left)のアダプターを除去
            nrec=param_nrec)               #一度に処理するリード数
res                                        #確認してるだけです
file.size(in_f)                            #入力ファイルのサイズを表示
file.size(out_f)                           #出力ファイルのサイズを表示
```

フィルタリング：発展形

出力予定ファイルが存在する状態で実行するとエラーが出るので注意。

5. gzip圧縮FASTQ形式ファイル(SRR616268sub_1.fastq.gz)の場合:

乳酸菌RNA-seqデータSRR616268の最初の100万リード分です。paired-endデータのforward側、約75MB(74,906,576 bytes)、全リード107 bpです。FastQC実行結果(SRR616268sub_1_fastqc.html)として「TruSeq Adapter, Index 3」が含まれているとレポートされました。これでググると塩基配列情報は "GATCGGAAGAGCACACGTCTGAACTCCAGTCACTTAGGCATCTCGTATGCCGTCTTCTGCTTG" と書いてあったので、これを入力として与えます。アダプター配列の位置は5'側(左側)にあるという前提であり、左側のアダプター配列しかトリムしないやり方です。それが、preprocessReads関数実行時にLpatternのみ記載している理由です。約2分。出力ファイル(998,664リード、71,426,499 bytes)でもう一度FastQCを実行すると、「TruSeq Adapter, Index 3」がOverrepresented sequencesの項目から消えていることまでは確認しました。しかし、まだ「TruSeq Adapter, Index 2」が残っています。

```
in_f <- "SRR616268sub_1.fastq.gz"       #入力ファイル名を指定してin_
out_f <- "hoge5.fastq.gz"               #出力ファイル名を指定してout
param_adapter <- "GATCGGAAGAGCACACGTCTGAACTCCAGTCACTTAGGCATCTCGTA
param_nrec <- 500000                    #一度に処理するリード数を指

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

#本番(アダプター配列除去)
res <- preprocessReads(filename=in_f,   #in_fを入力とする
            outputFilename=out_f,       #out_fを出力とする
            Lpattern=param_adapter,     #リードの左側(Left)のアダプ
            nrec=param_nrec)            #一度に処理するリード数
res                                     #確認してるだけです
file.size(in_f)                         #入力ファイルのサイズを表示
file.size(out_f)                        #出力ファイルのサイズを表示
```

R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files(pattern="fastq.gz")
 [1] "hoge2_1.fastq.gz"
 [2] "hoge2_2.fastq.gz"
 [3] "hoge3.fastq.gz"
 [4] "hoge4.fastq.gz"
 [5] "hoge5.fastq.gz"
 [6] "SRR609266.fastq.gz"
 [7] "SRR616268sub_1.fastq.gz"
 [8] "SRR616268sub_2.fastq.gz"
 [9] "SRR616268sub_trim_1.fastq.gz"
[10] "SRR616268sub_trim_2.fastq.gz"
[11] "SRR616268sub_trim2_1.fastq.gz"
[12] "SRR616268sub_trim2_2.fastq.gz"
>
```

フィルタリング：発展形

①こんな感じになります。②ここで見えている出力ファイルのサイズは、既に存在していたhoge5.fastq.gzのファイルサイズであり、ここでのコピペ実行結果で得られたものではない点に注意。

5. gzip圧縮FASTQ形式ファイル(SRR616268sub_1.fastq.gz)の場合:

乳酸菌RNA-seqデータSRR616268の最初の100万リード分です。paired-endデータのforward側、約75
bytes)、全リード107 bpです。FastQC実行結果(SRR616268sub_1_fastqc.html)として「TruSeq Adapter
れているとレポートされました。これでググると塩基配列情報は
"GATCGGAAGAGCACACGTCTGAACTCCAGTCACTTAGGCATCTCGTATGCCGTCTTCTGCTTG"と書いて
あったので、これを入力として与えます。
ター配列しかトリムしないやり方です。そ
2分。出力ファイル(998,664リード、71,42
Overrepresented sequencesの項目から消
残っています。

```
in_f <- "SRR616268sub_1.fastq
out_f <- "hoge5.fastq.gz"
param_adapter <- "GATCGGAAG
param_nrec <- 500000

#必要なパッケージをロード
library(QuasR)

#本番(アダプター配列除去)
res <- preprocessReads(filena
             outputFilename=o
             Lpattern=param_a
             nrec=param_nrec)
res
file.size(in_f)
file.size(out_f)
```

R Console

```
> #本番(アダプター配列除去)
> res <- preprocessReads(filename=in_f,   #in_fを入力とする
+                 outputFilename=out_f,   #out_fを出力とする
+                 Lpattern=param_adapter, #リードの左側(Left)のアダ$
+                 nrec=param_nrec)        #一度に処理するリード数
Error in preprocessReads(filename = in_f, outputFilename = out_f, $
  existing output files: hoge5.fastq.gz
> res                                     #確認してるだけです
                 SRR616268sub_trim2_1.fastq.gz:SRR616268sub_trim2_$
totalSequences                                                    $
matchTo5pAdapter                                                  $
matchTo3pAdapter                                                  $
tooShort                                                          $
tooManyN                                                          $
lowComplexity                                                     $
totalPassed                                                       $
> file.size(in_f)                       #入力ファイルのサイズを表$
[1] 74906576
> file.size(out_f)                      #出力ファイルのサイズを表$
[1] 63524791
>
```

①

②

フィルタリング：発展形

hoge5.fastq.gzをhogeフォルダから削除して再度実行。

5. gzip圧縮FASTQ形式ファイル(SRR616268sub_1.fastq.gz)の場合:

乳酸菌RNA-seqデータSRR616268の最初の100万リード分です。paired-endデータのforward側、約75MB(74,906,576 bytes)、全リード107 bpです。FastQC実行結果(SRR616268sub_1_fastqc.html)として「TruSeq Adapter, Index 3」が含まれているとレポートされました。これでググると塩基配列情報は "GATCGGAAGAGCACACGTCTGAACTCCAGTCACTTAGGCATCTCGTATGCCGTCTTCTGCTTG" と書いてあったので、これを入力として与えます。アダプター配列の位置は5'側(左側)にあるという前提であり、左側のアダプター配列しかトリムしないやり方です。それが、preprocessReads関数実行時にLpatternのみ記載している理由です。約2分。出力ファイル(998,664リード、71,426,499 bytes)でもう一度FastQCを実行すると、「TruSeq Adapter, Index 3」がOverrepresented sequencesの項目から消えていることまでは確認しました。しかし、まだ「TruSeq Adapter, Index 2」が残っています。

```
in_f <- "SRR616268sub_1.fastq.gz"       #入力ファイル名を指定して
out_f <- "hoge5.fastq.gz"               #出力ファイル名を指定して
param_adapter <- "GATCGGAAGAGCACACGTCTGAACTCCAGTCACTTAGGCATCTC
param_nrec <- 500000                    #一度に処理するリード数を

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

#本番(アダプター配列除去)
res <- preprocessReads(filename=in_f,   #in_fを入力とする
            outputFilename=out_f,       #out_fを出力とする
            Lpattern=param_adapter,     #リードの左側(Left)のア
            nrec=param_nrec)            #一度に処理するリード数
res                                     #確認してるだけです
file.size(in_f)                         #入力ファイルのサイズを表
file.size(out_f)                        #出力ファイルのサイズを表
```

R Console

```
+                Lpattern=param_adapter,    #$
+                nrec=param_nrec)           #$
  filtering SRR616268sub_1.fastq.gz
> res                                      #$
                    SRR616268sub_1.fastq.gz
totalSequences                      1000000
matchTo5pAdapter                     456561
matchTo3pAdapter                          0
tooShort                                 89
tooManyN                               1247
lowComplexity                             0
totalPassed                          998664
> file.size(in_f)                          #$
[1] 74906576
> file.size(out_f)                         #$
[1] 71426499
>
```

フィルタリング：発展形

得られたhoge5.fastq.gzを入力として、例題7を実行。

7. gzip圧縮FASTQ形式ファイル(hoge5.fastq.gz)の場合:

5.で得られたhoge5.fastq.gzファイル中には、まだ「TruSeq Adapter, Index 2」が残っています。「TruSeq Adapter, Index 2」の塩基配列情報は "GATCGGAAGAGCACACGTCTGAACTCCAGTCACCGATGTATCTCGTATGCCGTCTTCTGCTTG" と書いてあったので、これを入力として与えます。約2分。この出力ファイル(hoge7.fastq.gz)と同じものが SRR616268sub_trim_1.fastq.gz(998,658リード、71,227,695 bytes)です。FastQC実行結果は SRR616268sub_trim_1_fastqc.htmlです。

```
in_f <- "hoge5.fastq.gz"                #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.fastq.gz"               #出力ファイル名を指定してout_fに格納
param_adapter <- "GATCGGAAGAGCACACGTCTGAACTCCAGTCACCGATGTATCTC
param_nrec <- 500000                    #一度に処理するリード数を

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

#本番(アダプター配列除去)
res <- preprocessReads(filename=in_f,   #in_fを入力とする
             outputFilename=out_f,      #out_fを出力とする
             Lpattern=param_adapter,    #リードの左側(Left)のア
             nrec=param_nrec)           #一度に処理するリード数
res                                     #確認してるだけです
file.size(in_f)                         #入力ファイルのサイズを表
file.size(out_f)                        #出力ファイルのサイズを表
```

R Console

```
+                Lpattern=param_adapter,    #$
+                nrec=param_nrec)           #$
  filtering hoge5.fastq.gz
> res                                      #$
                 hoge5.fastq.gz
totalSequences           998664
matchTo5pAdapter         170144
matchTo3pAdapter              0
tooShort                      6
tooManyN                      0
lowComplexity                 0
totalPassed              998658
> file.size(in_f)                          #$
[1] 71426499
> file.size(out_f)                         #$
[1] 71227695
>
```

フィルタリング：発展形

例題8のpaired-end reverse側も実行。出力ファイルのサイズは63,442,904 bytes、999,136リード

8. gzip圧縮FASTQ形式ファイル(SRR616268sub_2.fastq.gz)の場合:

乳酸菌RNA-seqデータSRR616268の最初の100万リード分です。paired-endデータのreverse側、約67MB、全リード93 bpです。FastQC実行結果(SRR616268sub_2_fastqc.html)として「Illumina Single End PCR Primer 1」が含まれているとレポートされました。これでググると塩基配列情報は "AATGATACGGCGACCACCGAGATCTACACTCTTTCCCTACACGACGCTCTTCCGATCT" と書いてあったので、これを入力として与えます。アダプター配列の位置は5'側(左側)にあるという前提であり、左側のアダプター配列しかトリムしないやり方です。それが、preprocessReads関数実行時にLpatternのみ記載している理由です。約2分。この出力ファイル(hoge8.fastq.gz)と同じものが SRR616268sub_trim_2.fastq.gz (999,136リード、63,442,904 bytes)です。FastQC実行結果はSRR616268sub_trim_2_fastqc.htmlです。

```
in_f <- "SRR616268sub_2.fastq.gz"       #入力ファイル名を指定して
out_f <- "hoge8.fastq.gz"               #出力ファイル名を指定して
param_adapter <- "AATGATACGGCGACCACCGAGATCTACACTCTTTCCCTACACGA
param_nrec <- 500000                    #一度に処理するリード数を


#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(Biostrings)                     #パッケージの読み込み

#前処理(reverse complementの作成)
hoge <- reverseComplement(DNAString(param_adapter))#逆相補鎖(r

#本番(アダプター配列除去)
res <- preprocessReads(filename=in_f,   #in_fを入力とする
            outputFilename=out_f,       #out_fを出力とする
            Lpattern=as.character(hoge),#リードの左側(Left)の
            nrec=param_nrec)            #一度に処理するリード数
res                                     #確認してるだけです
file.size(in_f)                         #入力ファイルのサイズを表
file.size(out_f)                        #出力ファイルのサイズを表
```

```
R Console
+                Lpattern=as.character(hoge)$
+                nrec=param_nrec)           #$
  filtering SRR616268sub_2.fastq.gz
> res                                      #$
                  SRR616268sub_2.fastq.gz
totalSequences                    1000000
matchTo5pAdapter                   273793
matchTo3pAdapter                        0
tooShort                               17
tooManyN                              847
lowComplexity                           0
totalPassed                        999136
> file.size(in_f)                        #$
[1] 67158462
> file.size(out_f)                       #$
[1] 63442904
>
```

乳酸菌RNA-seqデータ

①リード数の異なるpaired-endのファイルを入力として、共通リードを出力するやり方を示します。

- 100万リードのオリジナル?!ファイル
  - SRR616268sub_1.fastq.gz : 107 bp, 74,906,576 bytes
  - SRR616268sub_2.fastq.gz : 93 bp, 67,158,462 bytes
- 100万リードのアダプター除去後のファイル
  - SRR616268sub_trim2_1.fastq.gz(hoge4.fastq.gz): 71,343,605 bytes
  - SRR616268sub_trim2_2.fastq.gz : 63,524,791 bytes
  - hoge2_1.fastq.gz: 998,208リード、71,194,586 bytes
  - hoge2_2.fastq.gz: 998,208リード、63,375,473 bytes
- アダプター除去およびフィルタリング後のファイル
  - SRR616268sub_trim_1.fastq.gz(hoge7.fastq.gz): 998,658リード、71,227,695 bytes
  - SRR616268sub_trim_2.fastq.gz(hoge8.fastq.gz): 999,136リード、63,442,904 bytes

Tips：file.info関数

file.info関数でもファイル名、サイズなど様々な情報をみることができます。

- 100万リードのオリジナル?!ファイル
  - SRR616268sub_1.fastq.gz：107 bp, 74,906,576 bytes
  - SRR616268sub_2.fastq.gz：93 bp, 67,158,462 bytes
- 100万リードのアダプター除去後のファイル
  - SRR616268sub_trim2_1.fastq.gz
  - SRR616268sub_trim2_2.fastq.gz
  - hoge2_1.fastq.gz: 998,208リード
  - hoge2_2.fastq.gz: 998,208リード
- アダプター除去およびフィルタ
  - SRR616268sub_trim_1.fastq.gz(
  - SRR616268sub_trim_2.fastq.gz(

R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> file.info(list.files(pattern="fastq.gz"))
                                 size isdir mode
hoge2_1.fastq.gz             71194586 FALSE  666
hoge2_2.fastq.gz             63375473 FALSE  666
hoge3.fastq.gz               71542907 FALSE  666
hoge4.fastq.gz               71343605 FALSE  666
hoge5.fastq.gz               71426499 FALSE  666
hoge7.fastq.gz               71227695 FALSE  666
hoge8.fastq.gz               63442904 FALSE  666
SRR609266.fastq.gz          383840833 FALSE  666
SRR616268sub_1.fastq.gz      74906576 FALSE  666
SRR616268sub_2.fastq.gz      67158462 FALSE  666
SRR616268sub_trim_1.fastq.gz 71227695 FALSE  666
SRR616268sub_trim_2.fastq.gz 63442904 FALSE  666
SRR616268sub_trim2_1.fastq.gz 71343605 FALSE  666
SRR616268sub_trim2_2.fastq.gz 63524791 FALSE  666
```

フィルタリング：発展形

入力はリード数の異なるpaired-endファイル。出力はリード数が同じpaired-endファイルです。description行の記述形式次第ではエラーが出ると思いますが、適宜変更してください。

① ←

前処理 | フィルタリング | paired-end | 共通リード抽出 | ShortRead(Morgan_2009)

NEW

ShortReadパッケージを用いて、ファイル内での配列長とリード数が異なるpaired-endデータの2つのファイルを入力として、両方に共通して存在するリードのみ抽出するやり方を示します。QuasRは入力ファイルのリード数が揃ってないと受けつけないので、ShortReadパッケージで別々に読み込む戦略を採用しています。

1. gzip圧縮FASTQ形式ファイル(SRR616268sub_trim_1.fastq.gzとSRR616268sub_trim_2.fastq.gz)の場合:

前処理 | トリミング | アダプター配列除去(基礎) | QuasR(Gaidatzis_2015) の例題7と8の実行結果ファイルで、100万リード弱のpaired-endデータです。SRR616268sub_trim_1.fastq.gzは、998,658リード、71,227,695 bytes (約71MB)です。SRR616268sub_trim_2.fastq.gzは、999,136リード、63,442,904 bytes (約63MB)です。前処理のところで長ったらしい作業をしているのは、FASTQファイル中のdescription行でリードのIDの対応付けがそのままではできないので、対応付け可能な部分文字列を抽出しているためです。イントロ | NGS | 読み込み | FASTQ形式 | description行の記述を整形のコードをテンプレートにしています。共通リードは998,428個です。リード数がほとんど減っていないにもかかわらず、ファイルサイズが(71,227,695 bytes -> 63,446,240 bytes; 63,442,904 bytes -> 56,019,410 bytes)と大幅に減っているのは、description部分の記述内容が減っていることに起因します。

```
in_f1 <- "SRR616268sub_trim_1.fastq.gz"#入力ファイル名を指定してin_f1に格納
in_f2 <- "SRR616268sub_trim_2.fastq.gz"#入力ファイル名を指定してin_f2に格納
out_f1 <- "hoge1_1.fastq.gz"           #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge1_2.fastq.gz"           #出力ファイル名を指定してout_f2に格納

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

#入力ファイルの読み込み
fastq1 <- readFastq(in_f1)             #in_f1で指定したファイルの読み込み
fastq2 <- readFastq(in_f2)             #in_f2で指定したファイルの読み込み
head(id(fastq1), n=4)                  #description部分を表示
head(id(fastq2), n=4)                  #description部分を表示
```

フィルタリング：発展形

最初の4リード分のdescription部分を表示。ここで見えているもので判断すると、SRR616268.7は2つの入力ファイル中に存在する。そして、少なくともforward側のファイル(*_1.fastq.gz)中の3リード(SRR616268.20-22)は、reverse側のファイル(*_2.fastq.gz)中には存在しないことがわかる。

1. gzip圧縮FASTQ形式ファイル(SRR616268sub_trim_1.fastq.gzとSRR616268sub_trim_2.

前処理 | トリミング | アダプター配列除去(基礎) | QuasR(Gaidatzis_2015)の例題7と8の実行
100万リード弱のpaired-endデータです。SRR616268sub_trim_1.fastq.gzは、998,658リード、
(約71MB)です。SRR616268sub_trim_2.fastq.gzは、999,136リード、63,442,904 bytes (約63M
のところで長ったらしい作業をしているのは、FASTQファイル中のdescription行でリードのID
のままではできないので、対応付け可能な部分文字列を抽出しているためです。イントロ |
FASTQ形式 | description行の記述を整形のコードをテンプレートにしています。共通リード
リード数がほとんど減っていないにもかかわらず、ファイルサイズが(71,227,695 bytes -> 63,446,240 bytes;
63,442,904 bytes -> 56,019,410 bytes)と大幅に減っているのは、de
に起因します。

```
in_f1 <- "SRR616268sub_trim_1.fastq.gz"#入力ファイ
in_f2 <- "SRR616268sub_trim_2.fastq.gz"#入力ファイ
out_f1 <- "hoge1_1.fastq.gz"           #出力ファイ
out_f2 <- "hoge1_2.fastq.gz"           #出力ファイ

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

#入力ファイルの読み込み
fastq1 <- readFastq(in_f1)             #in_f1で指
fastq2 <- readFastq(in_f2)             #in_f2で指
head(id(fastq1), n=4)                  #descripti
head(id(fastq2), n=4)                  #descripti

#前処理(FASTQのdescription行を整形; in_f1)
fastq <- fastq1                        #fastqとし
hoge <- strsplit(as.character(id(fastq)), " ", fi
description <- BStringSet(sapply(hoge,"[[", 1))#
fastq <- ShortReadQ(sread(fastq), quality(fastq),
head(id(fastq), n=4)                   #descripti
fastq1 <- fastq                        #元に戻す
```

R Console

```
> #入力ファイルの読み込み
> fastq1 <- readFastq(in_f1)                    #in_f1で指$
> fastq2 <- readFastq(in_f2)                    #in_f2で指$
> head(id(fastq1), n=4)                         #descripti$
  A BStringSet instance of length 4
    width seq
[1]    44 SRR616268.7 2291:6:...412:2249 length=107
[2]    45 SRR616268.20 2291:6...720:2221 length=107
[3]    45 SRR616268.21 2291:6...652:2235 length=107
[4]    45 SRR616268.22 2291:6...505:2236 length=107
> head(id(fastq2), n=4)                         #descripti$
  A BStringSet instance of length 4
    width seq
[1]    43 SRR616268.7 2291:6:...1412:2249 length=93
[2]    44 SRR616268.48 2291:6...2243:2242 length=93
[3]    44 SRR616268.61 2291:6...2360:2248 length=93
[4]    44 SRR616268.73 2291:6...2584:2248 length=93
>
```

フィルタリング：発展形

共通リード抽出用の基本情報として、リードIDの積集合(intersection)を得るのが基本戦略。しかし、description部分の情報をそのまま使うと、length=107とlength=93という部分文字列の違いにより、①同一リードを同一文字列として認識できないという問題がある

1. gzip圧縮FASTQ形式ファイル(SRR616268sub_trim_1.fastq.gzとSRR616268sub_trim_2.

前処理 | トリミング | アダプター配列除去(基礎) | QuasR(Gaidatzis_2015) の例題7と8の実行
100万リード弱のpaired-endデータです。SRR616268sub_trim_1.fastq.gzは、998,658リード、
(約71MB)です。SRR616268sub_trim_2.fastq.gzは、999,136リード、63,442,904 bytes (約63M
のところで長ったらしい作業をしているのは、FASTQファイル中のdescription行でリードのID
のままではできないので、対応付け可能な部分文字列を抽出しているためです。イントロ | NGS | 読み込み |
FASTQ形式 | description行の記述を整形のコードをテンプレートにしています。共通リードは998,428個です。
リード数がほとんど減っていないにもかかわらず、ファイルサイズが
63,442,904 bytes -> 56,019,410 bytes)と大幅に減っているのは、de
に起因します。

```
in_f1 <- "SRR616268sub_trim_1.fastq.gz"#入力ファイ
in_f2 <- "SRR616268sub_trim_2.fastq.gz"#入力ファイ
out_f1 <- "hoge1_1.fastq.gz"           #出力ファイ
out_f2 <- "hoge1_2.fastq.gz"           #出力ファイ

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

#入力ファイルの読み込み
fastq1 <- readFastq(in_f1)             #in_f1で指
fastq2 <- readFastq(in_f2)             #in_f2で指
head(id(fastq1), n=4)                  #descripti
head(id(fastq2), n=4)                  #descripti

#前処理(FASTQのdescription行を整形; in_f1)
fastq <- fastq1                        #fastqとし
hoge <- strsplit(as.character(id(fastq)), " ", fi
description <- BStringSet(sapply(hoge,"[[", 1))#h
fastq <- ShortReadQ(sread(fastq), quality(fastq
head(id(fastq), n=4)                   #descri
fastq1 <- fastq                        #元に戻
```

①

R Console

