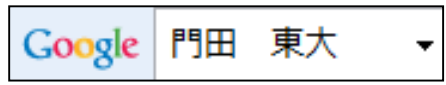
バイオインフォマティクス人材育成カリキュラム（次世代シークエンサ）速習コース
3. データ解析基礎 | 3-5. R Bioconductor II

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



Contents

- 3-4. R Bioconductor II、2014/09/09 15:00-18:15、中級、実習
  - **multi-FASTAファイルからの情報抽出（コンティグ数、総塩基数、N50、GC含量）**
    - GC含量計算の詳細説明。alphabetFrequency, apply関数、数値行列計算の基本
    - コンティグごとのGC含量計算
  - FASTQ形式ファイルの読み込み
  - ファイル形式の変換：FASTQ → FASTA
  - クオリティチェック(クオリティコントロール；QC)
  - フィルタリング
    - クオリティスコア、N、配列長など
    - 動作確認用のサブセット作成
  - その他（FASTA/FASTQファイルのdescription行を整形）

multi-FASTAファイルからの各種情報抽出

翻訳配列取得以外にも
様々な解析が可能です

(Rで)塩基配列解析
~NGS、RNA-seq、ゲノム、トランスクリプトーム、正規化、発現変動、統計、モデル、バイオ

- イントロ | NGS | アノテーション情報取得 | TranscriptDb | GenomicFeatures(Lawrence 2013)
- イントロ | NGS | アノテーション情報取得 | TranscriptDb | GFF/GTF形式ファイルから (last mod
- イントロ | NGS | 読み込み | FASTA形式 | 基本情報を取得 (last modified 2014/05/29)
- イントロ | NGS | 読み込み | FASTA形式 | description行の記述を整形 (last modified 2014/04/
- イントロ | NGS | 読み込み | FASTQ形
- イントロ | NGS | 読み込み | FASTQ形
- イントロ | NGS | 読み込み | Illumina(
- イントロ | NGS | 読み込み | Illumina(
- イントロ | ファイル形式の変換 | につ
- イントロ | ファイル形式の変換 | BAM
- イントロ | ファイル形式の変換 | FAST
- イントロ | ファイル形式の変換 | Genb
- イントロ | ファイル形式の変換 | qseq
- イントロ | ファイル形式の変換 | qseq
- イントロ | ファイル形式の変換 | qseq
- 前処理 | クオリティチェック | について
- 前処理 | クオリティチェック | qrqc (las
- 前処理 | クオリティチェック | PHRED

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

multi-FASTAファイルを読み込んで、Total lengthやaverage lengthなどの各種情報取得を行うためのやり方を示します。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

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

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

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

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

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

#本番(N50情報取得)
sorted <- rev(sort(width(fasta)))           #長さ情報を降順にソートした結果をsortedに格納
obj <- (cumsum(sorted) >= Total_len*0.5)#条件を満たすかどうかを判定した結果をobjに格納(長い
N50 <- sorted[obj][1]                       #objがTRUEとなる1番最初の要素のみ抽出した結果をN50に
```

入力ファイルの説明

手計算での検証可能なレベルの入力ファイル

ID	A	C	G	T	配列長	CG	ACGT	%GC含量
contig_1	4	9	7	4	24	16	24	66.6667
contig_2	20	34	31	18	103	65	103	63.1068
contig_3	16	13	20	16	65	33	65	50.7692
contig_4	14	15	10	10	49	25	49	51.0204
Total	54	71	68	48	241	139	241	57.6763

イントロ | NGS | 読み込み | FASTA形式 | 基本情報を取得

multi-FASTAファイルを読み込んで、Total lengthやaverage lengthなどの各種情報取得
「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

1. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたmulti-FASTA

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

#必要なパッケージ読み込み
library(Bi

#入力ファイ
fasta <- r

#本番(基本情
Total_len
Number_of_
Average_le
Median_len
Max_len <-
Min_len <-

#本番(N50情
sorted <-
obj <- (cu
N50 <- sor
```

hoge4.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

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

入力ファイルの説明

イントロ | NGS | 読み込み | FASTA形式 | 基本情報を取得

multi-FASTAファイルを読み込んで、Total lengthやaverage lengthなどの各種情報取得
「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

1. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたmulti-FASTA

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

#必要なパッケージ読み込み
library(Bi

#入力ファイ
fasta <- r

#本番(基本情
Total_len
Number_of_
Average_le
Median_len
Max_len <-
Min_len <-

#本番(N50情
sorted <-
obj <- (cu
N50 <- sor
```

hoge4.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

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

手計算での検証可能なレベルの入力ファイル

ID	A	C	G	T	配列長	CG	ACGT	%GC含量
contig_1	4	9	7	4	24	16	24	66.6667
contig_2	20	34	31	18	103	65	103	63.1068
contig_3	16	13	20	16	65	33	65	50.7692
contig_4	14	15	10	10	49	25	49	51.0204
Total	54	71	68	48	241	139	241	57.6763

contig_1の配列長は24塩基です。その中にはAが4つあります。この入力ファイルは、NなどのACGT以外の文字が1つもないので**配列長**列と**ACGT**列の数値が同じになっています。

基本はコピペ

イントロ | NGS | 読み込み | FASTA形式 | 基本情報を取得

multi-FASTAファイルを読み込んで、Total lengthやaverage lengthなどの各種情報取得
「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

1. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたmulti-FASTA

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

#必要なパッケージをロー
library(Biostrings)                   #FASTA形式の読み込み

#入力ファイルの読み込み
fasta <- readDNAString

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

#本番(N50情報取得)
sorted <- rev(sort(wid
obj <- (cumsum(sorted) >= Total_len*0.5)#条件を
N50 <- sorted[obj][1]                 #objがTR
```

切り取り(T)
コピー(C) ①
貼り付け
すべて選択(A)
印刷(I)...
印刷プレビュー(N)...
Bing でマップ
Bing で翻訳
Google で検索
電子メール (Windows Live Ho
すべてのアクセラレータ
Send to OneNote

手計算での検証可能なレベルの入力ファイル

ID	A	C	G	T	配列長	CG	ACGT	%GC含量
contig_1	4	9	7	4	24	16	24	66.6667
contig_2	20	34	31	18	103	65	103	63.1068
contig_3	16	13	20	16	65	33	65	50.7692
contig_4	14	15	10	10	49	25	49	51.0204
Total	54	71	68	48	241	139	241	57.6763

①一連のコマンド群をコピーして
②R Console画面上でペースト

R Console

```
一定の条件に従えば、自由にこれを再配布することができ$
配布条件の詳細に関しては、'license()' あるいは 'lice$

R は多くの貢献者による共同プロ
詳しくは 'contributors()
また、R や R のパッケージを出版
'citation()' と入力してく

'demo()' と入力すればデモを
'help()' とすればオンライン
'help.start()' で HTM
'q()' と入力すれば R を終
```