```
> id(fastq1)[1:4]
  A BStringSet instance of length 4
    width seq
[1]    44 SRR616268.7 2291:6:1...412:2249 length=107
[2]    45 SRR616268.20 2291:6:...720:2221 length=107
[3]    45 SRR616268.21 2291:6:...652:2235 length=107
[4]    45 SRR616268.22 2291:6:...505:2236 length=107
> id(fastq2)[1:4]
  A BStringSet instance of length 4
    width seq
[1]    43 SRR616268.7 2291:6:1...1412:2249 length=93
[2]    44 SRR616268.48 2291:6:...2243:2242 length=93
[3]    44 SRR616268.61 2291:6:...2360:2248 length=93
[4]    44 SRR616268.73 2291:6:...2584:2248 length=93
> intersect(c("AA", "B", "C"), c("A", "C"))
[1] "C"
> intersect(id(fastq1), id(fastq2))
  A BStringSet instance of length 0
>
```

フィルタリング：発展形

description部分を眺め、部分文字列で同一リードを認識できる箇所を探す。ここでは、①description部分を区切り文字「スペース(“ ”)で」分断し、②1番目の要素を取り出せば、赤下線部分のみの部分文字列で評価できるという戦略のもとでプログラムを組んでいる。

```
#前処理(FASTQのdescription行を整形; in_f1)
fastq <- fastq1                                  #fastqとして取り扱う
hoge <- strsplit(as.character(id(fastq)), " ", fixed=TRUE)#id(fastq)
description <- BStringSet(sapply(hoge,"[[", 1))#hogeのリスト中の1番目の
fastq <- ShortReadQ(sread(fastq), quality(fastq), description)#ReadFa
head(id(fastq), n=4)                             #description部分を表示
fastq1 <- fastq                                  #元に戻す

#前処理(FASTQのdescription行を整形; in_f2)
fastq <- fastq2                                  #fastqとして
hoge <- strsplit(as.character(id(fastq)), " ", fix
description <- BStringSet(sapply(hoge,"[[", 1))#ho
fastq <- ShortReadQ(sread(fastq), quality(fastq),
head(id(fastq), n=4)                             #descriptio
fastq2 <- fastq                                  #元に戻す

#本番
common <- intersect(id(fastq1), id(fastq2))#共通し
length(fastq1)                                   #fastq1のリ
length(fastq2)                                   #fastq2のリ
length(common)                                   #共通リード数
obj1 <- is.element(as.character(id(fastq1)), as.ch
```

①

②

R Console

```
> #入力ファイルの読み込み
> fastq1 <- readFastq(in_f1)                     #in_f1で指$
> fastq2 <- readFastq(in_f2)                     #in_f2で指$
> head(id(fastq1), n=4)                          #descripti$
  A BStringSet instance of length 4
    width seq
[1]    44 SRR616268.7 2291:6:...412:2249 length=107
[2]    45 SRR616268.20 2291:6...720:2221 length=107
[3]    45 SRR616268.21 2291:6...652:2235 length=107
[4]    45 SRR616268.22 2291:6...505:2236 length=107
> head(id(fastq2), n=4)                          #descripti$
  A BStringSet instance of length 4
    width seq
[1]    43 SRR616268.7 2291:6:...1412:2249 length=93
[2]    44 SRR616268.48 2291:6...2243:2242 length=93
[3]    44 SRR616268.61 2291:6...2360:2248 length=93
[4]    44 SRR616268.73 2291:6...2584:2248 length=93
>
```

フィルタリング：発展形

Forward側。うまくリードID(SRR6161268.7)のみにできていることがわかります

```
#前処理(FASTQのdescription行を整形; in_f1)
fastq <- fastq1                                  #fastqとして取り扱う
hoge <- strsplit(as.character(id(fastq)), " ", fixed=TRUE)#id(fastq)中の文字列を" "で区切っ
description <- BStringSet(sapply(hoge,"[[", 1))#hogeのリスト中の1番目の要素のみ取り出してdesc
fastq <- ShortReadQ(sread(fastq), quality(fastq), description)#ReadFastQ関数を用いてReadFas
head(id(fastq), n=4)                             #description部分を表示
fastq1 <- fastq                                  #元に戻す

#前処理(FASTQのdescription行を整形; in_f2)
fastq <- fastq2                                  #fastqとして
hoge <- strsplit(as.character(id(fastq)), " ", fix
description <- BStringSet(sapply(hoge,"[[", 1))#ho
fastq <- ShortReadQ(sread(fastq), quality(fastq),
head(id(fastq), n=4)                             #descriptio
fastq2 <- fastq                                  #元に戻す

#本番
common <- intersect(id(fastq1), id(fastq2))#共通し
length(fastq1)                                   #fastq1のリ
length(fastq2)                                   #fastq2のリ
length(common)                                   #共通リード数
obj1 <- is.element(as.character(id(fastq1)), as.ch
```

R Console

```
> head(id(fastq1), n=4)
  A BStringSet instance of length 4
    width seq
[1]    44 SRR616268.7 2291:6:1...412:2249 length=107
[2]    45 SRR616268.20 2291:6:...720:2221 length=107
[3]    45 SRR616268.21 2291:6:...652:2235 length=107
[4]    45 SRR616268.22 2291:6:...505:2236 length=107
> fastq <- fastq1                                #fastqとし$
> hoge <- strsplit(as.character(id(fastq)), " ", fi$
> description <- BStringSet(sapply(hoge,"[[", 1))#h$
> fastq <- ShortReadQ(sread(fastq), quality(fastq),$
> head(id(fastq), n=4)                           #descripti$
  A BStringSet instance of length 4
    width seq
[1]    11 SRR616268.7
[2]    12 SRR616268.20
[3]    12 SRR616268.21
[4]    12 SRR616268.22
> fastq1 <- fastq                                #元に戻す
>
```

フィルタリング：発展形

Reverse側。うまくリードID(SRR6161268.7)のみにできていることがわかります

```
#前処理(FASTQのdescription行を整形; in_f1)
fastq <- fastq1                                  #fastqとして取り扱う
hoge <- strsplit(as.character(id(fastq)), " ", fixed=TRUE)#id(fastq)中の文字列を" "で区切っ
description <- BStringSet(sapply(hoge,"[[", 1))#hogeのリスト中の1番目の要素のみ取り出してdesc
fastq <- ShortReadQ(sread(fastq), quality(fastq), description)#ReadFastQ関数を用いてReadFas
head(id(fastq), n=4)                             #description部分を表示
fastq1 <- fastq                                  #元に戻す

#前処理(FASTQのdescription行を整形; in_f2)
fastq <- fastq2                                  #fastqとして
hoge <- strsplit(as.character(id(fastq)), " ", fix
description <- BStringSet(sapply(hoge,"[[", 1))#ho
fastq <- ShortReadQ(sread(fastq), quality(fastq),
head(id(fastq), n=4)                             #descriptio
fastq2 <- fastq                                  #元に戻す

#本番
common <- intersect(id(fastq1), id(fastq2))#共通し
length(fastq1)                                   #fastq1のリ
length(fastq2)                                   #fastq2のリ
length(common)                                   #共通リード数
obj1 <- is.element(as.character(id(fastq1)), as.ch
```

R Console

```
> head(id(fastq2), n=4)
  A BStringSet instance of length 4
    width seq
[1]    43 SRR616268.7 2291:6:1...1412:2249 length=93
[2]    44 SRR616268.48 2291:6:...2243:2242 length=93
[3]    44 SRR616268.61 2291:6:...2360:2248 length=93
[4]    44 SRR616268.73 2291:6:...2584:2248 length=93
> fastq <- fastq2                                #fastqとし$
> hoge <- strsplit(as.character(id(fastq)), " ", fi$
> description <- BStringSet(sapply(hoge,"[[", 1))#h$
> fastq <- ShortReadQ(sread(fastq), quality(fastq),$
> head(id(fastq), n=4)                           #descripti$
  A BStringSet instance of length 4
    width seq
[1]    11 SRR616268.7
[2]    12 SRR616268.48
[3]    12 SRR616268.61
[4]    12 SRR616268.73
> fastq2 <- fastq                                #元に戻す
>
```

• 前処理 | フィルタリング | paired-end | 共通リード抽出 | ShortRead(Morgan 2009) 参考

Tips：コピペで使いまわす

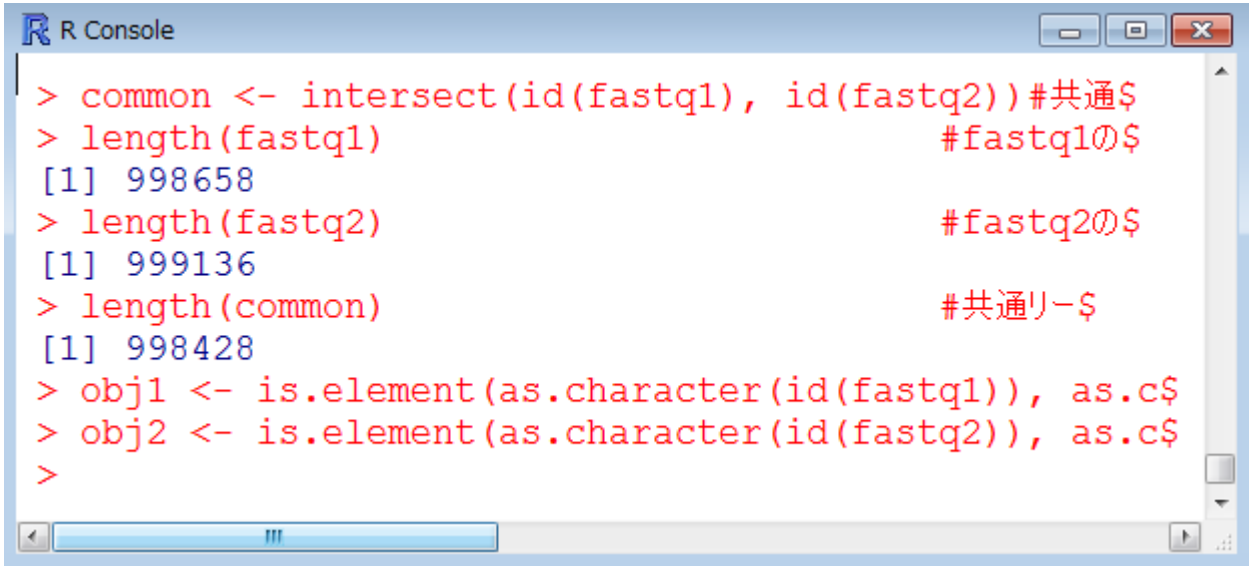
わざわざ別のオブジェクト名に置き換えているので、一見ややこしいですが、黒枠内のコードを使いまわせるので、間違いを減らせるというメリットがあります。

```
#前処理(FASTQのdescription行を整形; in_f1)
fastq <- fastq1                          #fastqとして取り扱う
hoge <- strsplit(as.character(id(fastq)), " ", fixed=TRUE)#id(fastq)中の文字列を" "で区
description <- BStringSet(sapply(hoge,"[[", 1))#hogeのリスト中の1番目の要素のみ取り出してdesc
fastq <- ShortReadQ(sread(fastq), quality(fastq), description)#ReadFastQ関数を用いてReadFas
head(id(fastq), n=4)                     #description部分を表示
fastq1 <- fastq                          #元に戻す

#前処理(FASTQのdescription行を整形; in_f2)
fastq <- fastq2                          #fastqとして取り扱う
hoge <- strsplit(as.character(id(fastq)), " ", fixed=TRUE)#id(fastq)中の文字列を" "で区切っ
description <- BStringSet(sapply(hoge,"[[", 1))#hogeのリスト中の1番目の要素のみ取り出してdesc
fastq <- ShortReadQ(sread(fastq), quality(fastq), description)#ReadFastQ関数を用いてReadFas
head(id(fastq), n=4)                     #description部分を表示
fastq2 <- fastq                          #元に戻す

#本番
common <- intersect(id(fastq1), id(fastq2))#共通して現れるリードID情報を抽出
length(fastq1)                           #fastq1のリード数を表示
length(fastq2)                           #fastq2のリード数を表示
length(common)                           #共通リード数を表示
obj1 <- is.element(as.character(id(fastq1)), as.character(common))#共通リードの位置をTRUE、
```

フィルタリング：発展形

```
#本番
common <- intersect(id(fastq1), id(fastq2))#共通して現れるリードID情報を抽出
length(fastq1)                              #fastq1のリード数を表示
length(fastq2)                              #fastq2のリード数を表示
length(common)                              #共通リード数を表示
obj1 <- is.element(as.character(id(fastq1)), as.character(common))#共通リードの位置をTRUE、
obj2 <- is.element(as.character(id(fastq2)), as.character(common))#共通リードの位置をTRUE、

#ファイルに保存
writeFastq(fastq1[obj1], out_f1, compress=T)#fastqの中身を指定したファイル名で保存
writeFastq(fastq2[obj2], out_f2, compress=T)#fastqの中身を指定したファイル名で保存
file.size(in_f1)                            #入力ファイルのサイズを表示
file.size(in_f2)                            #入力ファイルのサイズを表示
file.size(out_f1)                           #出力ファイルのサイズを表示
file.size(out_f2)                           #出力ファイルのサイズを表示
```

得られた共通リード数(998,428)は、forward側(998,658)およびreverse側(999,136)より少ないので妥当。



フィルタリング：発展形

fastq1オブジェクトの1, 13, 17, 22番目の要素が共通。

```
#本番
common <- intersect(id(fastq1), id(fastq2))#共通して現れるリードID情報を抽出
length(fastq1)                              #fastq1のリード数を表示
length(fastq2)                              #fastq2のリード数を表示
length(common)                              #共通リード数を表示
obj1 <- is.element(as.character(id(fastq1)), as.character(common))#共通リードの位置をTRUE、
obj2 <- is.element(as.character(id(fastq2)), as.character(common))#共通リードの位置をTRUE、

#ファイルに保存
writeFastq(fastq1[obj1], out_f1, compress=T)#fastq
writeFastq(fastq2[obj2], out_f2, compress=T)#fastq
file.size(in_f1)                            #入力ファイ
file.size(in_f2)                            #入力ファイ
file.size(out_f1)                           #出力ファイ
file.size(out_f2)                           #出力ファイ
```

R Console

```
> as.character(id(fastq1))[1:24]
 [1] "SRR616268.7"  "SRR616268.20" "SRR616268.21"
 [4] "SRR616268.22" "SRR616268.30" "SRR616268.31"
 [7] "SRR616268.32" "SRR616268.33" "SRR616268.44"
[10] "SRR616268.45" "SRR616268.46" "SRR616268.47"
[13] "SRR616268.48" "SRR616268.58" "SRR616268.59"
[16] "SRR616268.60" "SRR616268.61" "SRR616268.69"
[19] "SRR616268.70" "SRR616268.71" "SRR616268.72"
[22] "SRR616268.73" "SRR616268.86" "SRR616268.87"
> as.character(id(fastq2))[1:9]
[1] "SRR616268.7"   "SRR616268.48"  "SRR616268.61"
[4] "SRR616268.73"  "SRR616268.101" "SRR616268.117"
[7] "SRR616268.140" "SRR616268.141" "SRR616268.154"
> as.character(common)[1:9]
[1] "SRR616268.7"   "SRR616268.48"  "SRR616268.61"
[4] "SRR616268.73"  "SRR616268.101" "SRR616268.117"
[7] "SRR616268.140" "SRR616268.141" "SRR616268.154"
>
```

フィルタリング：発展形

```
#本番
common <- intersect(id(fastq1), id(fastq2))#共通して現れるリードID情報を抽出
length(fastq1)                             #fastq1のリード数を表示
length(fastq2)                             #fastq2のリード数を表示
length(common)                             #共通リード数を表示
obj1 <- is.element(as.character(id(fastq1)), as.character(common))#共通リードの位置をTRUE、
obj2 <- is.element(as.character(id(fastq2)), as.character(common))#共通リードの位置をTRUE、

#ファイルに保存
writeFastq(fastq1[obj1], out_f1, compress=T)#fastq
writeFastq(fastq2[obj2], out_f2, compress=T)#fastq
file.size(in_f1)                           #入力ファイ
file.size(in_f2)                           #入力ファイ
file.size(out_f1)                          #出力ファイ
file.size(out_f2)                          #出力ファイ
```

is.element関数は、右側の集合の中の要素が存在する位置をTRUE、それ以外をFALSEとして返す。共通であるfastq1オブジェクトの1, 13, 17, 22番目の要素がTRUEになっていることがわかる。

R Console

```
> is.element(c(3, 8, 2, 4), c(4, 8))
[1] FALSE  TRUE FALSE  TRUE
> is.element(c(3, 8, 2, 4), c(4, 8, 5))
[1] FALSE  TRUE FALSE  TRUE
> obj1[1:27]
 [1]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [9] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
[17]  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
[25] FALSE FALSE FALSE
> obj2[1:27]
 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[10] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[19] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
> sum(obj1)
[1] 998428
> sum(obj2)
[1] 998428
>
```

フィルタリング：発展形

共通リードのみなので、リード数が揃っていることがわかる。

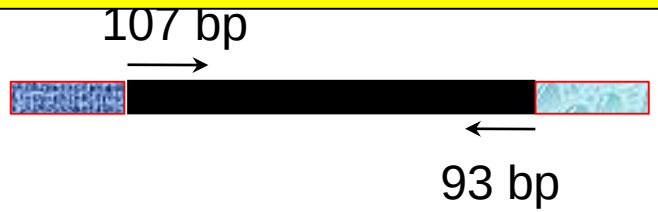
```
#本番
common <- intersect(id(fastq1), id(fastq2))#共通して現れるリードID情報を抽出
length(fastq1)                              #fastq1のリード数を表示
length(fastq2)                              #fastq2のリ
length(common)                              #共通リード数
obj1 <- is.element(as.character(id(fastq1)), as.ch
obj2 <- is.element(as.character(id(fastq2)), as.ch

#ファイルに保存
writeFastq(fastq1[obj1], out_f1, compress=T)#fastq
writeFastq(fastq2[obj2], out_f2, compress=T)#fastq
file.size(in_f1)                            #入力ファイ
file.size(in_f2)                            #入力ファイ
file.size(out_f1)                           #出力ファイ
file.size(out_f2)                           #出力ファイ
```

R Console

```
> fastq1
class: ShortReadQ
length: 998658 reads; width: 15..107 cycles
> fastq2
class: ShortReadQ
length: 999136 reads; width: 14..93 cycles
> fastq1[obj1]
class: ShortReadQ
length: 998428 reads; width: 15..107 cycles
> fastq2[obj2]
class: ShortReadQ
length: 998428 reads; width: 14..93 cycles
> writeFastq(fastq1[obj1], out_f1, compress=T)#fast$
> writeFastq(fastq2[obj2], out_f2, compress=T)#fast$
> file.size(in_f1)                            #入力ファ$
[1] 71227695
> file.size(in_f2)                            #入力ファ$
[1] 63442904
> file.size(out_f1)                           #出力ファ$
[1] 63446240
> file.size(out_f2)                           #出力ファ$
[1] 56019410
>
```

乳酸菌RNA-seqデータ

①はデフォルトの「許容するN数が2、最短配列長が14」のときの結果。②は若干厳しめの「許容するN数が1、最短配列長が18」のときの結果なので、②のほうがリード数が若干少ない。



- 100万リードのオリジナル?!ファイル
  - SRR616268sub_1.fastq.gz : 107 bp, 74,906,576 bytes
  - SRR616268sub_2.fastq.gz : 93 bp, 67,158,462 bytes
- 100万リードのアダプター除去後のファイル
  - SRR616268sub_trim2_1.fastq.gz(hoge4.fastq.gz): 71,343,605 bytes
  - SRR616268sub_trim2_2.fastq.gz : 63,524,791 bytes
  - hoge2_1.fastq.gz: 998,208リード、71,194,586 bytes
  - hoge2_2.fastq.gz: 998,208リード、63,375,473 bytes

②

- アダプター除去およびフィルタリング後のファイル
  - SRR616268sub_trim_1.fastq.gz(hoge7.fastq.gz): 998,658リード、71,227,695 bytes
  - SRR616268sub_trim_2.fastq.gz(hoge8.fastq.gz): 999,136リード、63,442,904 bytes
  - hoge1_1.fastq.gz: 998,428リード、63,446,240 bytes
  - hoge1_2.fastq.gz: 998,428リード、56,019,410 bytes

①

Contents

- 先週の課題やエラーについて
  - 課題1の解説
  - アダプター配列除去（QuasR）実行時のエラーは門田のミス
  - Paired-endデータの取り扱い
- フィルタリング(filtering)
  - 基本形：リード数が同じpaired-endの場合（QuasRパッケージ）
  - 発展形：リード数が異なる場合（ShortReadパッケージ）
- アセンブル(assemble)
  - ゲノム用とトランスクリプトーム用
  - Rockhopper2（バクテリアのトランスクリプトーム用）
- マッピング（mapping）
  - 基礎、オプション、仮想データ説明
  - マッピング本番、出力ファイル形式、オプションと結果の関係
  - カウント情報取得
- 実データ解析
  - QuasRでマッピング → トリミング → 高精度なアセンブルとマッピングへ

アセンブル:ゲノム用

ゲノム用。②非モデル生物やヘテロ接合度の高い生物種用、③微生物など小～中規模ゲノム配列決定用。アセンブルのアルゴリズム周辺は「2014.06.25」でwebページ内を検索

- 前処理 | フィルタリング | paired-end | 配列長とN数 | QuasR(Gaidatzis 2015) (last mo
- 前処理 | フィルタリング | paired-end | 共通リード抽出 | ShortRead(Morgan 2009) (las
- アセンブル | について (
- アセンブル | ゲノム用 ①
- アセンブル | トランスクリプト
- マッピング | について (last m
- マッピング | basic aligner (las
- マッピング | splice-aware alig
- マッピング | Bisulfite sequen
- マッピング | (ESTレベルの長
- マッピング | 基礎 (last modif
- マッピング | single-end | ゲノ
- マッピング | single-end | ゲノ
- マッピング | single-end | ゲノ

アセンブル | ゲノム用

Rパッケージはおそらくありません。

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,
- 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., 200
- CABOG: Miller et al., Bioinformatics, 2008
- EULER-SR: Chaisson et al., Genome Res., 200
- SHORTY: Hossain et al., BMC Bioinformatics
- QSRA: Bryant et al., BMC Bioinformatics, 200
- 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 B
- SOAPdenovo: Li et al., Genome Res., 2010
- SGA: Simpson et al., Genome Res., 2012
- Mapsembler: Peterlongo and Chikhi, BMC Bio
- IDBA-UD: Peng et al., Bioinformatics, 2012
- SPAdes: Bankevich et al., J Comput Biol., 201

- SparseAssembler: Ye et al., BMC Bioinformatics, 2012
- Readjoiner: Gonnella et al., BMC Bioinformatics, 2012
- Fermi: Li H., Bioinformatics, 2012
- MetaVelvet(メタゲノム用): Namiki et al., Nucleic Acids Res., 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 ②
- Omega(メタゲノム用): Haider et al., Bioinformatics, 2014

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 ③
- iMetAMOS(パイプライン): Koren et al., BMC Bioinformatics, 2014
- MyPro(パイプライン; Prokaryote用): Liao et al., J Microbiol Methods., 2015

転写物(トランスクリプトーム)用

アセンブル:転写物用

- アセンブル | について (last modified 2014/06/20)
- アセンブル | ゲノム用 (last modified 2015/05/1
- アセンブル | トランスクリプトーム(転写物)用 (last modified 2015/04/22)
- マッピング | について (last modified 2015/01/1
- マッピング | basic aligner (last modified 2014/08/08)
- マッピング | splice-aware aligner (last modified 2015/06/02)
- マッピング | Bisulfite sequencing用 (last modified 2014/07/09)
- マッピング | (ESTレベルの長さの)contig (last modified 2014/06/24)
- 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(Gaidatzis_201
- マッピング | single-end | ゲノム | basic aligner(応用) | QuasR(Gaidatzis_201
- マッピング | single-end | ゲノム | splice-aware aligner | QuasR(Gaidatzis_20
- マップ後 | について (last modified 2013/06/19)
- マップ後 | 出力ファイル形式について (last modified 2013/11/05)
- マップ後 | 出力ファイルの読み込み | BAM形式 (last modified 2014/06/21)
- マップ後 | 出力ファイルの読み込み | Bowtie形式 (last modified 2013/06/1
- マップ後 | 出力ファイルの読み込み | SOAP形式 (last modified 2013/06/19

① (赤矢印)

アセンブル | トランスクリプトーム(転写物)用

Rパッケージはおそらくありません。

プログラム:

- Multiple-k: Surget-Groba and Montoya-Burgos, Genome Res., 2010
- Trans-ABySS: Robertson et al., Nat Methods, 2010
- Rnnotator: Martin et al., BMC Genomics, 2010
- Trinity: Grabherr et al., Nat Biotechnol, 2011
- SOAPdenovo-trans: Luo et al., Gigascience, 2012
- Oases: Schulz et al., Bioinformatics, 2012
- EBARDenovo: Chu et al., Bioinformatics, 2013
- BRANCH: Bao et al., Bioinformatics, 2013
- IDBA-tran: Peng et al., Bioinformatics, 2013
- SOAPdenovo-Trans: Xie et al., Bioinformatics, 2014
- Rockhopper 2(バクテリア用): Tjaden B, Genome Biol., 2015
- DETONATE(RSEM-EVAL): Li et al., Genome Biol., 2014
- Bridger: Chang et al., Genome Biol., 2015
- IFRAT: Mbandi et al., BMC Bioinformatics, 2015

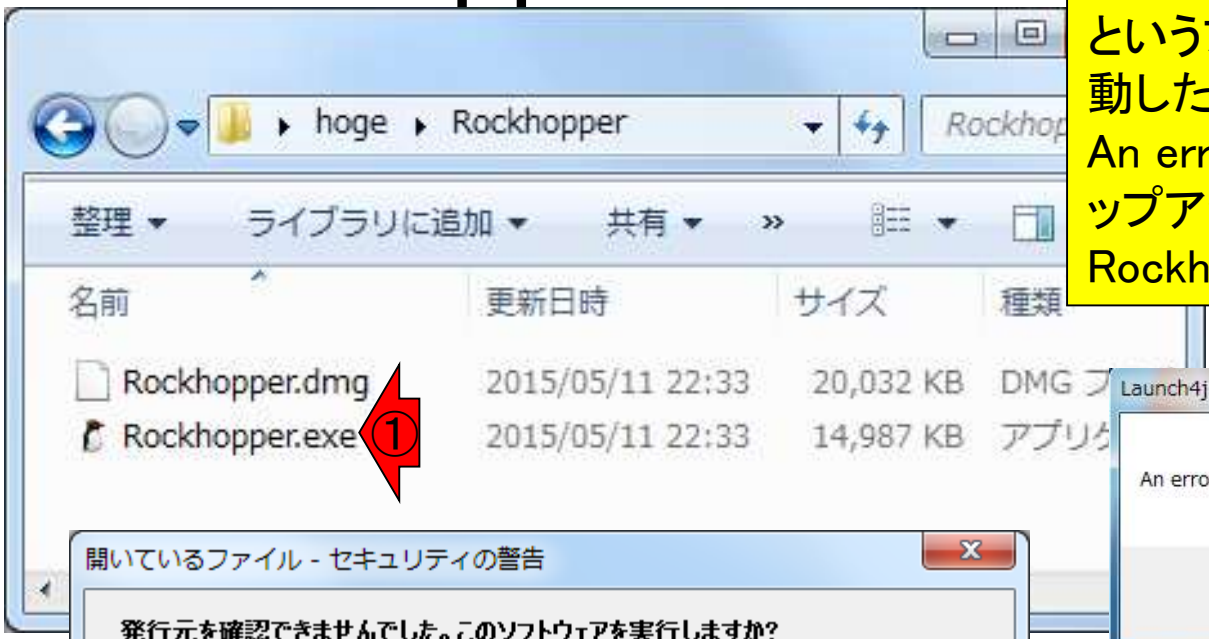
② (赤矢印)

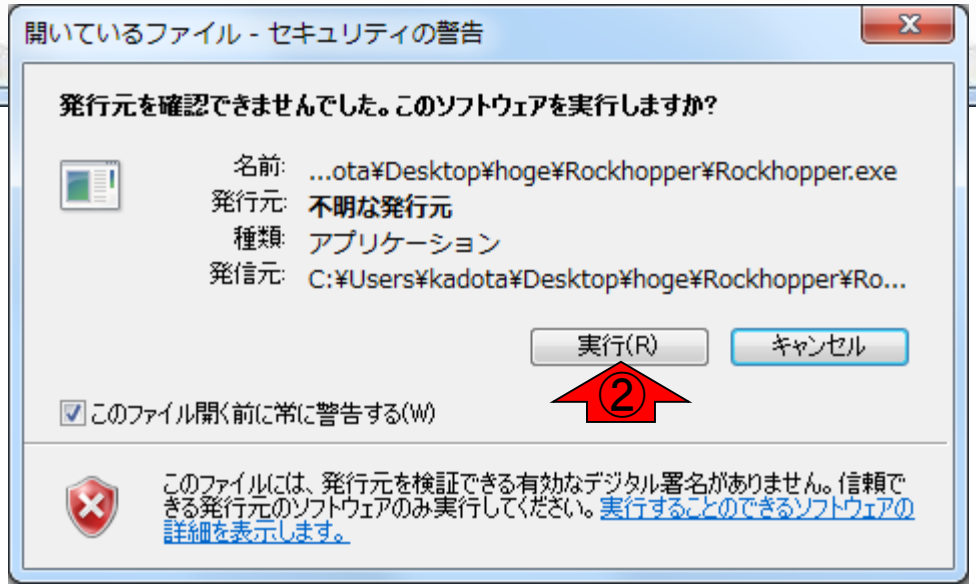
Review、ガイドライン、パイプライン系:

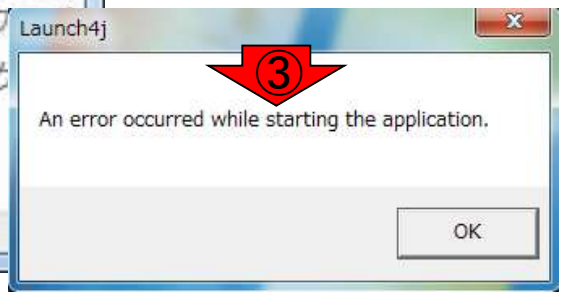
- Review: Martin and Wang, Nat Rev Genet., 2011
- ガイドライン: Haznedaroglu et al., BMC Bioinformatics, 2012
- Review: Góngora-Castillo, Nat Prod Rep., 2013
- ガイドライン: Yang and Smith, BMC Genomics, 2013
- ガイドライン: O'Neil et al., BMC Genomics, 2013
- ガイドライン: Feldmesser et al., BMC Genomics, 2014
- パイプライン(454用): Melicher et al., BMC Genomics, 2014
- パイプライン(組合せ系; SAMP and CDTA): He et al., BMC Genomics, 2015

Rockhopper

①Windowsのヒトは.exeファイル、Macintoshのヒトは.dmgファイルでRockhopperを起動。「hoge – Rockhopper」フォルダ中にRockhopper_Resultsというフォルダがないヒトは、ダブルクリックで起動したときに自動的に作成されます。③最初「An error occurred while opening the file」的なポップアップが出現しますが、30秒ほど待つと、④RockhopperのGUIが起動します。



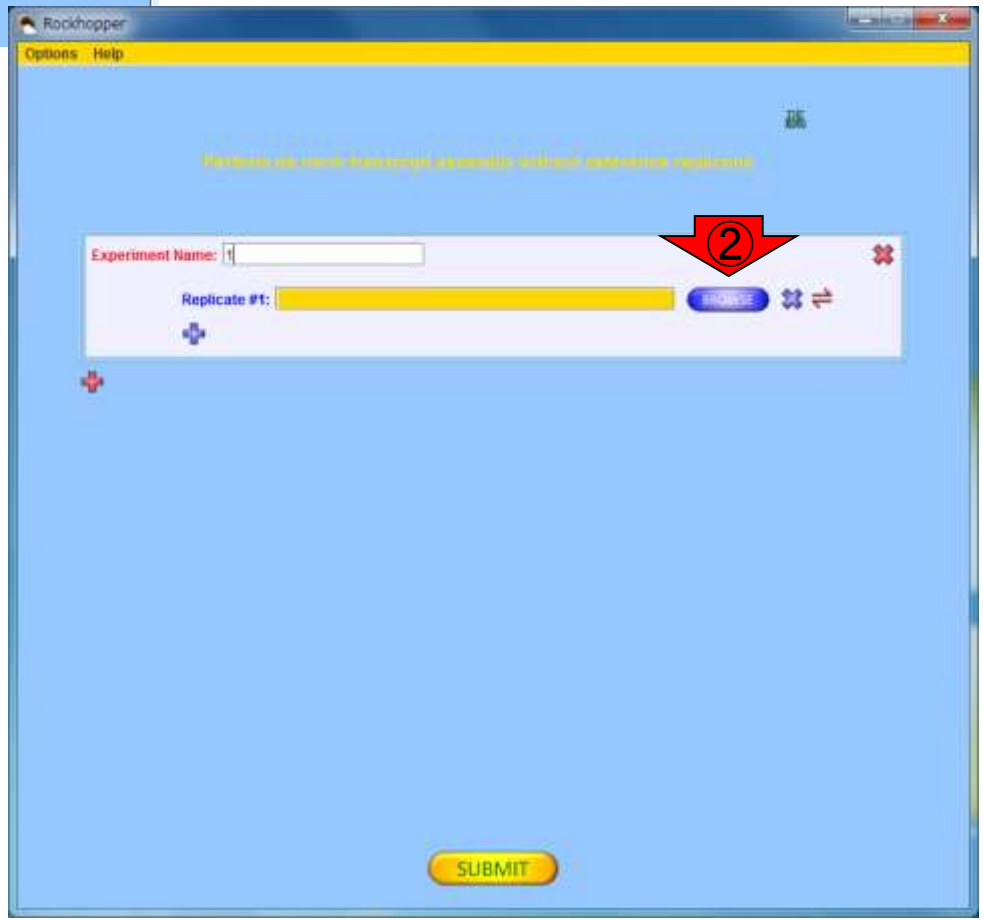




④

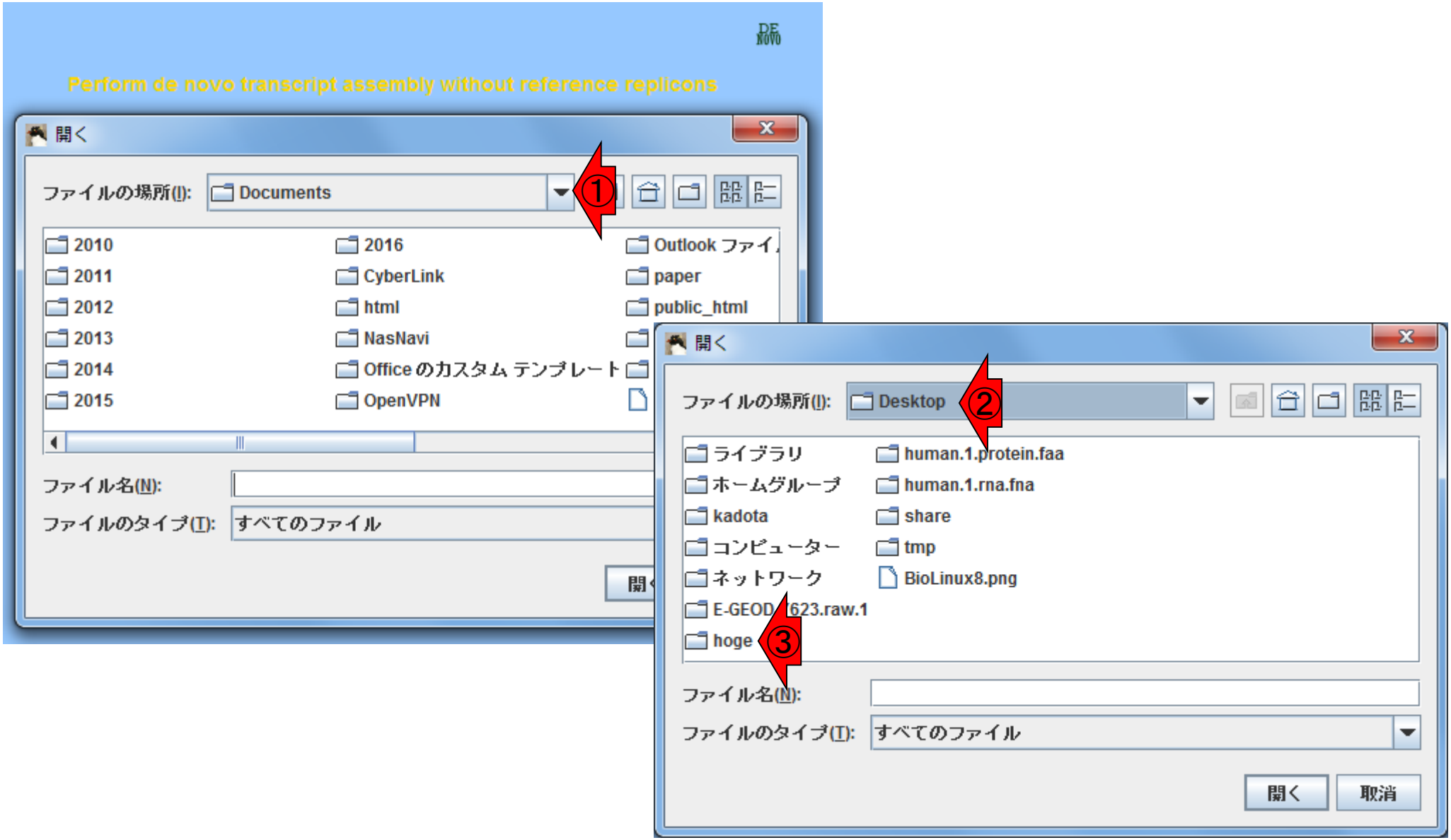
Rockhopper

①(見づらいが)DE NOVOと赤字で書いている部分をクリック。②入力ファイルを指定すべく、BROWSEボタンを押す。



Rockhopper

①ファイルの場所を指定、②Desktop、③hoge。



乳酸菌RNA-seqデータ

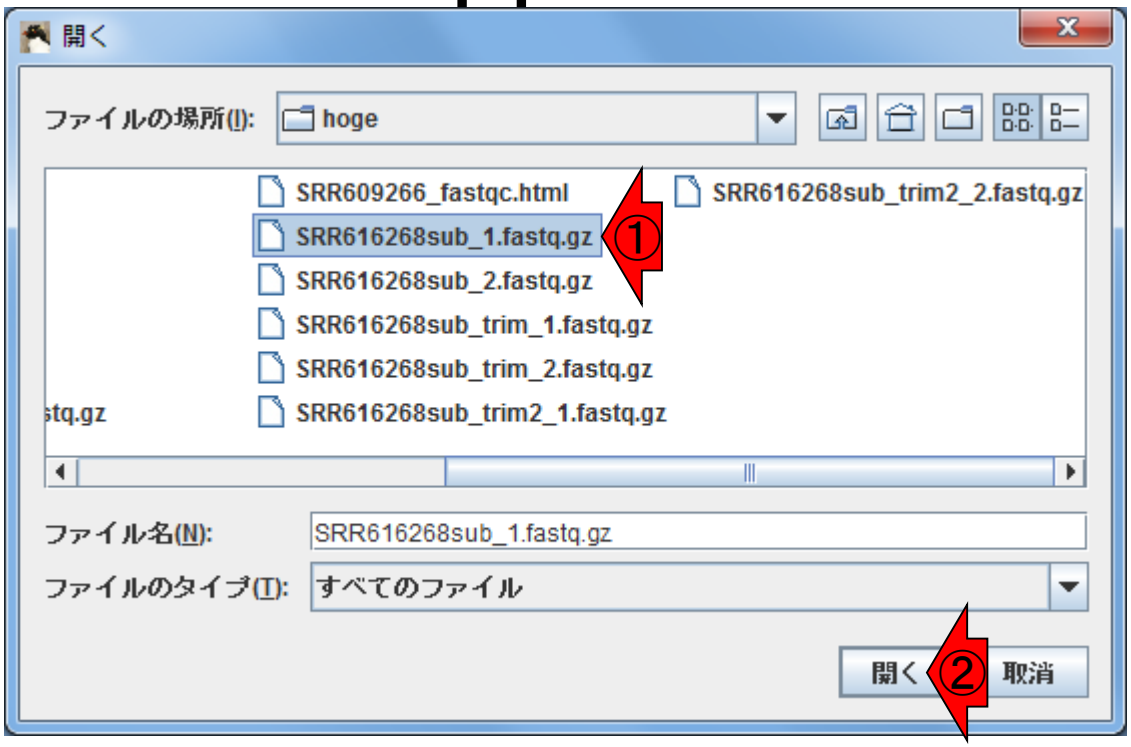
オリジナルの100万リードからなる生データファイルで実行（2015.05.12の発展課題と同じ）

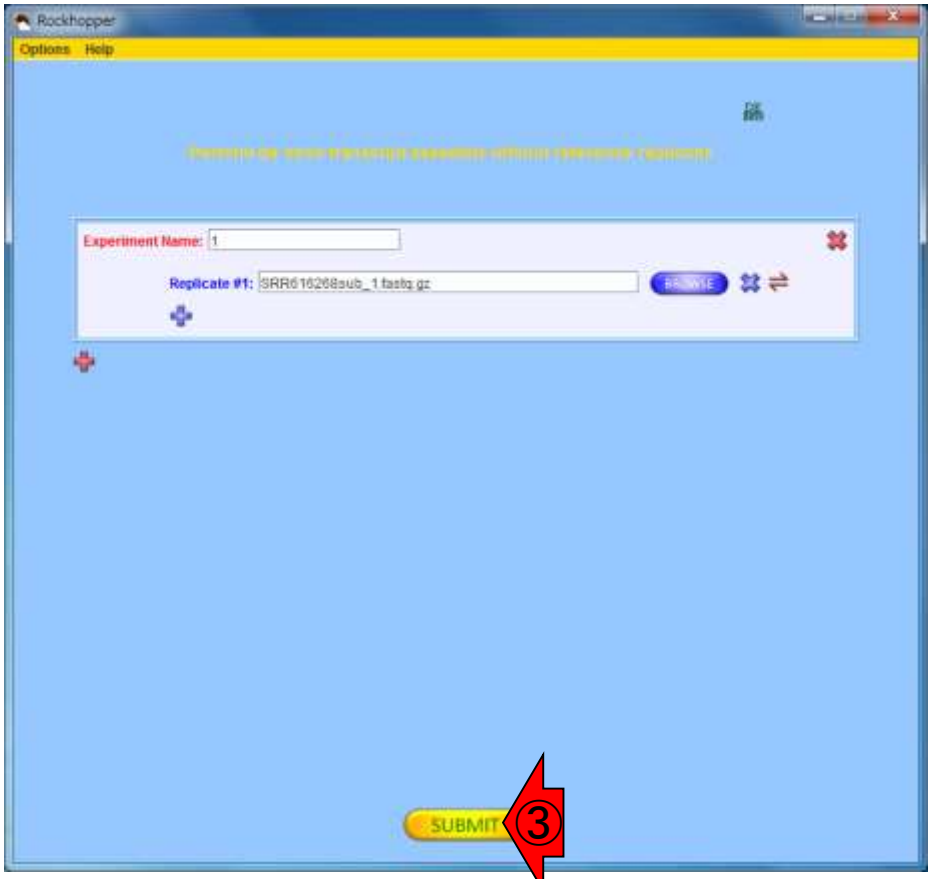
- 100万リードのオリジナル?!ファイル
  - SRR616268sub_1.fastq.gz : 107 bp, 74,906,576 bytes
  - SRR616268sub_2.fastq.gz : 93 bp, 67,158,462 bytes

- 100万リードのアダプター除去後のファイル
  - SRR616268sub_trim2_1.fastq.gz(hoge4.fastq.gz) : 71,343,605 bytes
  - SRR616268sub_trim2_2.fastq.gz : 63,524,791 bytes
  - hoge2_1.fastq.gz: 998,208リード、71,194,586 bytes
  - hoge2_2.fastq.gz: 998,208リード、63,375,473 bytes
- アダプター除去およびフィルタリング後のファイル
  - SRR616268sub_trim_1.fastq.gz(hoge7.fastq.gz) : 998,658リード、71,227,695 bytes
  - SRR616268sub_trim_2.fastq.gz(hoge8.fastq.gz) : 999,136リード、63,442,904 bytes
  - hoge1_1.fastq.gz: 998,428リード、63,446,240 bytes
  - hoge1_2.fastq.gz: 998,428リード、56,019,410 bytes

Rockhopper：SE

まずはsingle-end (SE)データとして実行。①入力ファイルを選び、②開く。③SUBMIT。約1分。





Rockhopper：SE

①入力ファイルは1,000,000リードだったので、一定の基準で内部的にフィルタリングしているのだろう。②アセンブル後のリファレンス配列に対してマッピングした結果、1,964リードがマップされたということだろう。③アセンブルによって得られた転写物配列数は8個しかなかった。④その平均配列長(121 bp)と⑤中央値(106 bp)。⑥アセンブルされたトータルの塩基数は974。これは③×④の結果(8×121=968)とほぼ同じなので、まあよしとしよう。