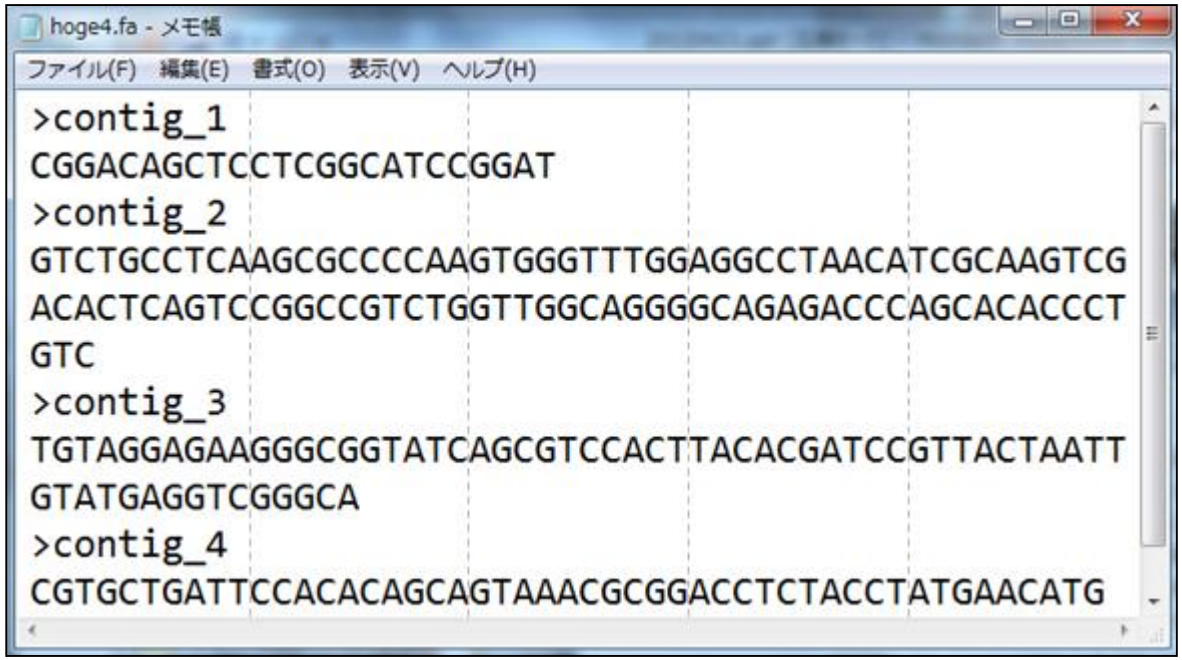
コピー Ctrl+C
ペースト ② Ctrl+V
コマンドのみペースト
コピー&ペースト Ctrl+X
ウインドウの消去 Ctrl+L
全て選択
✓ バッファに出力 Ctrl+W
ウィンドウを常にトップに置く

解析結果

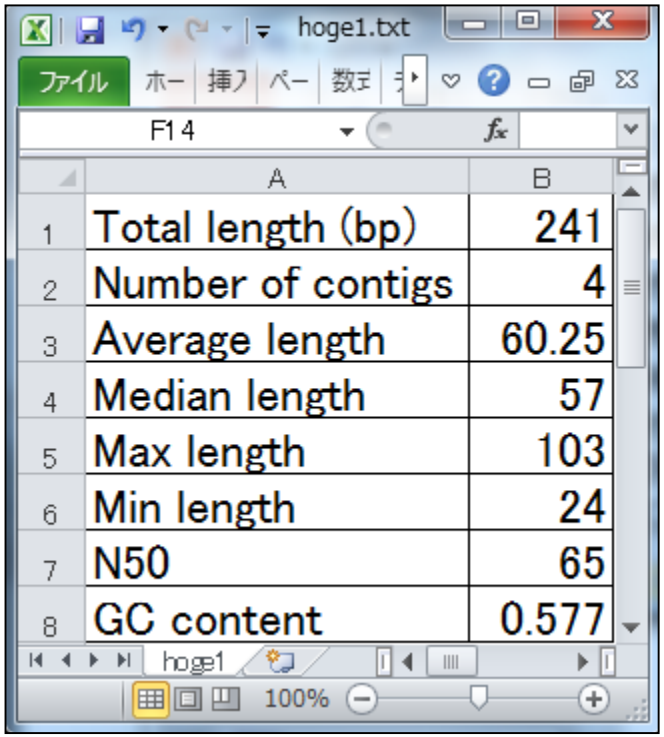
手計算での検証可能なレベルの入力ファイル

ID	A	C	G	T	配列長	CG	ACGT	%GC含量
contig_1	4	9	7	4	24	16	24	66.6667
contig_2	20	34	31	18	103	65	103	63.1068
contig_3	16	13	20	16	65	33	65	50.7692
contig_4	14	15	10	10	49	25	49	51.0204
Total	54	71	68	48	241	139	241	57.6763