Progress

Initializing RNAseq analysis...

Assembling transcripts from reads in file: C:\Users\kadota\Desktop\hoge\SRR616268s

Aligning reads to assembled transcripts using file: C:\Users\kadota\Desktop\hoge\SRR616268s

Total reads in file:	996739 ①	
Perfectly aligned reads:	1964 ②	0%

Total number of assembled transcripts:	8 ③
Average transcript length:	121 ④
Median transcript length:	106 ⑤
Total number of assembled bases:	974 ⑥

Summary of results written to file: Rockhopper_Results/summary.txt
Details of assembled transcripts written to file: Rockhopper_Results/transcripts.txt

FINSIHED.

乳酸菌RNA-seqデータ

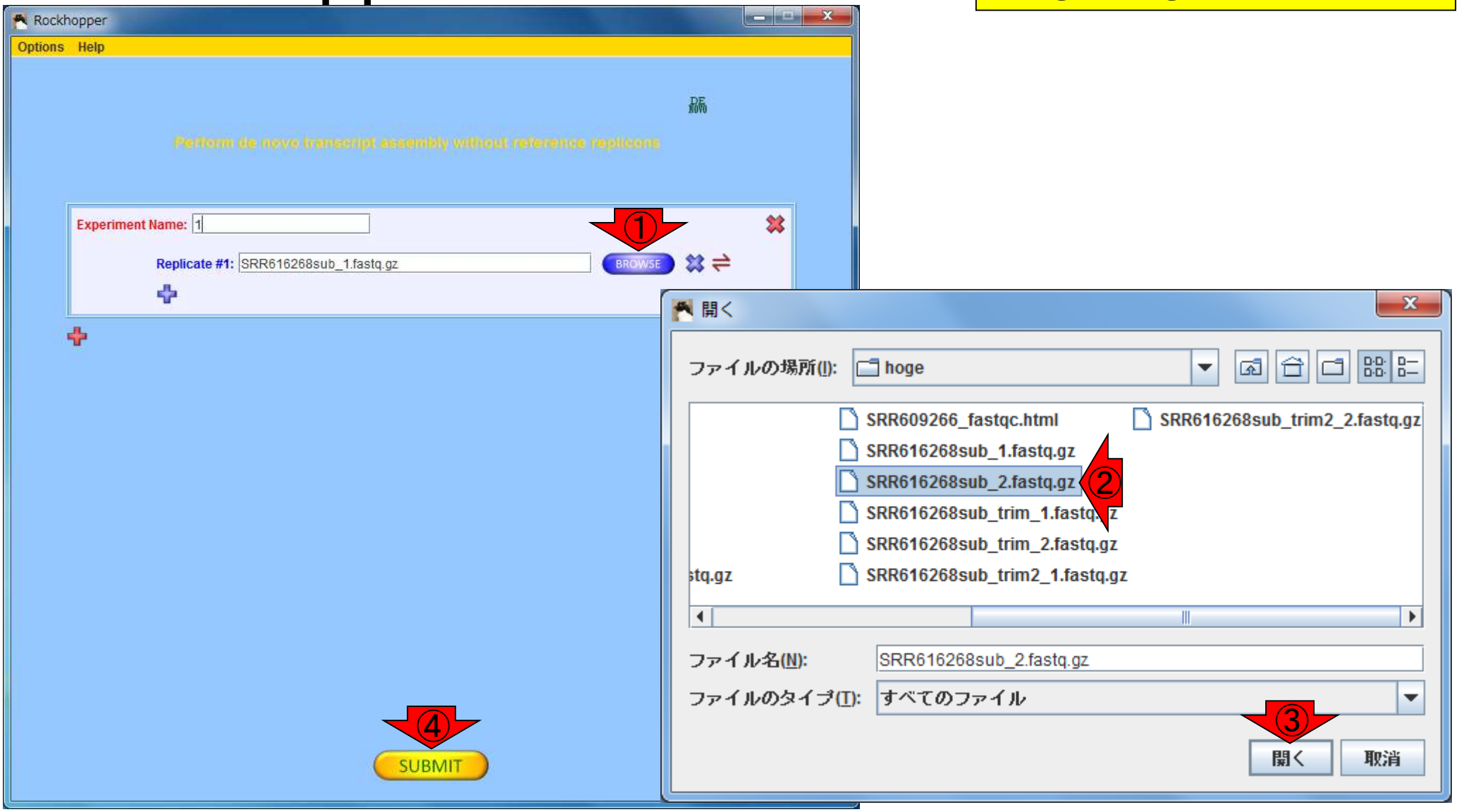
次にreverse側をsingle-end (SE) データとしてde novo assemble。

- 100万リードのオリジナル?!ファイル
  - SRR616268sub_1.fastq.gz : 107 bp, 74,906,576 bytes
  - SRR616268sub_2.fastq.gz : 93 bp, 67,158,462 bytes

- 100万リードのアダプター除去後のファイル
  - SRR616268sub_trim2_1.fastq.gz(hoge4.fastq.gz) : 71,343,605 bytes
  - SRR616268sub_trim2_2.fastq.gz : 63,524,791 bytes
  - hoge2_1.fastq.gz: 998,208リード、71,194,586 bytes
  - hoge2_2.fastq.gz: 998,208リード、63,375,473 bytes
- アダプター除去およびフィルタリング後のファイル
  - SRR616268sub_trim_1.fastq.gz(hoge7.fastq.gz) : 998,658リード、71,227,695 bytes
  - SRR616268sub_trim_2.fastq.gz(hoge8.fastq.gz) : 999,136リード、63,442,904 bytes
  - hoge1_1.fastq.gz: 998,428リード、63,446,240 bytes
  - hoge1_2.fastq.gz: 998,428リード、56,019,410 bytes

Rockhopper：SE

まずはsingle-end (SE)データとして実行。①入力ファイルを選び、②開く。③SUBMIT。約1分。



Rockhopper：SE

①入力ファイルは1,000,000リード。内部的にフィルタリングした結果、983,854リードになったのだろう。②アセンブル後のリファレンス配列に対してマッピングした結果、710,393リードがマップされたということだろう。③アセンブルによって得られた転写物配列数は424個。④その平均配列長(436 bp)と⑤中央値(228 bp)。⑥アセンブルされたトータルの塩基数は185,233。これは③×④の結果(424×436=184,864)とほぼ同じなので、まあよしとしよう

Progress

Initializing RNAseq analysis...

Assembling transcripts from reads in file: C:\Users\kadota\Desktop\hoge\SRR616268s

Aligning reads to assembled transcripts using file: C:\Users\kadota\Desktop\hoge\SRR616268s

Total reads in file:	983854 ①	
Perfectly aligned reads:	710393 ②	72%

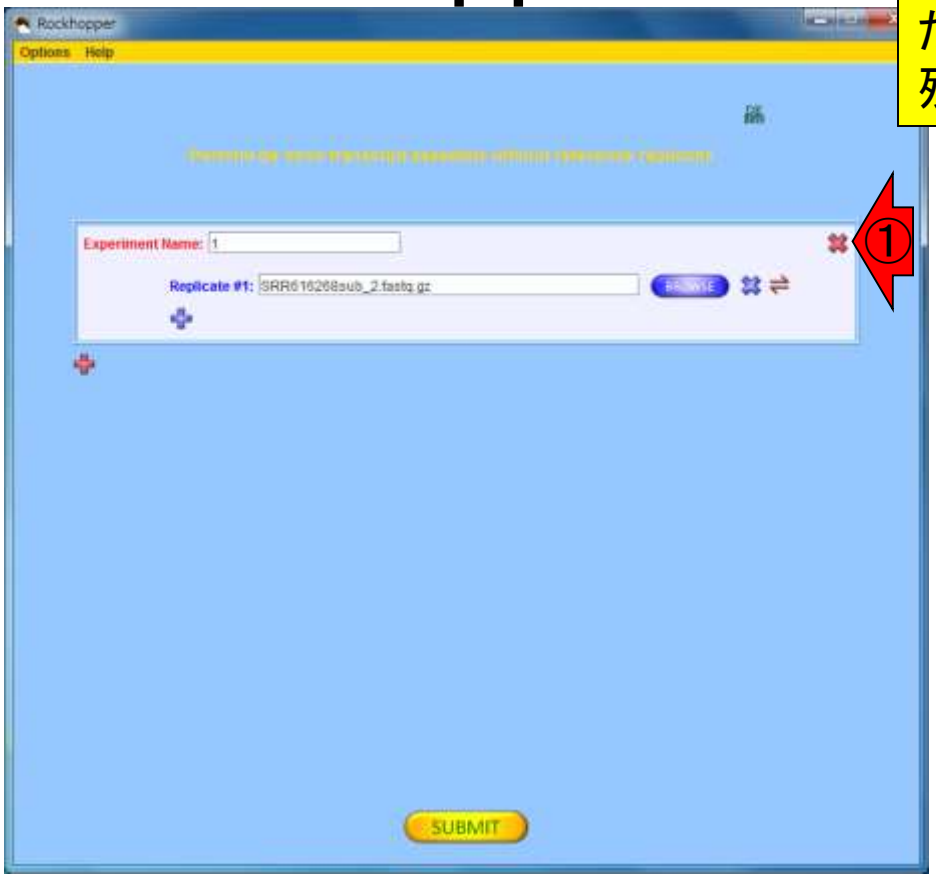
Total number of assembled transcripts:	424 ③
Average transcript length:	436 ④
Median transcript length:	228 ⑤
Total number of assembled bases:	185233 ⑥

Summary of results written to file: Rockhopper_Results/summary.txt
Details of assembled transcripts written to file: Rockhopper_Results/transcripts.txt

FINSIHED.

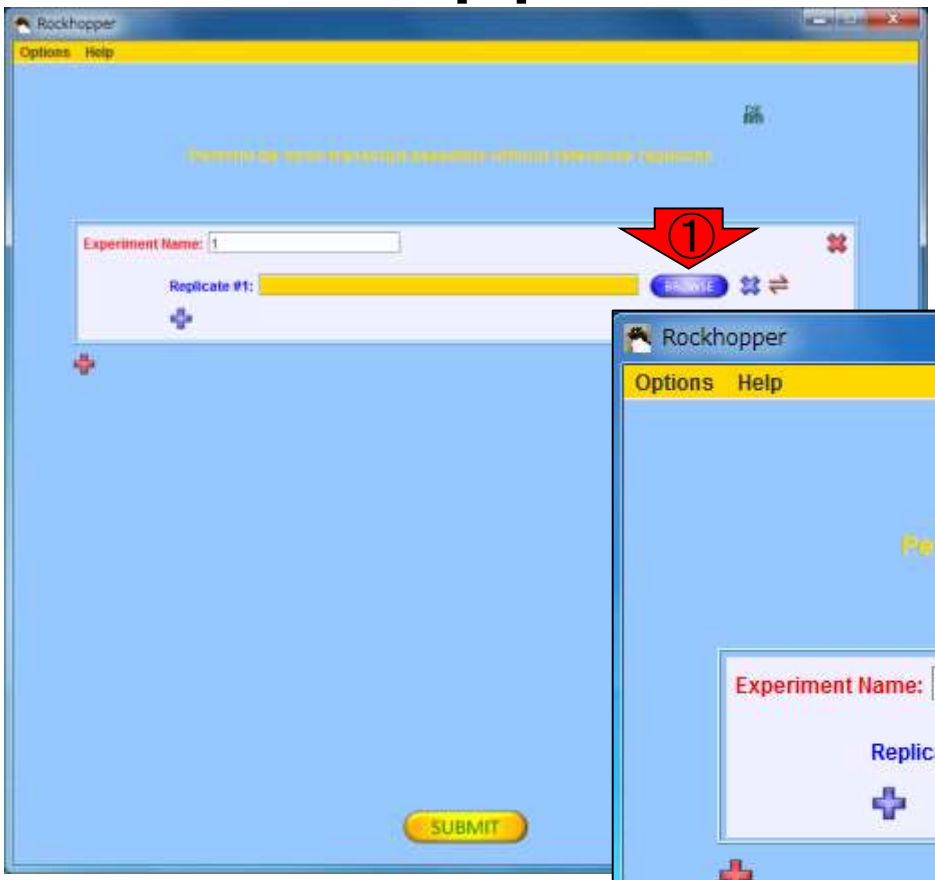
Rockhopper：PE

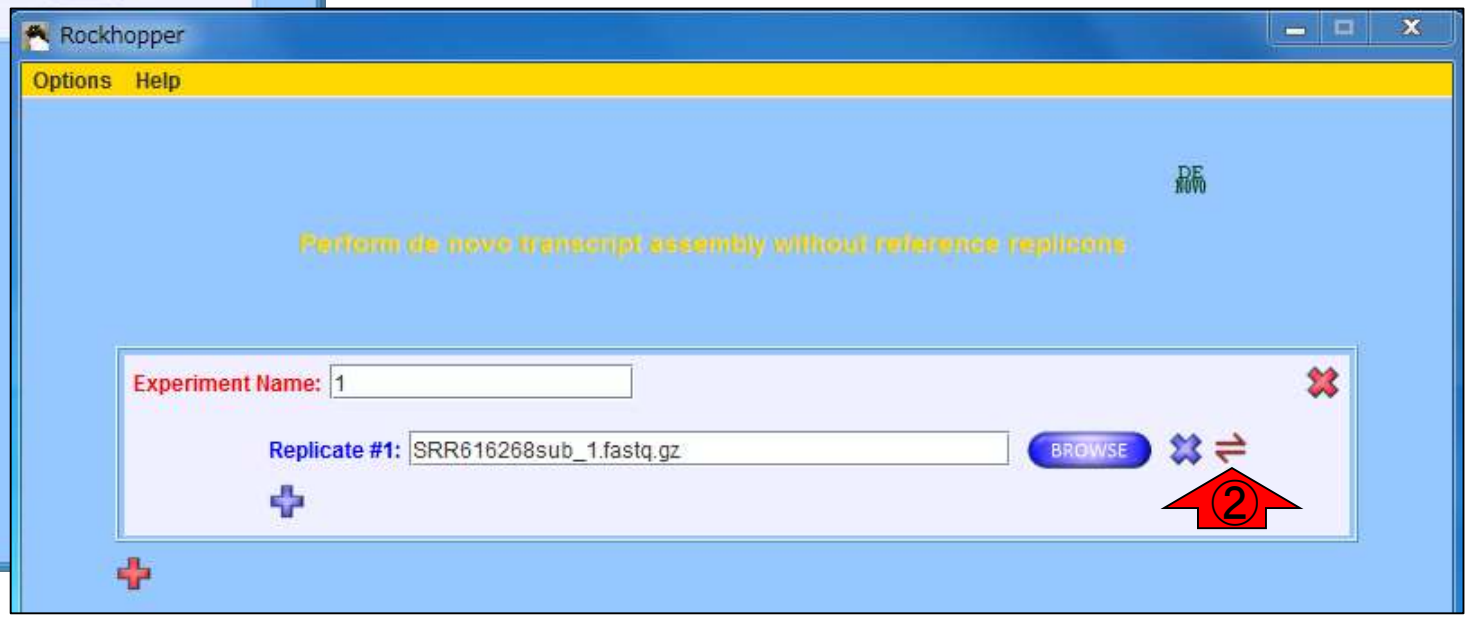
Paired-endデータとしてアセンブル。①念のため一旦消す。②「Options – Clear cache」というのがあったので、こちらも念のためやってみる。数分フリーズ状態になりました。③Rockhopperアプリごと再起動しても、キャッシュに残っているようなので、Clear cacheをやるのが無難かも

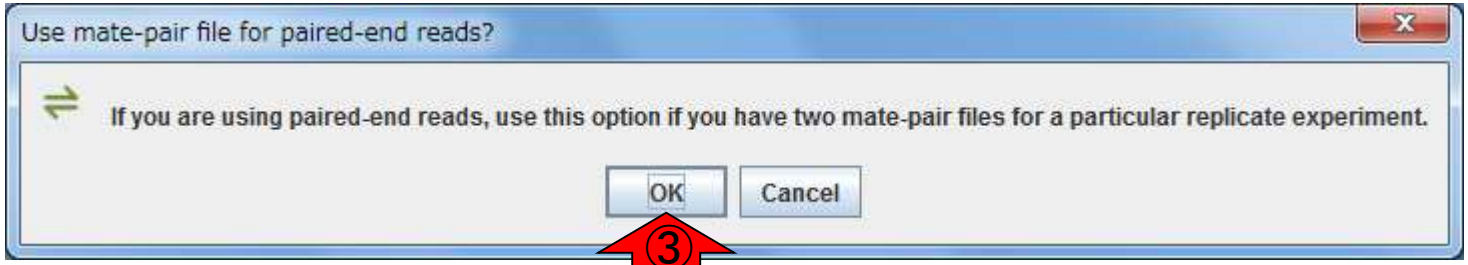


Rockhopper：PE

しばらく経つと左のような画面になるので、①paired-endの1つ目のforward側のファイルを指定。②「双方向矢印」のところにカーソルをもっていくと、「Using paired-end reads; …」と書いてあるので、ここをクリック。③OKを押す。

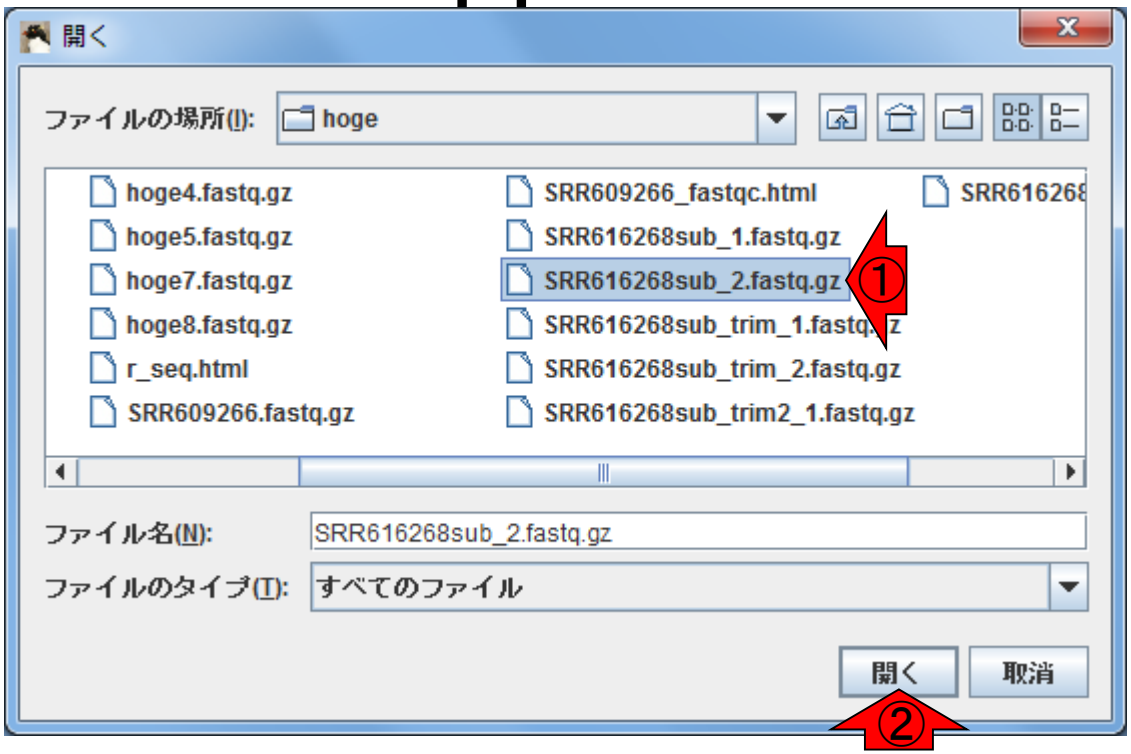


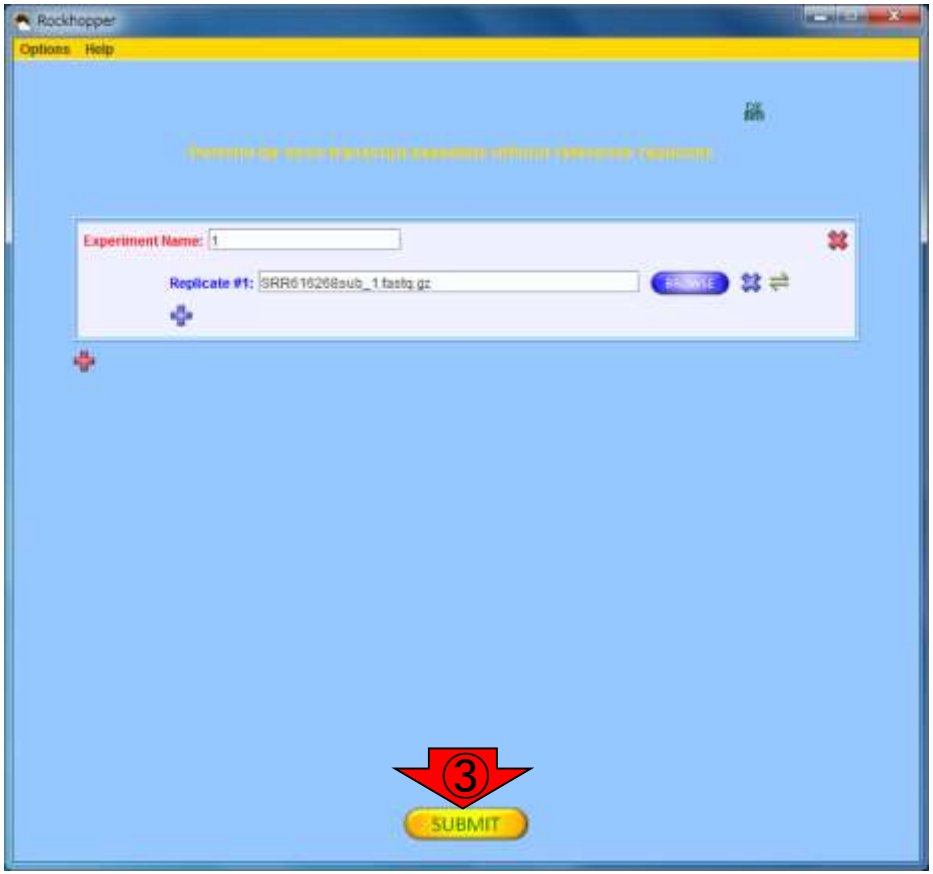




Rockhopper：PE

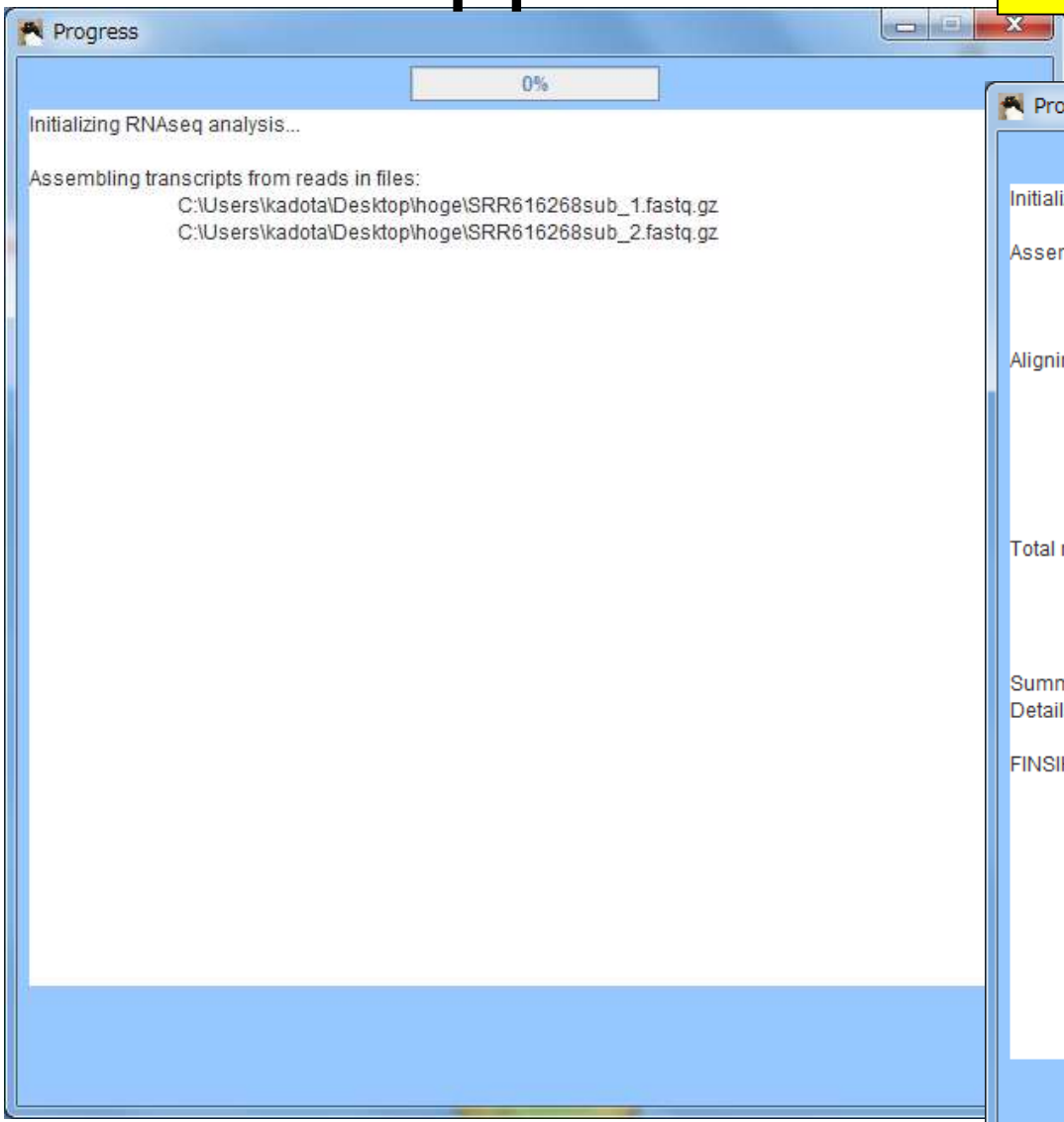
①reverse側のファイルを選択して、②開く。Paired-end の2つのファイルが見られるわけではないが、③SUBMIT

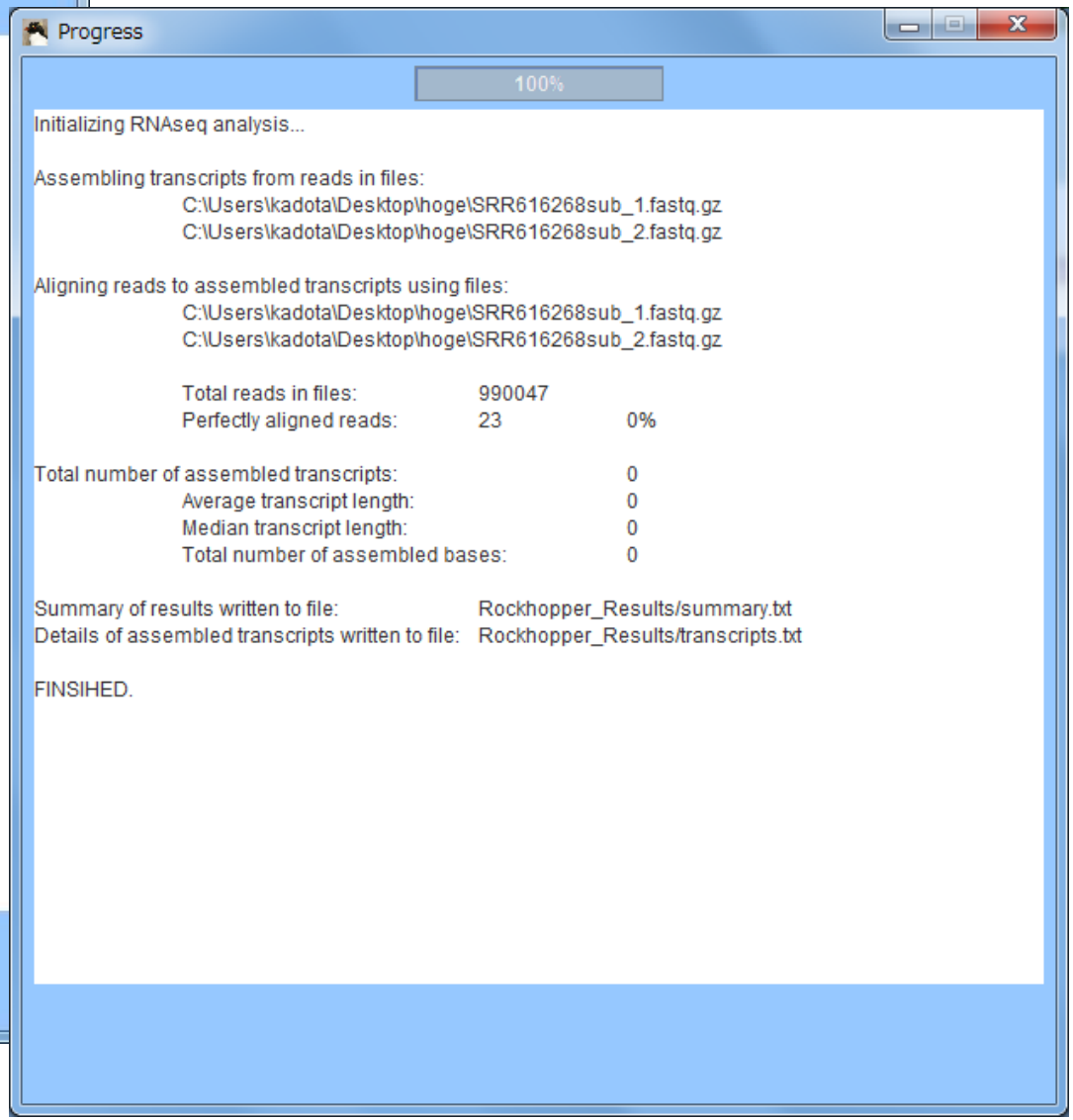




Rockhopper：PE

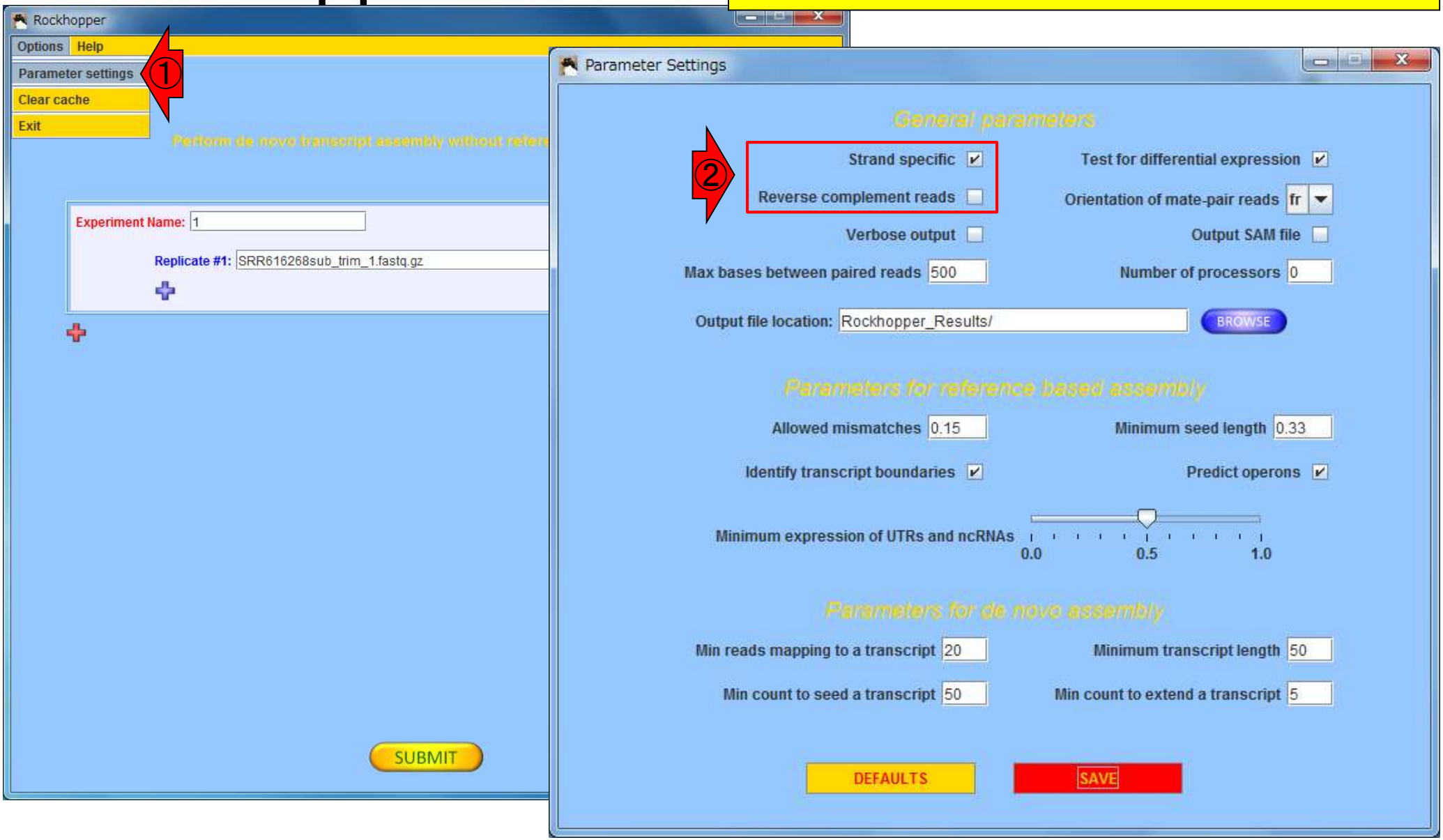
約2分で終わるが、残念ながらうまくアセンブルできていないようだ。後にこの原因は、forwardファイル中の3’側のadapter/primer配列が主原因と判明





Rockhopper：PE

それゆえ、①「Options – Parameter settings」で、②Strand specific やReverse complement reads のあたりをいじっても状況に変化はない。



いろいろやった結果のまとめです

アセンブル結果まとめ

ファイル名	Total number of assembled transcripts	Average length	Median length	Total number of assembled bases	Total reads in file	Perfectly aligned reads
・100万リードのオリジナル?!ファイル						
SRR616268sub_1.fastq.gz	8	121	106	974	996739	1964
SRR616268sub_2.fastq.gz	424	436	228	185233	983854	710393
paired-end	0	0	0	0	990047	23
・100万リードのアダプター除去後のファイル						
SRR616268sub_trim2_1.fastq.gz	8	121	104	974	993311	1231
SRR616268sub_trim2_2.fastq.gz	425	433	228	184437	981649	708951
hoge2_1.fastq.gz	8	121	104	974	993001	1231
hoge2_2.fastq.gz	424	434	229	184210	980997	708395
paired-end	0	0	0	0	989869	25
・アダプター除去およびフィルタリング後のファイル						
SRR616268sub_trim_1.fastq.gz	8	121	104	974	993311	1231
SRR616268sub_trim_2.fastq.gz	425	433	228	184437	981649	708951
hoge1_1.fastq.gz	8	121	104	974	993081	1231
hoge1_2.fastq.gz	423	435	229	184211	981125	708482
paired-end	0	0	0	0	989869	25

Contents

- 先週の課題やエラーについて
  - 課題1の解説
  - アダプター配列除去（QuasR）実行時のエラーは門田のミス
  - Paired-endデータの取り扱い
- フィルタリング(filtering)
  - 基本形：リード数が同じpaired-endの場合（QuasRパッケージ）
  - 発展形：リード数が異なる場合（ShortReadパッケージ）
- アセンブル(assemble)
  - ゲノム用とトランスクリプトーム用
  - Rockhopper2（バクテリアのトランスクリプトーム用）
- マッピング（mapping）
  - 基礎、オプション、仮想データ説明
  - マッピング本番、出力ファイル形式、オプションと結果の関係
  - カウント情報取得
- 実データ解析
  - QuasRでマッピング → トリミング → 高精度なアセンブルとマッピングへ

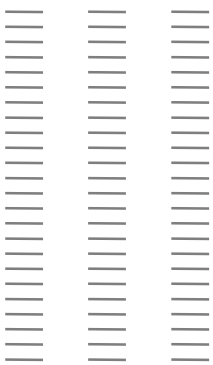
マッピング基礎

基本イメージ。「マップされたリード数 = 発現量」ではないが、マップされたリード数のカウント情報は、発現量推定の基本情報

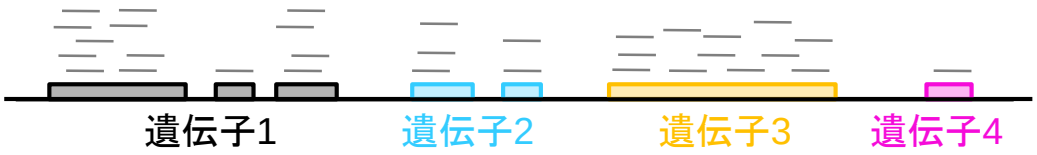
- 基本的なマッピングプログラム（bowtieなど）を用いた場合

T1サンプルの
RNA-Seqデータ

mapping



リファレンス配列：ゲノム



count

	T1
遺伝子1	14
遺伝子2	5
遺伝子3	12
遺伝子4	1
遺伝子5	…
…	…

リファレンス配列：トランスクリプトーム

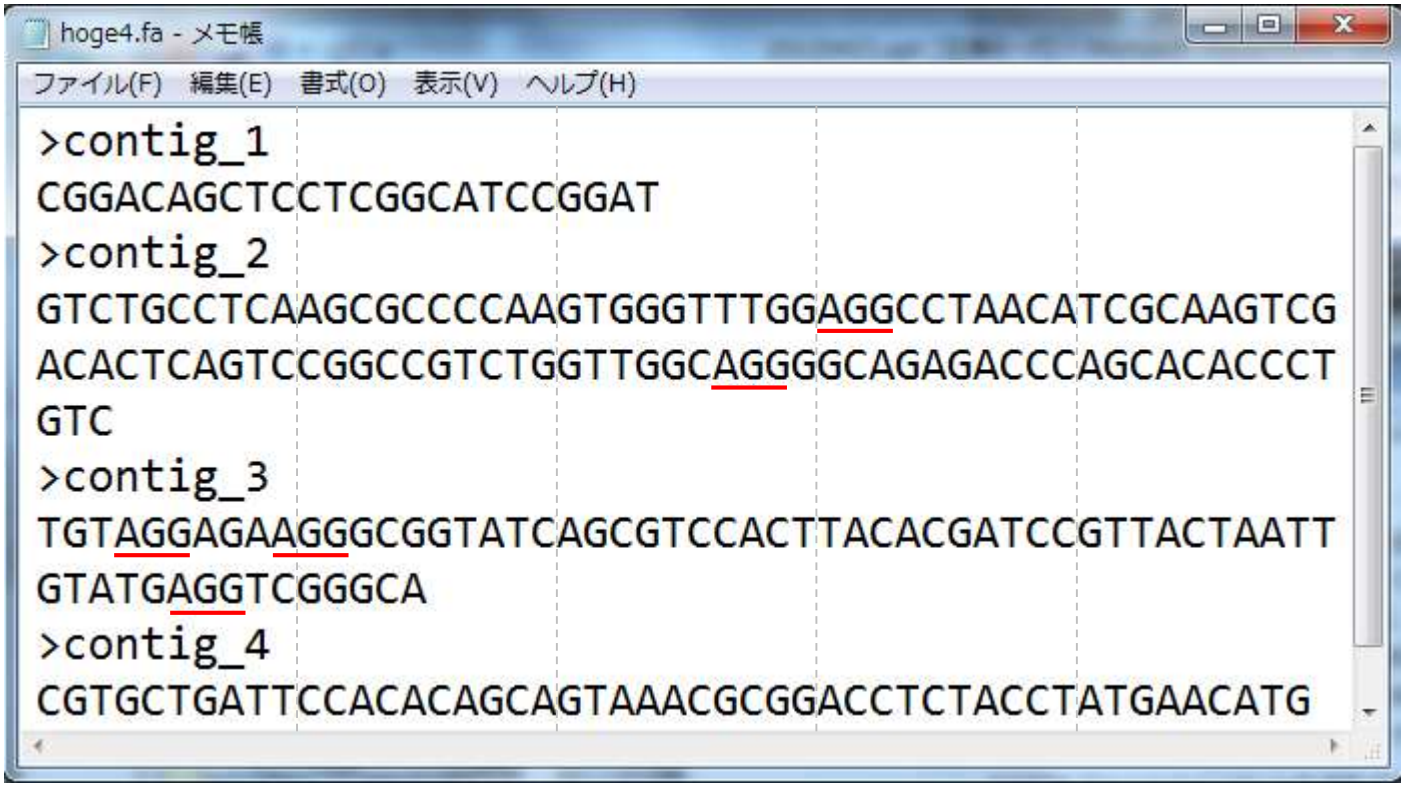
count

	T1
遺伝子1	19
遺伝子2	7
遺伝子3	12
遺伝子4	1
遺伝子5	…
…	…

マッピング基礎

- 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”

「マッピング ＝ 大量高速文字列検索」という捉え方でよい。マッピングプログラムの出力は、（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



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

教科書p81-89

マッピング

- 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 2014/06/20)
- アセンブル | ゲノム用 (last modified 2014/07/08)
- アセンブル | トランスクリプトーム(転写物)用 (last modified 2014/07/08)
- マッピング | について (last modified 2015/01/16)
- マッピング | basic aligner (last modified 2014/08/08)
- マッピング | splice-aware aligner (last modified 2014/07/09)
- マッピング | Bisulfite sequencing用 (last modified 2014/07/09)
- マッピング | (ESTレベルの長さの)contig (last modified 2014
- 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
- マッピング | single-end | ゲノム | basic aligner(応用) | QuasR
- マッピング | single-end | ゲノム | splice-aware aligner | QuasR
- マップ後 | について (last modified 2013/06/19)
- マップ後 | 出力ファイル形式について (last modified 2013/11

①

Rパッケージとして提供されているものは圧倒的に少ないですが、QuasRパッケージ(Gaidatzis et al., 2015)のおかげでWindows OS上でもマッピングを気軽に利用できるようになりました。これは内部的にBowtie (Langmead et al., 2009)プログラムを利用しています。

マッピング | について

リファレンス配列にマッピングを行うプログラム達です。basic aligner (unspliced aligner)はsplice-aware aligner (spliced aligner)内部で使われていたりします。

R用:

- Rsubread(Windows版なし): Liao et al., Nucleic Acids Res., 2013
- QuasR(Windows版あり): Gaidatzis et al., Bioinformatics, 2014
- HTSeqGenie(Windows版なし): 原著論文はまだみたいです

R以外(basic aligner; unspliced aligner):

- SSAHA2: Ning et al., Genome Res., 2001
- RMAP: Smith et al., BMC Bioinformatics, 2008
- MAQ: Li et al., Genome Res., 2008
- PASS: Campagna et al., Bioinformatics, 2009
- MOM: Eaves and Gao, Bioinformatics, 2009
- Bowtie: Langmead et al., Genome Biol., 2009
- BWA: Li and Durbin, Bioinformatics, 2009(BWA-shortの論文)
- SHRiMP: Rumble et al., PLoS Comput. Biol., 2009

教科書p81-89

オプション

オプションは、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

- QuasRパッケージは内部的にBowtie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(mismatch)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
  - …

仮想リファレンス配列

- マップされる側のリファレンス配列：ref_genome.fa

chr3とchr5の違いは、2番目と7番目の塩基のみ。主に”-m”オプション(-m 1：1ヶ所にマップされるリードのみ出力)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
```

教科書p81-89

コピペで作成しています

仮想リファレンス配列

- マップされる側のリファレンス配列：ref_genome.fa

- 個別パッケージのインストール (last modified 2015/02/20) NEW
- 基本的な利用法 (last modified 2015/01/16)
- サンプルデータ (last modified 2015/02/15) NEW
- バイオインフォマティクス人材育成カリキュラム(次世代シークエンサ) | 速習コース (last
- 書籍
- 書籍
- 書籍

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(SupplementaryTable2.txt)です。サンプルは二つの濃度(1.5 pM and 3 pM)でシーケンスされており、「3 pMのものが5 samples vs. 5

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
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
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, n) {             #関数名や引数の作成
```

仮想RNA-seqリード

- マップする側のRNA-seqデータ：sample_RNAseq1.fa

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行に情報を記載。

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
```

教科書p81-89

コピペで作成しています

仮想RNA-seqリード

- マップする側のRNA-seqデータ：sample_RNAseq1.fa

- 個別パッケージのインストール (last modified 2015/02/20) NEW
- 基本的な利用法 (last modified 2015/01/16)
- サンプルデータ (last modified 2015/02/15) NEW
- バイオインフォマティクス人材育成カリキュラム(次世代シークエンサ) | 速習コース (last
- 書籍
- 書籍
- 書籍

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(Supplementa
す。サンプルは二つの濃度(1.5 pM and 3 pM)でシーケンスされており、「3 pMのものが5

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 alig
グする際の動作確認用RNA-seqデータ(sample_RNAseq1.fa)とその
ル(sample_RNAseq1.fa.gz)。リファレンス配列を読み込んで、list_su
部分配列を抽出したものです。どこに置換を入れているかがわかっ
basic alignerで許容するミスマッチ数を変えてマップされるorされな
ます。DNAStringSetオブジェクトを入力として塩基置換を行うDNAS
数を用いて、最後のリードのみ4番目の塩基にミスマッチを入れてい

```
in_f1 <- "ref_genome.fa"                #入力ファイル
in_f2 <- "list_sub3.txt"                #入力ファイル
out_f <- "sample_RNAseq1.fa"            #出力ファイル
param <- 4                              #塩基置換した

#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
```

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
```

Contents

- 先週の課題やエラーについて
  - 課題1の解説
  - アダプター配列除去（QuasR）実行時のエラーは門田のミス
  - Paired-endデータの取り扱い
- フィルタリング(filtering)
  - 基本形：リード数が同じpaired-endの場合（QuasRパッケージ）
  - 発展形：リード数が異なる場合（ShortReadパッケージ）
- アセンブル(assemble)
  - ゲノム用とトランスクリプトーム用
  - Rockhopper2（バクテリアのトランスクリプトーム用）
- マッピング（mapping）
  - 基礎、オプション、仮想データ説明
  - マッピング本番、出力ファイル形式、オプションと結果の関係
  - カウント情報取得
- 実データ解析
  - QuasRでマッピング → トリミング → 高精度なアセンブルとマッピングへ

やってみましょう

マッピング本番

- マッピング | (ESTレベルの長さの)contig (last modified 2014/06/24)
- 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(Gaidatzis_2014) (last modified 2014/06/21)
- マッピング | single-end | ゲノム | basic aligner(応用) | QuasR(Gaidatzis_2014) (last modified 2015/02/24) ①
- マッピング | single-end | ゲノム | splice-aware aligner | QuasR(Gaidatzis_2014) (last modified 2014/06/21)
- マップ後 | について (last modified 2013/06/19)
- マップ後 | 出力フ
- マップ後 | 出力フ

マッピング | single-end | ゲノム | basic aligner(応用) | QuasR(Gaidatzis_2014) 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。

②

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_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

#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out
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
```

やってみましょう

マッピング本番

- マッピング | (ESTレベルの長さの)contig (last modified 2014/06/24)
- 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(Gaidatzis_2014) (last modified 2014/06/21)
- マッピング | single-end | ゲノム | basic aligner(応用) | QuasR(Gaidatzis_2014) (last modified 2015/02/24)
- マッピング | single-end | ゲノム | splice-aware aligner | QuasR(Gaidatzis_2014) (last modified 2014/06/21)
- マップ後 | について (last modified 2013/06/19)
- マップ後 | 出力フ
- マップ後 | 出力フ

①

	A	B
1	FileName	SampleName
2	sample_RNAseq1.fa	namae

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

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
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

②

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

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

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

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

#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out
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i
```

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”)

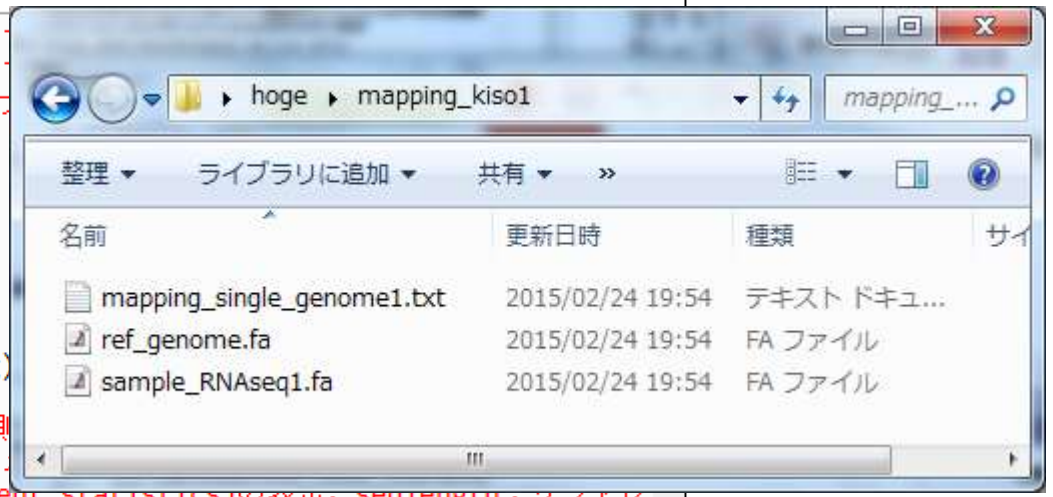
マッピング本番

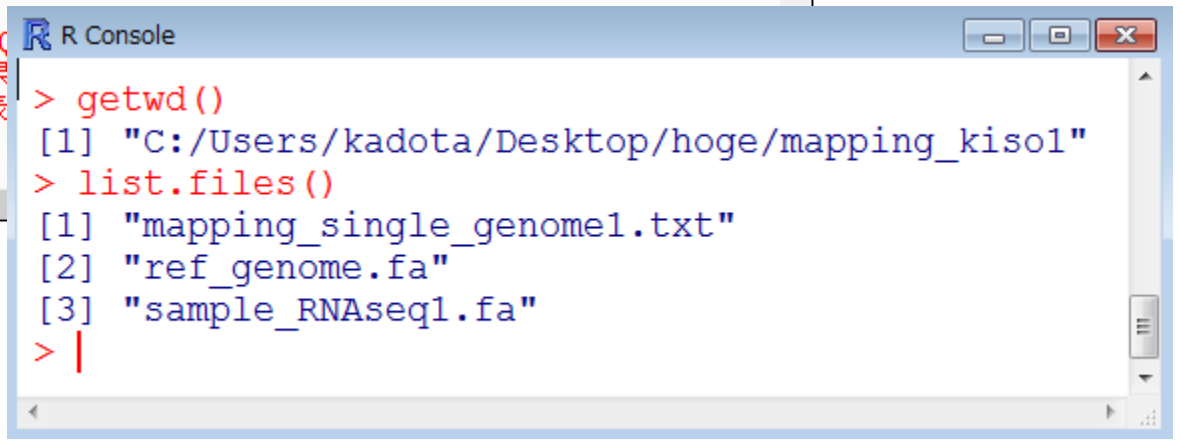
「hoge – mapping_kiso1」に最低限必要な3つの入力ファイルが揃っていることを確認してコピペ。ファイヤーフォールか何かでアクセス許可関連のポップアップが出たらOKを押す

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(m

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
("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("ch
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
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

#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)
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
out                                    #マッピングに用いたパラ
alignmentStats(out)                    #マッピング結果(alignment statistics)の表示。seqlength:リファレ

#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])#Q
qQCReport(out, pdfFilename=out_f)      #QCレポート結果
out_f                                  #ファイル名を表

#ファイルに保存(BED形式ファイル)
```





出力ファイル形式

出力ファイルとして実際に取り扱うのは①BAM形式ファイルです。②赤枠部分はランダムに発生させる文字列部分なので、ヒトによって異なります

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(

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

```
in_f1 <- "mapping_single_genome1.txt"  #入力ファイル名を指定してin_f1に格納(RNA-seqファイル)
in_f2 <- "ref_genome.fa"
param_mapping <- "-m 1 --best --st

#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()
out <- qAlign(in_f1, in_f2, alignm
time_e <- proc.time()
time_e - time_s
out
alignmentStats(out)

#ファイルに保存(QCレポート用のpdfフ
out_f <- sub(".bam", "_QC.pdf", ou
qQCReport(out, pdfFilename=out_f)
out_f

#ファイルに保存(BED形式ファイル)
```

R Console

```
+    out_f <- sub(".bam", ".bed", tmpfname[i])#BED形式ファイル名$
+    out_f                                    #ファイル名を表示して$
+    write.table(tmp, out_f, sep="\t", append=F, quote=F, row.na$
+ }
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/mapping_kiso1"
> list.files()
 [1] "mapping_single_genome1.txt"
 [2] "QuasR_log_21744443273a.txt"
 [3] "ref_genome.fa"
 [4] "ref_genome.fa.fai"
 [5] "ref_genome.fa.md5"
 [6] "ref_genome.fa.Rbowtie"
 [7] "sample_RNAseq1.fa"
 [8] "sample_RNAseq1_217479832d2f.bam"
 [9] "sample_RNAseq1_217479832d2f.bam.bai"
[10] "sample_RNAseq1_217479832d2f.bam.txt"
[11] "sample_RNAseq1_217479832d2f.bed"
[12] "sample_RNAseq1_217479832d2f_QC.pdf"
>
```

②

①

②

実用上はBAM形式、視覚上はBED形式

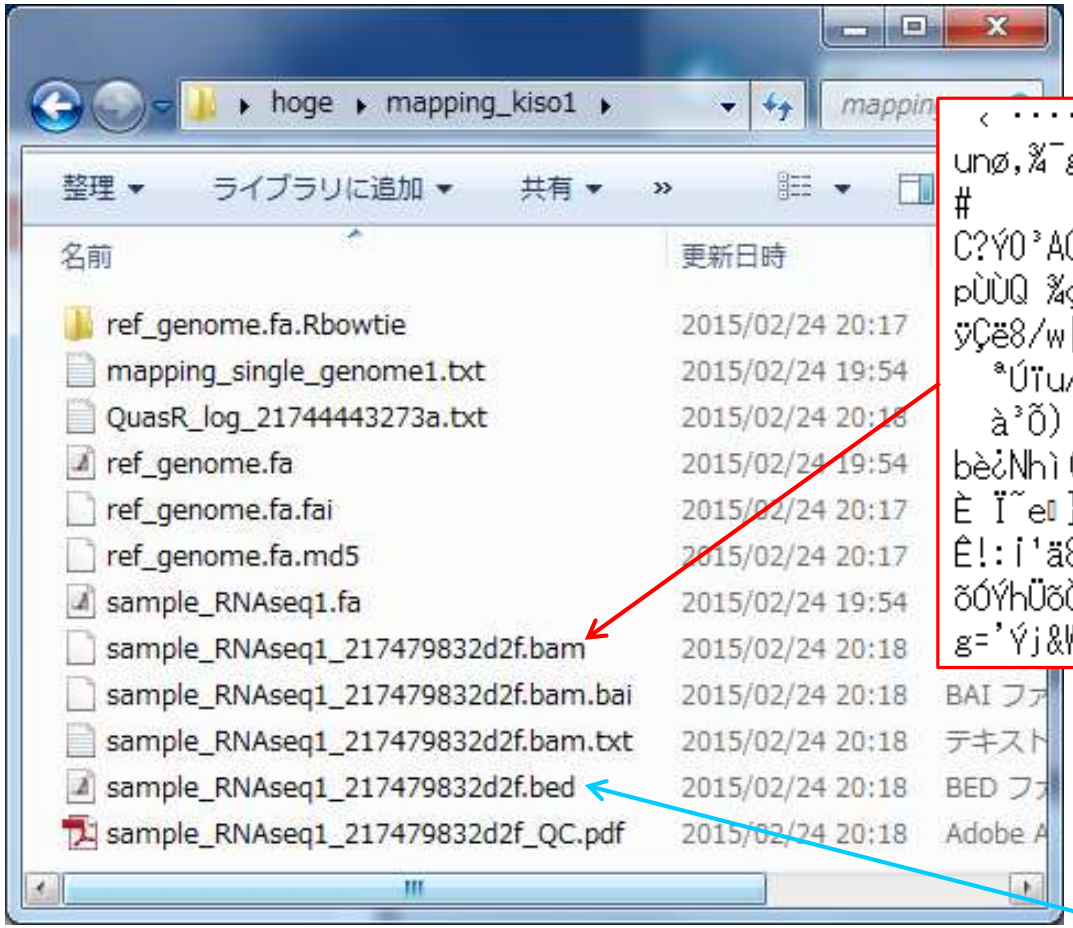
出力ファイル形式

- 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）を表す出力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)

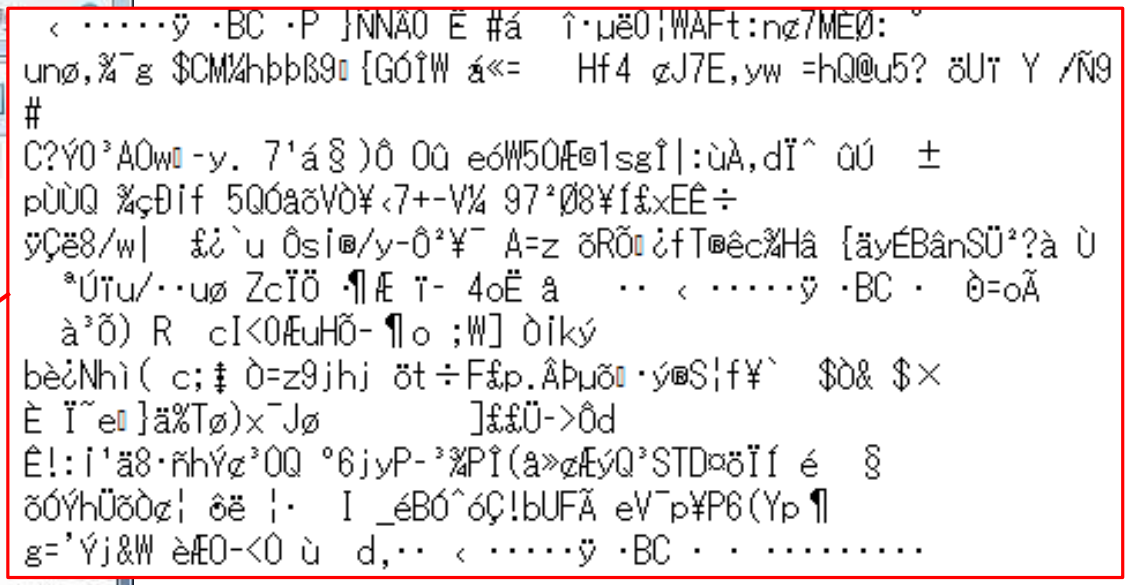
  ...

BEDの最小限の情報は、リードIDを含まない

出力ファイル形式



BAM形式ファイル



BED形式ファイル

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

オプションと結果の関係

マップされなかったのは、
赤枠の8リード中3リード

- “-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
```

オプションと結果の関係

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のは2リード。-m 1：1ヶ所にマップされるリードのみ出力

- “-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
```

オプションと結果の関係

1か所にのみマップされるがミスマッチのため落とされたのは1リード。-v 0: 許容するミスマッチ数は0

- “-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
```

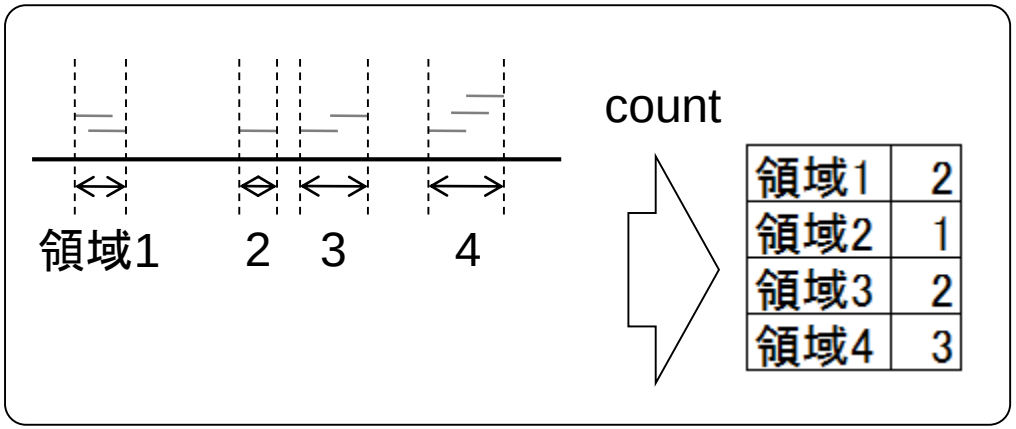
Contents

- 先週の課題やエラーについて
  - 課題1の解説
  - アダプター配列除去（QuasR）実行時のエラーは門田のミス
  - Paired-endデータの取り扱い
- フィルタリング(filtering)
  - 基本形：リード数が同じpaired-endの場合（QuasRパッケージ）
  - 発展形：リード数が異なる場合（ShortReadパッケージ）
- アセンブル(assemble)
  - ゲノム用とトランスクリプトーム用
  - Rockhopper2（バクテリアのトランスクリプトーム用）
- マッピング（mapping）
  - 基礎、オプション、仮想データ説明
  - マッピング本番、出力ファイル形式、オプションと結果の関係
  - カウント情報取得
- 実データ解析
  - QuasRでマッピング → トリミング → 高精度なアセンブルとマッピングへ

カウント情報取得

Rでのアノテーション情報利用は、TranscriptDbが基本

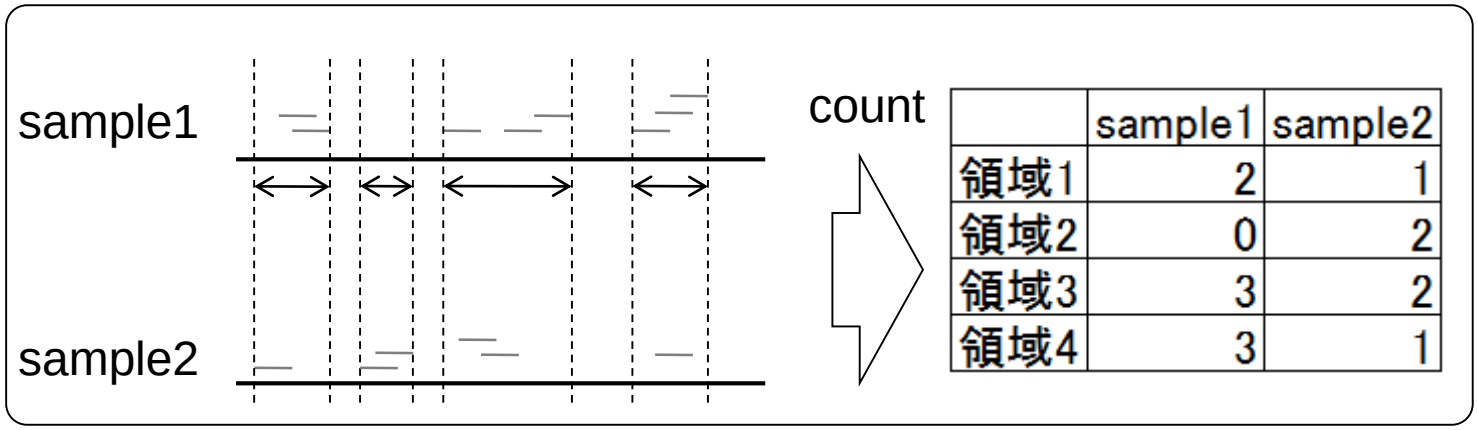
- アノテーション情報を利用する場合
  - UCSC known Genes, Ensembl Genesなど様々なテーブル名を指定可能
  - gene, exon, promoter, junctionなど様々なレベルを指定可能
- アノテーション情報がない場合
  - マップされたリードの和集合領域を同定したのち、領域ごとのリード数をカウント
  - BEDtools (Quinlan et al., 2010)中のmergeBedプログラムを実行して和集合領域同定後、intersectBedプログラムを実行してリード数をカウントする作業に相当



カウント情報取得

アノテーション情報がない場合の戦略は、複数サンプルの場合には領域が変わりうる。Cufflinksを知っているヒトはcuffmergeと同じイメージだと思えばよい

- アノテーション情報を利用する場合
  - UCSC known 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

カウント情報取得1

アノテーション情報がない場合のやり方です

- マップ後 | 出力ファイルの読み込み | Bowtie形式 (last modified 2013/06/18)
- マップ後 | 出力ファイルの読み込み | SOAP形式 (last modified 2013/06/19)
- マップ後 | 出力ファイルの読み込み | htSeqTools(Planet_2012) (last modified 2013/06/19)
- マップ後 | カウント情報取得 | について (last modified 2014/12/17)
- マップ後 | カウント情報取得 | ゲノム | アノテーション有 | QuasR(Gaidatzis_2014) (last modified 2015/02/24)
- マップ後 | カウント情報取得 | ゲノム | アノテーション無 | QuasR(Gaidatzis_2014) (last modified 2014/06/22)
- 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 2014/06/21)
- マップ後 | 配列長とカウント数の関係 (last modified 2014/07/03)
- 正規化 | について

①

マップ後 | カウント情報取得 | ゲノム | アノテーション無 | QuasR(Gaidatzis_2014)

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。

②

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
```

カウント情報取得1

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

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
in_f2 <- "ref_genome.fa"               #入力ファイル名を指定してin_f2に格納
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
out <- qAlign(in_f1, in_f2, alignmentParameter=pa
time_e <- proc.time()                  #計算時間を
time_e - time_s                        #計算時間を
out                                    #マッピング
alignmentStats(out)                    #マッピング

#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
tmpsname <- out@alignments[,2]         #サンプル名
for(i in 1:length(tmpfname)){          #サンプル数
  if(i == 1){
    k <- readGAlignments(tmpfname[i])  #BAM形式フ
```

①

「hoge – mapping_kiso1」に入力ファイルは揃っています。マッピング結果ファイルが既に存在しますが、logファイルを見て同じオプションで実行した結果があるかどうかを自動判定します。再度マッピングを実行する必要がない場合は、以前に得られていたbamファイルを入力として、速やかに①のところの作業に移行します。

R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/mapping_kiso1"
> list.files()
 [1] "mapping_single_genome1.txt"
 [2] "QuasR_log_21744443273a.txt"
 [3] "ref_genome.fa"
 [4] "ref_genome.fa.fai"
 [5] "ref_genome.fa.md5"
 [6] "ref_genome.fa.Rbowtie"
 [7] "sample_RNAseq1.fa"
 [8] "sample_RNAseq1_217479832d2f.bam"
 [9] "sample_RNAseq1_217479832d2f.bam.bai"
[10] "sample_RNAseq1_217479832d2f.bam.txt"
[11] "sample_RNAseq1_217479832d2f.bed"
[12] "sample_RNAseq1_217479832d2f_QC.pdf"
>
```

カウント情報取得1

出力ファイルは何も指定していませんが、*_range.txtという名前のファイルが作成されます。これは、.bamという名前のファイルを内部的に入力として読み込み、その文字列中の.bamを_range.txtに置換したものを出力ファイル名として自動作成しているからそうなります。

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
(mapping_single_genome1.txt):

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

```
for(i in 1:length(tmpfname)){          #サンプル数(ファイル数)分だけループを
  if(i == 1){
    k <- readGAlignments(tmpfname[i])  #BAM形式フ
  } else{
    k <- c(k, readGAlignments(tmpfname[i]))#BAM形
  }
}
m <- reduce(granges(k))                #GRangesオ

#本番(カウント情報取得)
tmp <- as.data.frame(m)                #出力ファイ
for(i in 1:length(tmpfname)){          #サンプル数
  tmpcount <- summarizeOverlaps(m, tmpfname[i])#G
  count <- assays(tmpcount)$counts     #Summarize
  colnames(count) <- tmpsname[i]       #行列count
  tmp <- cbind(tmp, count)             #保存したい
}

#ファイルに保存
out_f <- sub(".bam", "_range.txt", tmpfname[i])#
write.table(tmp, out_f, sep="\t", append=F, quote
```

R Console

```
> #ファイルに保存
> out_f <- sub(".bam", "_range.txt", tmpfname[i])#$
> write.table(tmp, out_f, sep="\t", append=F, quote$
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/mapping_kiso1"
> list.files()
 [1] "mapping_single_genome1.txt"
 [2] "QuasR_log_21744443273a.txt"
 [3] "ref_genome.fa"
 [4] "ref_genome.fa.fai"
 [5] "ref_genome.fa.md5"
 [6] "ref_genome.fa.Rbowtie"
 [7] "sample_RNAseq1.fa"
 [8] "sample_RNAseq1_217479832d2f.bam"
 [9] "sample_RNAseq1_217479832d2f.bam.bai"
[10] "sample_RNAseq1_217479832d2f.bam.txt"
[11] "sample_RNAseq1_217479832d2f.bed"
[12] "sample_RNAseq1_217479832d2f_QC.pdf"
[13] "sample_RNAseq1_217479832d2f_range.txt"
>
```

カウント情報取得1

.bedファイルと*_range.txtファイルを見比べると理解が深まるでしょう。*_range.txtファイルの一番右側の列がカウント情報です。

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

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

#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
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。
out                                    #マッピングに用いたパラメータや入
```

	A	B
1	FileName	SampleName
2	sample_RNAseq1.fa	namae

*.bed

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

*_range.txt

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

カウント情報取得2

複数のRNA-seqリードファイルのマッピングからカウントデータ取得までを一気に行う例です。

- 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 | について (last modified 2014/12/17)
- マップ後 | カウント情報取得 | ゲノム | アノテーション有 | QuasR(Gaidatzis_2014) (last modified 2015/02/24)
- マップ後 | カウント情報取得 | ゲノム | アノテーション無 | QuasR(Gaidatzis_2014) (last modified 2014/06/22)
- 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 2014/06/21)
- マップ後
- 正規化 |

①

マップ後 | カウント情報取得 | ゲノム | アノテーション無 | QuasR(Gaidatzis_201

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
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
合領域(union
「ファイル」-「

②

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
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
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

#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qAl
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
alignmentStats(out)                    #マッピング結果(alignment statistics)の表示
```

カウント情報取得2

「hoge – mapping_kiso2」に入力ファイルは揃っています。

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-se
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
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

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

#前処理(マッピング)
time_s <- proc.time()                  #計算時間
out <- qAlign(in_f1, in_f2, alignmentParameter=p
time_e <- proc.time()                  #計算時間
time_e - time_s                        #計算時間
out                                    #マッピン
alignmentStats(out)                    #マッピン

#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
tmpsname <- out@alignments[,2]         #サンプル
for(i in 1:length(tmpfname)){          #サンプル
```

R Console

```
  詳しくは 'contributors()' と入力してください。
  また、R や R のパッケージを出版物で引用する際の$
  'citation()' と入力してください。

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

> getwd()
[1] "C:/Users/kadota/Desktop/hoge/mapping_kiso2"
> list.files()
[1] "mapping_single_genome4.txt"
[2] "ref_genome.fa"
[3] "sample_RNAseq1.fa"
[4] "sample_RNAseq2.fa"
>
```

カウント情報取得2

コピペ。出力ファイル群のうち、主に取り扱うのは、カウントデータを含むファイル(hoge4_count.txt)。

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
in_f2 <- "ref_genome.fa"               #入力ファ
out_f1 <- "hoge4_count.txt"            #出力ファ
out_f2 <- "hoge4_genelength.txt"       #出力ファ
param_mapping <- "-m 1 --best --strata -v 1"#マ

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

#前処理(マッピング)
time_s <- proc.time()                  #計算時間
out <- qAlign(in_f1, in_f2, alignmentParameter=p
time_e <- proc.time()                  #計算時間
time_e - time_s                        #計算時間
out                                    #マッピン
alignmentStats(out)                    #マッピン

#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
tmpsname <- out@alignments[,2]         #サンプル
for(i in 1:length(tmpfname)){          #サンプル
```

R Console

```
> tmp <- cbind(tmp, h$width)                    #和集合$
> write.table(tmp, out_f2, sep="\t", append=F, q$
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/mapping_kiso2"
> list.files()
 [1] "hoge4_count.txt"
 [2] "hoge4_genelength.txt"
 [3] "mapping_single_genome4.txt"
 [4] "QuasR_log_201814983c7.txt"
 [5] "ref_genome.fa"
 [6] "ref_genome.fa.fai"
 [7] "ref_genome.fa.md5"
 [8] "ref_genome.fa.Rbowtie"
 [9] "sample_RNAseq1.fa"
[10] "sample_RNAseq1_2018acaf46.bam"
[11] "sample_RNAseq1_2018acaf46.bam.bai"
[12] "sample_RNAseq1_2018acaf46.bam.txt"
[13] "sample_RNAseq2.fa"
[14] "sample_RNAseq2_201843d33cbe.bam"
[15] "sample_RNAseq2_201843d33cbe.bam.bai"
[16] "sample_RNAseq2_201843d33cbe.bam.txt"
>
```

カウント情報取得2

リストファイル中で指定したサンプル名(sample1とsample2)がカウントデータ行列の列名となります

5.サンプルデータ18-20の複数のRNA-seqデータ(sample_RNAseq1.faとsample_RNAseq
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
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

```
in_f1 <- "mapping_single_genome4.txt"  #入力ファイル名を指定してin_f
in_f2 <- "ref_genome.fa"               #入力ファイル名を指定してin_f
out_f1 <- "hoge4_count.txt"            #出力ファイル名を指定してout_
out_f2 <- "hoge4_genelength.txt"       #出力ファイル名を指定してout_
param_mapping <- "-m 1 --best --strata -v 1"#マッピング時のオプショ

#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
```

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

Contents

- 先週の課題やエラーについて
  - 課題1の解説
  - アダプター配列除去（QuasR）実行時のエラーは門田のミス
  - Paired-endデータの取り扱い
- フィルタリング(filtering)
  - 基本形：リード数が同じpaired-endの場合（QuasRパッケージ）
  - 発展形：リード数が異なる場合（ShortReadパッケージ）
- アセンブル(assemble)
  - ゲノム用とトランスクリプトーム用
  - Rockhopper2（バクテリアのトランスクリプトーム用）
- マッピング（mapping）
  - 基礎、オプション、仮想データ説明
  - マッピング本番、出力ファイル形式、オプションと結果の関係
  - カウント情報取得
- 実データ解析
  - QuasRでマッピング → トリミング → 高精度なアセンブルとマッピングへ

実データ解析

アノテーション情報がない場合のやり方です

- 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(Gaidatzis_2015) (last modified 2014/06/21)
- マッピング | single-end | ゲノム | basic aligner(応用) | QuasR(Gaidatzis_2015) (last modified 2015/06/28)
- マッピング | single-end | ゲノム | splice-aware aligner | QuasR(Gaidatzis_2015) (last modified 2014/06/21)
- マッピング | paired-end | ゲノム | basic aligner(応用) | QuasR(Gaidatzis_2015) (last modified 2015/06/28)
- マップ後 | について (last modified 2013/06/19)
- マップ後 | 出力ファイル形式について (last modif
- マップ後 | 出力ファイルの読み込み | BAM形式
- マップ後 | 出力ファイルの読み込み | Bowtie形式
- マップ後 | 出力ファイルの読み込み | SOAP形式
- マップ後 | 出力ファイルの読み込み | htSeqTools
- マップ後 | カウント情報取得 | について (last mod
- マップ後 | カウント情報取得 | ゲノム | アノテーシ
- マップ後 | カウント情報取得 | ゲノム | アノテーシ
- マップ後 | カウント情報取得 | トランスクリプトーム

①

マッピング | paired-end | ゲノム | basic aligner(応用) | QuasR(Gaidatzis_2015)

NEW

QuasRパッケージを用いてpaired-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1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。mapping_paired_genome1.txtのような2行目の1列目と2列目に「マッピングしたいRNA-seqファイル名1と2」(例: sample_RNAseq_1.faとsample_RNAseq_2.fa)、そして2行目の3列目に「任意のサンプル名」(例: namae)を記載したタブ区切りテキストファイルを用意した上で行います。1行目の文字列は変えてはいけません(つまり"FileName1", "FileName2",および"SampleName"のままにしておくということです)

「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. mapping_paired_genome1.txt中のFASTQ形式ファイルを乳酸菌ゲノムにマッピングする場合:

乳酸菌RNA-seqデータSRR616268の最初の100万リード分です。SRR616268sub_1.fastq.gzは、約75MB、全リード107 bpです。SRR616268sub_2.fastq.gzは、約67MB、全リード93 bpです。Ensembl (Flicek et al., 2014)から提供されている Lactobacillus casei 12Aの multi-FASTA形式ゲノム配列ファイル (Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa) がリファレンス配列です。マッピングオプションはデフォルトです。

```
in_f1 <- "mapping_paired_genome1.txt"  #入力ファイル名を指定してin_f1に格納(RNA-seqファイ
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa"#入
#必要なパッケージをロード
time_s <- proc.time()                  #計算時間を計測するため
out <- qAlign(in_f1, in_f2)            #マッピングを行うqAlign関数を実行した結果をoutに格
```

	A	B	C
1	FileName1	FileName2	SampleName
2	SRR616268sub_1.fastq.gz	SRR616268sub_2.fastq.gz	namae_paired

実データ解析

「hoge – mapping_paired1」に解析したいfastq.gzファイルを移動させて実行。約8分。

1. mapping_paired_genome1.txt中のFASTQ形式ファイルを乳酸菌ゲノムにマッピングする場合:

乳酸菌RNA-seqデータSRR616268の最初の100万リード分です。SRR616268sub_1.fastq.gzは、約75MB、全リード107 bpです。SRR616268sub_2.fastq.gzは、約67MB、全リード93 bpです。Ensembl (Flicek et al., 2014)から提供されているLactobacillus casei 12Aの multi-FASTA形式ゲノム配列ファイル(Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa)がリファレンス配列です。マッピングオプションはデフォルトです。

```
in_f1 <- "mapping_paired_genome1.txt"  #入力ファイル名を指定してin_f1に格納(RNA-seqファイ
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa"#入

#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

#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)            #マッピングを行うqAlign関数を実行した結果をoutに格
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
alignmentStats(out)                    #マッピング結果(alignment statistics)の表示。seq
```

```
#ファイルに保存
out_f <- sub("
qQCReport(out,
out_f

#ファイルに保存
tmpfname <- ou
```

R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge/mapping_paired1"
> list.files()
[1] "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa"
[2] "mapping_paired_genome1.txt"
[3] "SRR616268sub_1.fastq.gz"
[4] "SRR616268sub_2.fastq.gz"
>
```

実データ解析

①このあたりで約1分。リファレンス配列のインデックス作成部分。約3MBの乳酸菌ゲノムなので短時間で終わる。ヒトゲノムだと数十分から数時間といったところか。②のところで約7分。オリジナルの総リード数は1.4億。1/100以下の100万リードだから10分足らずで終わる。

1. mapping_paired_genome1.txt中のFASTQ形式ファイルを乳酸菌ゲノムにマッピング

乳酸菌RNA-seqデータSRR616268の最初の100万リード分です。SRR616268sub_1.fast
107 bpです。SRR616268sub_2.fastq.gzは、約67MB、全リード93 bpです。Ensembl (Flic
ている Lactobacillus casei 12Aの multi-FASTA形式ゲノム配列ファイル
(Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa) がリフ
グオプションはデフォルトです。

```
in_f1 <- "mapping_paired_genome1.txt"  #入力ファイル名を指定してin_f1に格納(RNA-seqファイ
in_f2 <- "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa"#入

#必要なパッケー
library(QuasR)
library(Genomi

#本番(マッピン
time_s <- proc
out <- qAlign(
time_e <- proc
time_e - time_
out
alignmentStats

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

#ファイルに保存
tmpfname <- ou
```

R Console

```
> #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)              #マッピングを行うqAlign関数を実行し$
Creating .fai file for: C:/Users/kadota/Desktop/hoge/mapping_paired1/Lactobac$
alignment files missing - need to:
    create alignment index for the genome
    create 1 genomic alignment(s)
will start in ..9s..8s..7s..6s..5s..4s..3s..2s..1s
Creating an Rbowtie index for C:/Users/kadota/Desktop/hoge/mapping_paired1/La$
Finished creating index
Testing the compute nodes...OK
Loading QuasR on the compute nodes...OK
Available cores:
nodeNames
KADOTA-PC
        1
Performing genomic alignments for 1 samples. See progress in the log file:
```

①

②

実データ解析

①マッピングの実体であるqAlign関数実行前後の時間を計測し、423.14/60 = 7.05分という結果を得ている。

```
#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)            #マッピングを行うqAlign関数を実行した結果をoutに格
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
alignmentStats(out)                    #マッピング結果(alignment statistics)の表示。seq

#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])#Quality Controlレポートのpdfファイ
qQCReport(out, p
out_f

#ファイルに保存(B
tmpfname <- out@
for(i in 1:lengt
  hoge <- readGA
  hoge <- as.dat
  tmp <- hoge[,
  out_f <- sub("
  out_f
  write.table(tm
```

R Console

```
Performing genomic alignments for 1 samples. See progress in the log file:
C:/Users/kadota/Desktop/hoge/mapping_paired1\QuasR_log_d0470eb4872.txt
Genomic alignments have been created successfully

> 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$
   ユーザ    システム          経過
      0.92        0.76      423.14 ①
> out                                  #マッピングに用いたパラメータや入力$
Project: qProject
 Options    : maxHits          : 1
              paired           : fr
              splicedAlignment: FALSE
              bisulfite        : no
              snpFile          : none
 Aligner    : Rbowtie v1.8.0 (parameters: -m 1 --best --strata --maxins 500)
 Genome     : .../Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Ch$
```

実データ解析

①マッピングに用いられたオプション(パラメータ)情報。②これがメインのマッピング結果ファイルの名前(いわゆるBAMファイル)。③マッピング結果の概観。forward側とreverse側合わせて200万リード中8,204リードしかマップされてない!

```
#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)              #マッピングを行うqAlign関数を
time_e <- proc.time()                    #計算時間を計測するため
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
tmpfname <- out@
for(i in 1:lengt
  hoge <- readGA
  hoge <- as.dat
  tmp <- hoge[,
  out_f <- sub("
  out_f
  write.table(tm
```

R Console

```
> out                                    #マッピングに用いたパラメータや入力$
Project: qProject
 Options    : maxHits          : 1
              paired           : fr
              splicedAlignment: FALSE
              bisulfite        : no
              snpFile          : none
 Aligner    : Rbowtie v1.8.0 (parameters: -m 1 --best --strata --maxins 500)
 Genome     : .../Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.C$

 Reads      : 1 pair of files, 1 sample (fastq format):
   1. SRR616268sub_1.fastq.gz  SRR616268sub_2.fastq.gz  namae_paired (phred33)

 Genome alignments: directory: same as reads
   1. SRR616268sub_1_d0466162946.bam

 Aux. alignments: none

> alignmentStats(out)                    #マッピング結果(alignment statistics$
                    seqlength mapped unmapped
namae_paired:genome   2907892   8204  1991796
>
```

実データ解析

```
#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)              #マッピングを行うqAlign関数を実行し
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
out                                      #マッピングに用いたパラメータや入力
alignmentStats(out)                      #マッピング結果(alignment statist

#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
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)
tmpfname <- out@alignments[,1]           #ファイル名(in_f1の1列目に相当)をtmpfnameとして取
```

①list.filesを用いて、ここまでの作業で作成されたファイルを概観。元は黒枠の4つのファイルのみだったが、マッピングが終わった時点で数多くのファイルが作成されていることが分かる。②マッピング結果を取扱うときの基本はBAMファイル。③マッピングに用いられたオプション(パラメータ)情報は、このlogファイルからも閲覧可能。

R Console

①

```
> list.files()
 [1] "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa"
 [2] "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa.fai"
 [3] "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa.md5"
 [4] "Lactobacillus_casei_12a.GCA_000309565.2.25.dna.chromosome.Chromosome.fa.Rbowtie"
 [5] "mapping_paired_genome1.txt"
 [6] "QuasR_log_d0470eb4872.txt"
 [7] "SRR616268sub_1.fastq.gz"
 [8] "SRR616268sub_1_d0466162946.bam"
 [9] "SRR616268sub_1_d0466162946.bam.bai"
[10] "SRR616268sub_1_d0466162946.bam.txt"
[11] "SRR616268sub_2.fastq.gz"
>
```

③ ②

QCレポート作成

```
#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
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)
tmpfname <- out@alignments[,1]          #ファイル名(in_f1の1列目に相当)をtmpf
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
  hoge <- readGAlignments(tmpfname[i]) #BAM形式ファイルを読み込んだ結果をhogeに格納(これは
  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結果をtmpに格納
  out_f <- sub(".bam", ".bed", tmpfname[i])#BED形式ファイル名を作成した結果をout_fに格納
  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.names=F)#tmpの
}
```

①sub関数を用いて黒下線で見られるファイル名中の拡張子部分.bamを_QC.pdfに置換したファイル名を作成してout_fに格納している。拡張子部分周辺の文字列だけを変更する程度の出力ファイル名であれば、このように自動的に作成することができる。数十秒程度。

R Console

```
> out@alignments[,1]
[1] "C:/Users/kadota/Desktop/hoge/mapping_paired1\\SRR616268sub_1_d0466162946.bam"
> out_f <- sub(".bam", "_QC.pdf", out@alignments[,1])#Quqlity Controlレポートのpdfフ$
> 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/mapping_paired1\\SRR616268sub_1_d0466162946_QC.pdf"
>
```

①

BEDファイル作成

```
#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に
out_f                                    #ファイル名を表示してるだけ
```

```
#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
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
  hoge <- readGAlignments(tmpfname[i]) #BAM形式ファイルを読み込んだ結果をhogeに格納(
  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結果をtmpに格納
  out_f <- sub(".bam", ".bed", tmpfname[i])#BED形式ファイル名を作成した結果をout_fに格納
  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.names=F)#tmpの
}
```

①これはBAMファイルがバイナリ形式でそんなものだと慣れてしまえば不必要。単純に視覚的に見やすいようにbamファイルを読み込んでBED形式ファイルを作成しているところ（つまりファイル形式の変換）。②得られたBED形式ファイルの中身。マップされたリード数が8,204個であったことから、BEDファイルの行数が8,204行なのは妥当

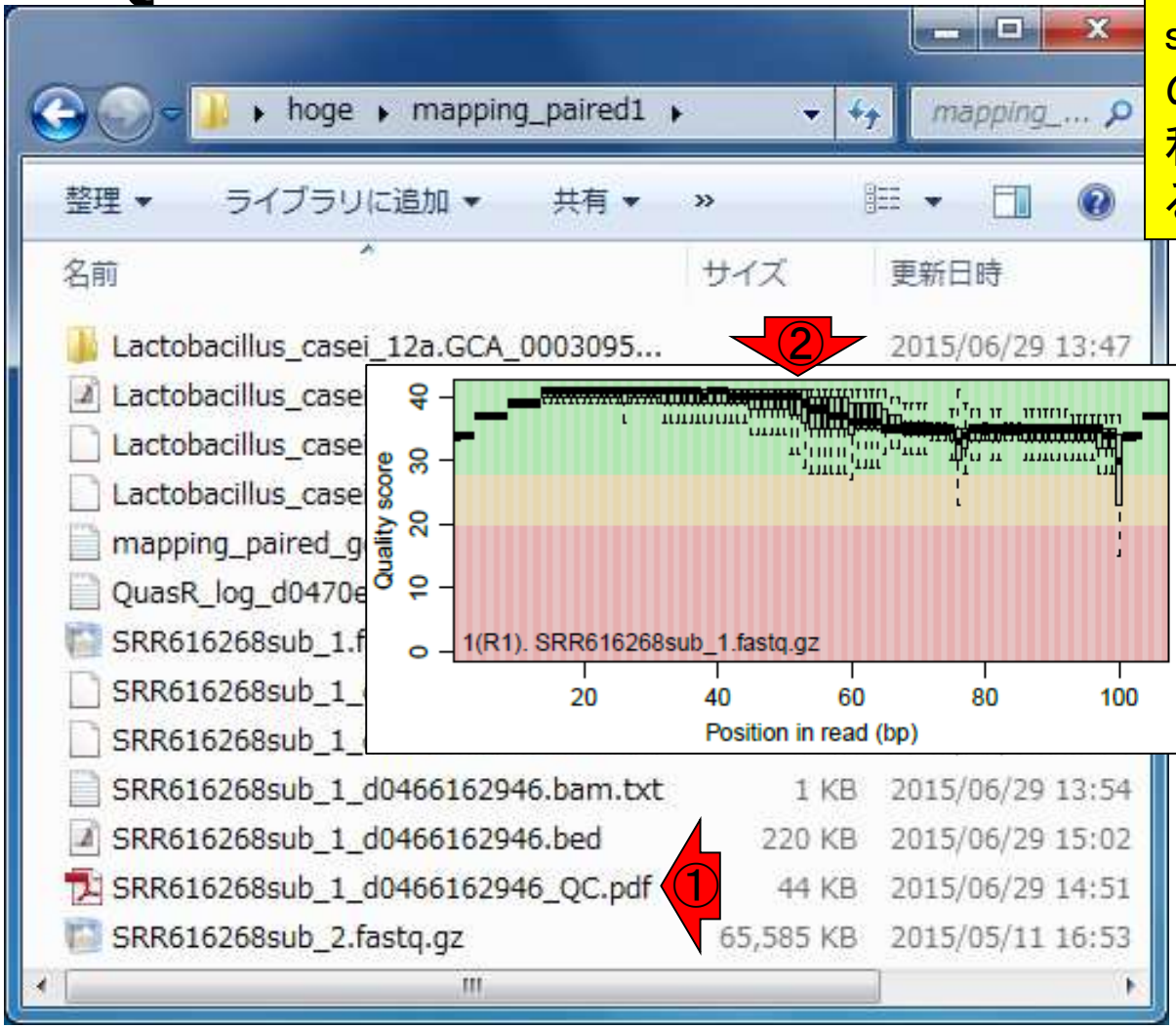
①

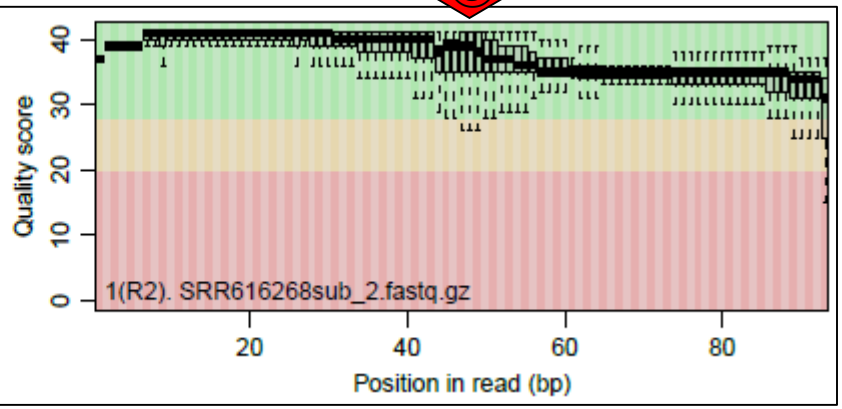
②

	A	B	C
1	Chromosome	592	684
2	Chromosome	634	740
3	Chromosome	971	1063
4	Chromosome	1032	1138
5	Chromosome	1105	1197
6	Chromosome	1289	1395
8198	Chromosome	2906800	2906972
8199	Chromosome	2906881	2906973
8200	Chromosome	2906897	2906989
8201	Chromosome	2906904	2906996
8202	Chromosome	2906919	2907011
8203	Chromosome	2907095	2907187
8204	Chromosome	2907108	2907214

QCレポート概観

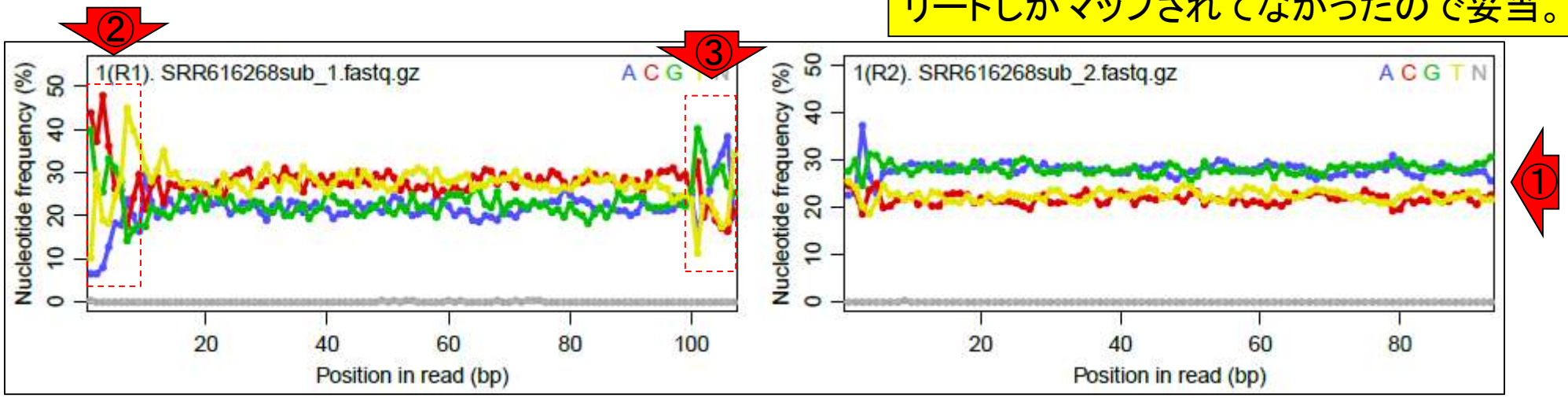
①QCレポートを眺める。最初のページはquality score分布。②forward側と③reverse側で左右に表示されている。カイコsmall RNA-seqデータ（single-end)で左側のみにしか表示されていなかったときは違和感を覚えたが、paired-endデータで眺めると至極妥当な配置であったことが分かる

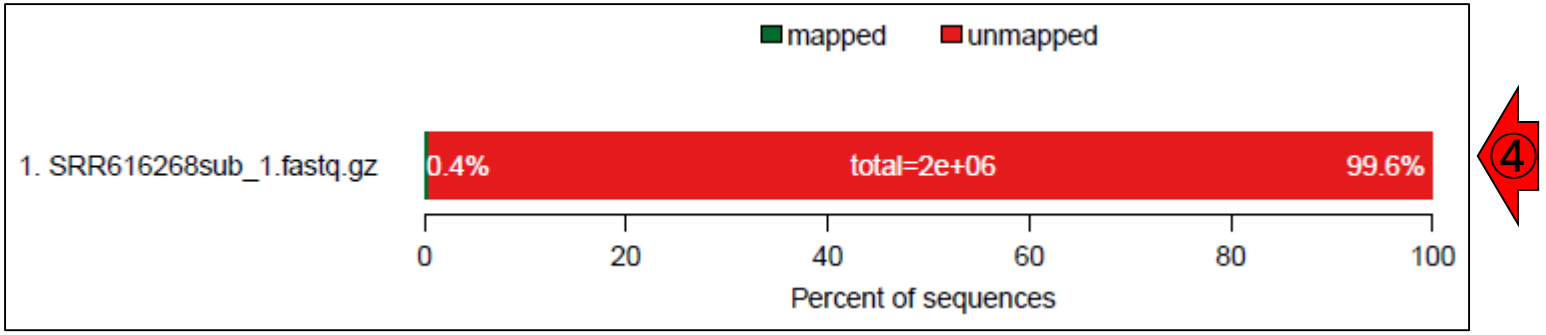




QCレポート概観

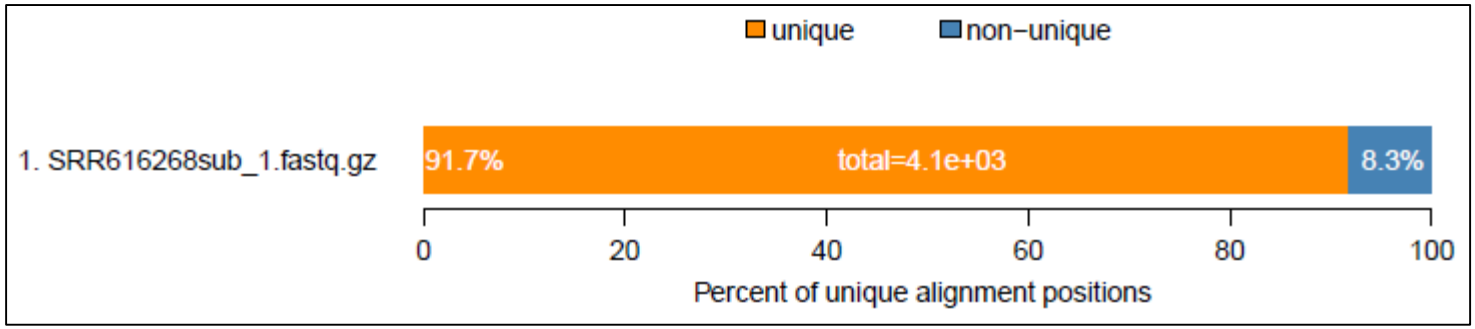
①QCレポートの2page目。②Forwardのリードの左側は確かにガタガタしていたので妥当だが、③右側もこんなになってたっけ?!と思いつつ…(後にこれが犯人と判明)。④4page目。リード全体のmapped/unmappedの割合を表示。確かに200万リード中8,204リードしかマップされてなかったので妥当。

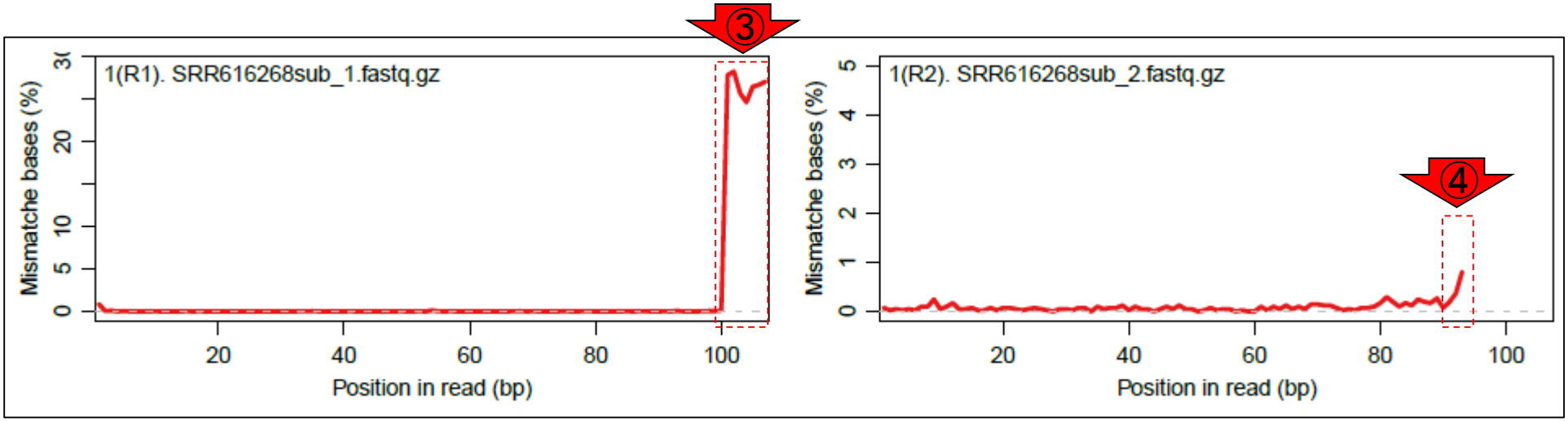




QCレポート概観

①5page目。200万リード中8,204リードマップされていたが、paired-endなので、ここでは分母を8,204/2 = 4,102≒4.1e+03と表現しているようだ。1ヶ所にのみマップされたのは全体の91.7%。②6page目。これはおそらくマップされたリードの中でどのポジションにミスマッチがあったかを示す部分。③と④を眺めて、確かにFastQCで眺めていたのは最初の50 bp分で、adapters/primersの除去は5’側のみだった!





アセンブル結果再考

このぶざまな状況を打破する糸口が完璧に見つかった！

ファイル名	Total number of assembled transcripts	Average length	Median length	Total number of assembled bases	Total reads in file	Perfectly aligned reads
・100万リードのオリジナル?!ファイル						
SRR616268sub_1.fastq.gz	8	121	106	974	996739	1964
SRR616268sub_2.fastq.gz	424	436	228	185233	983854	710393
paired-end	0	0	0	0	990047	23
・100万リードのアダプター除去後のファイル						
SRR616268sub_trim2_1.fastq.gz	8	121	104	974	993311	1231
SRR616268sub_trim2_2.fastq.gz	425	433	228	184437	981649	708951
hoge2_1.fastq.gz	8	121	104	974	993001	1231
hoge2_2.fastq.gz	424	434	229	184210	980997	708395
paired-end	0	0	0	0	989869	25
・アダプター除去およびフィルタリング後のファイル						
SRR616268sub_trim_1.fastq.gz	8	121	104	974	993311	1231
SRR616268sub_trim_2.fastq.gz	425	433	228	184437	981649	708951
hoge1_1.fastq.gz	8	121	104	974	993081	1231
hoge1_2.fastq.gz	423	435	229	184211	981125	708482
paired-end	0	0	0	0	989869	25

Contents

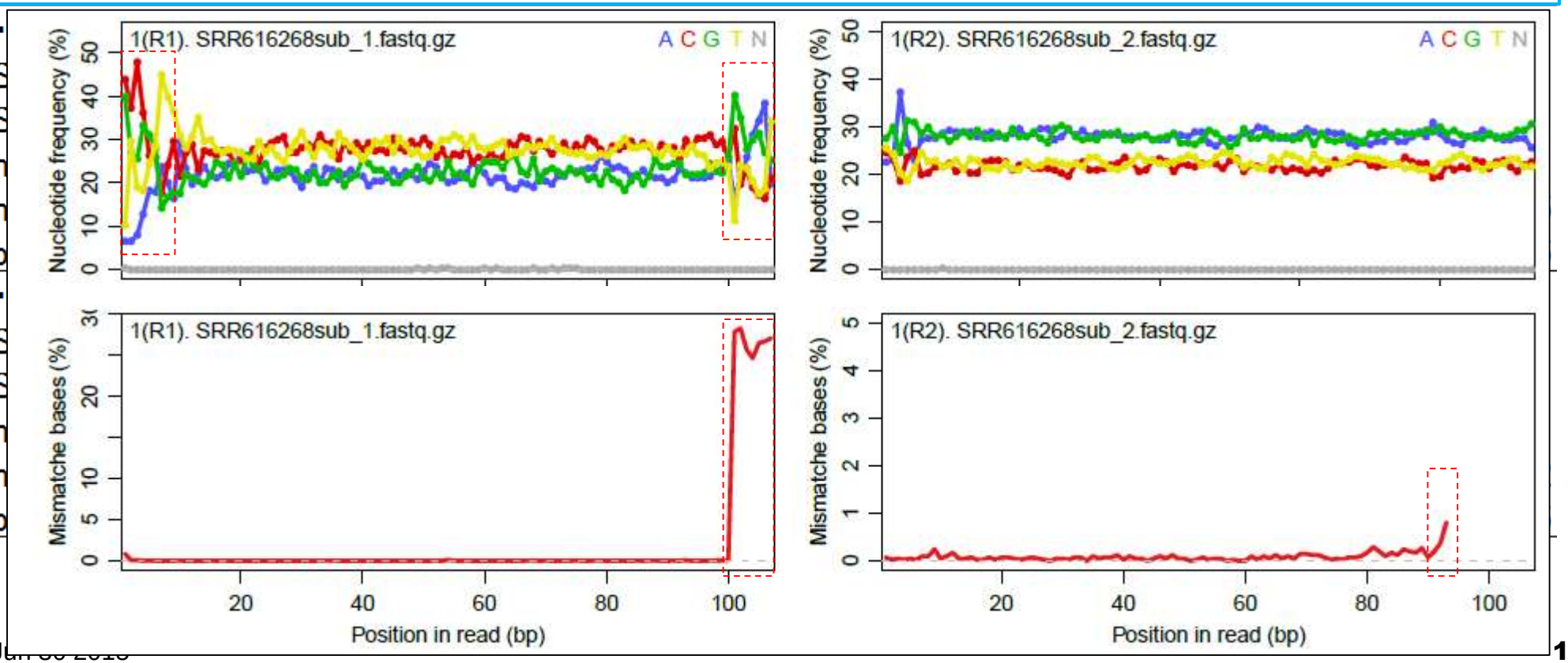
- 先週の課題やエラーについて
  - 課題1の解説
  - アダプター配列除去（QuasR）実行時のエラーは門田のミス
  - Paired-endデータの取り扱い
- フィルタリング(filtering)
  - 基本形：リード数が同じpaired-endの場合（QuasRパッケージ）
  - 発展形：リード数が異なる場合（ShortReadパッケージ）
- アセンブル(assemble)
  - ゲノム用とトランスクリプトーム用
  - Rockhopper2（バクテリアのトランスクリプトーム用）
- マッピング（mapping）
  - 基礎、オプション、仮想データ説明
  - マッピング本番、出力ファイル形式、オプションと結果の関係
  - カウント情報取得
- 実データ解析
  - QuasRでマッピング → トリミング → 高精度なアセンブルとマッピングへ

課題1, 2, 3

課題1:「Rockhopperのアセンブル結果」を「QuasRマッピング結果のQCレポート」と絡めて説明せよ。**課題2**:どうすればマッピング結果を改善できるのか(マップ率を上げることができるのか)、考えられる戦略を述べよ。**課題3**:今回の結果を踏まえ、リファレンスゲノムがない(マッピング結果がない)場合の高精度なアセンブル戦略について述べよ。

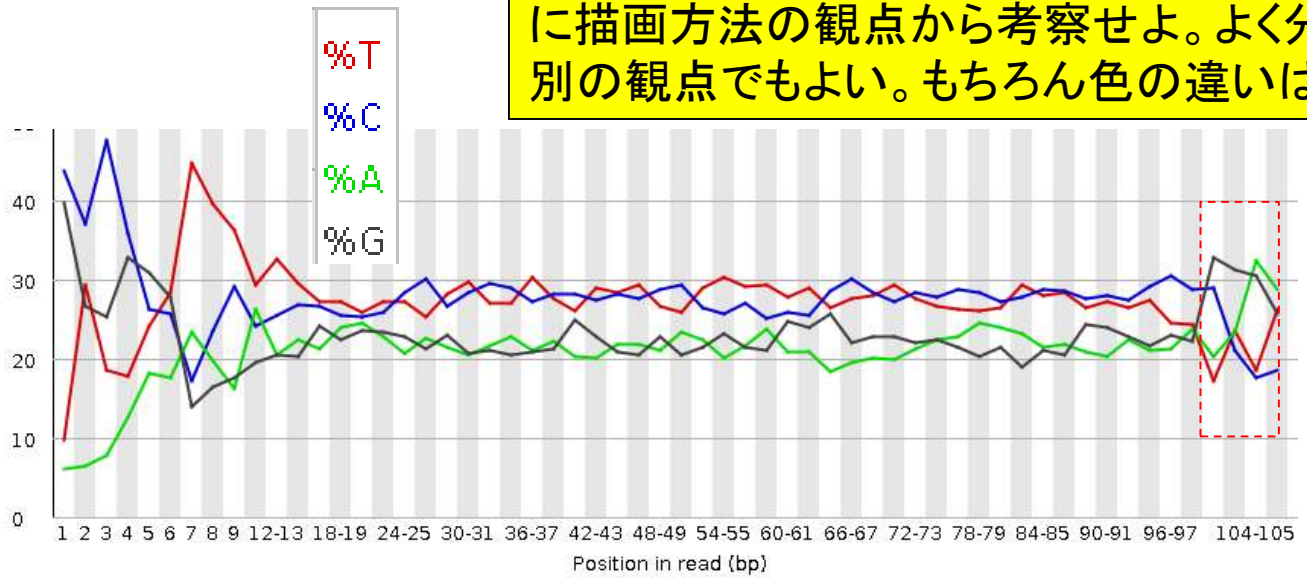
・100万リードのオリジナル?!ファイル

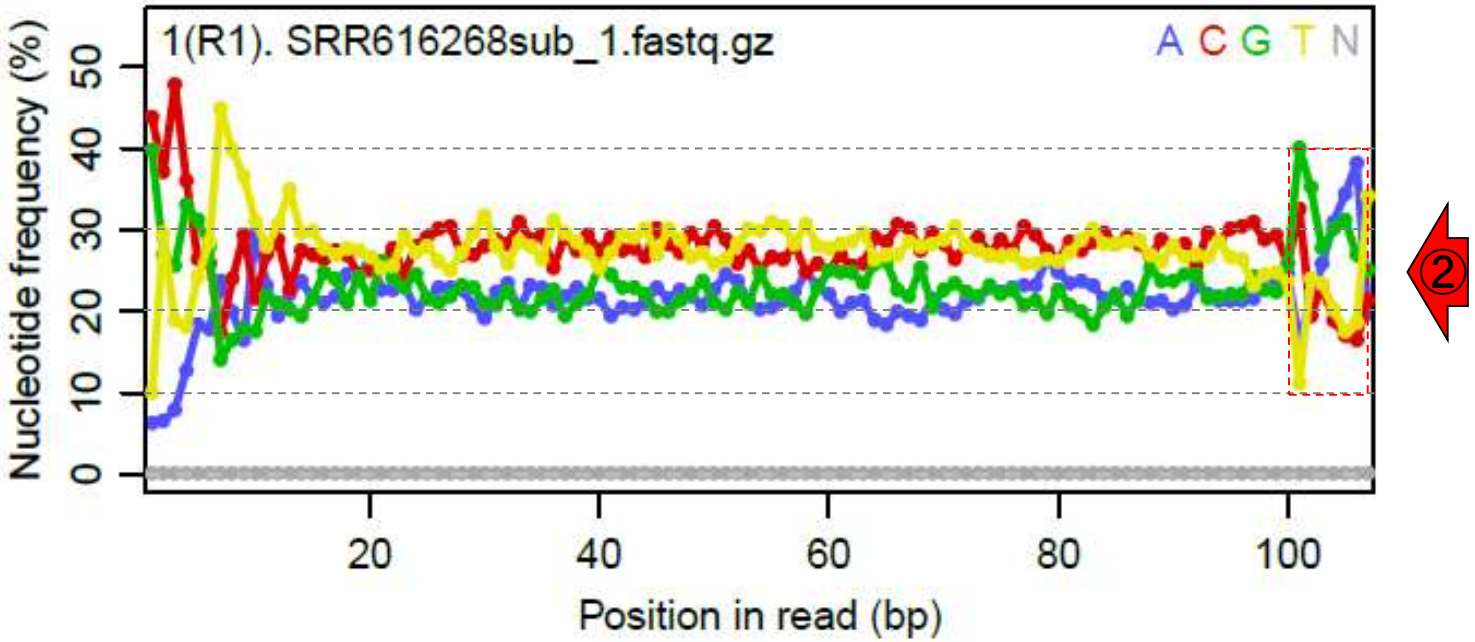
ファイル名	Total numb of assembl transcripts					
SRR616268sub_1.fastq.gz	8	121	106	974	996739	1964
SRR616268sub_2.fastq.gz	424	436	228	185233	983854	710393
paired-end	0	0	0	0	990047	23



課題4

課題4：①はFastQCの「Per base sequence content」、②はQuasRのQCレポートファイル中の同様な結果である。赤点線枠部分に違いが見られるが、この理由について主に描画方法の観点から考察せよ。よく分からないヒトは別の観点でもよい。もちろん色の違いは本質的ではない





Rockhopperリトライ

これだけ結果が変わります。
かなりイケてます。

ファイル名	Total number of assembled transcripts	Average length	Median length	Total number of assembled bases	Total reads in file	Perfectly aligned reads
・100万リードのオリジナル?!ファイル（トリム後）						
SRR616268sub_1.fastq.gz	738	250	154	184550	994562	635565
SRR616268sub_2.fastq.gz	401	442	228	177642	983854	709473
paired-end	776	577	311	447950	987886	584728
・100万リードのアダプター除去後のファイル（トリム後）						
SRR616268sub_trim2_1.fastq.gz						
SRR616268sub_trim2_2.fastq.gz	388	449	229	174454	981649	706952
hoge2_1.fastq.gz						
hoge2_2.fastq.gz						
paired-end						
・アダプター除去およびフィルタリング後のファイル						
SRR616268sub_trim_1.fastq.gz						
SRR616268sub_trim_2.fastq.gz						
hoge1_1.fastq.gz						
hoge1_2.fastq.gz						
paired-end						

QuasRリトライ

これだけ結果が変わります。
かなりイケてます。