入力：hoge4.fa

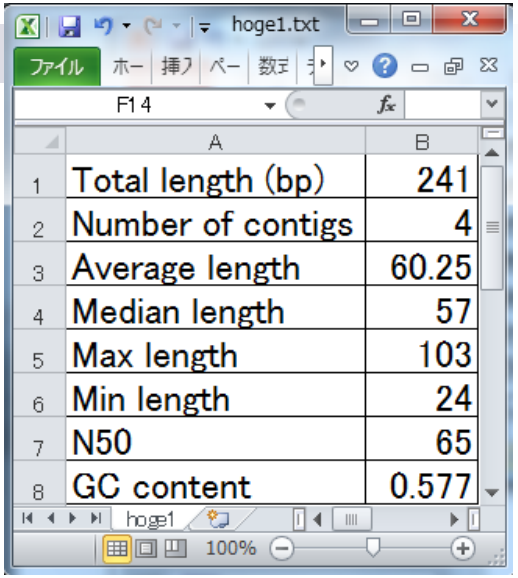


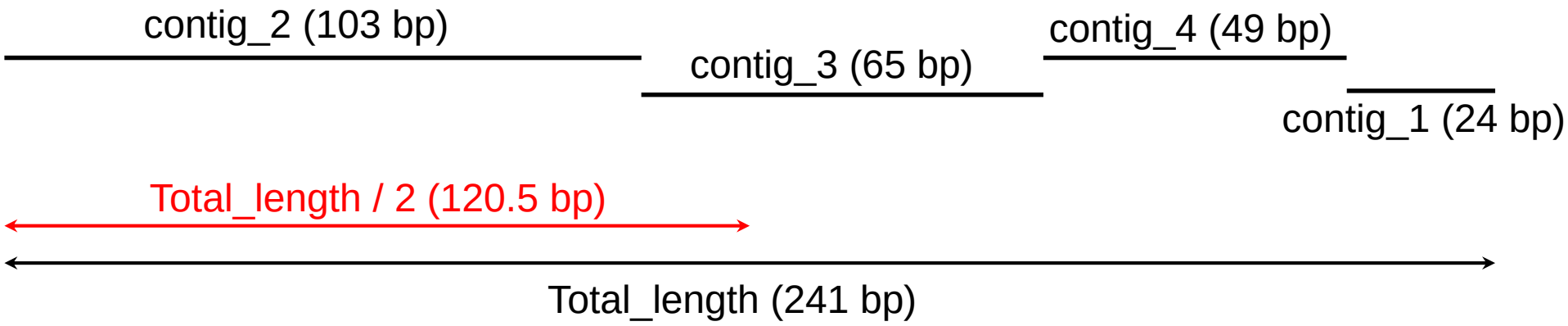
出力: hoge1.txt



N50

- アセンブル結果の評価基準の一つ
  - 長いコンティグから足していってTotal_lengthの50%に達したときのコンティグの長さ
  - 一般に数値が大きいほどよい





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

Contents

- 3-4. R Bioconductor II、2014/09/09 15:00-18:15、中級、実習
  - multi-FASTAファイルからの情報抽出（コンティグ数、総塩基数、N50、GC含量）
    - **GC含量計算の詳細説明。alphabetFrequency, apply関数、数値行列計算の基本**
    - コンティグごとのGC含量計算
  - FASTQ形式ファイルの読み込み
  - ファイル形式の変換：FASTQ → FASTA
  - クオリティチェック(クオリティコントロール；QC)
  - フィルタリング
    - クオリティスコア、N、配列長など
    - 動作確認用のサブセット作成
  - その他（FASTA/FASTQファイルのdescription行を整形）

GC含量計算の詳細説明

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

```
sorted <- rev(sort(width(fasta)))       #長さ情報を降順にソートした結果をsortedに格納
obj <- (cumsum(sorted) >= Total_len*0.5)#条件を満たすかどうかを判定した結果をobjに格納(長い
N50 <- sorted[obj][1]                   #objがTRUEとなる1番最初の要素のみ抽出した結果をN50に

#本番(GC含量情報取得)
hoge <- alphabetFrequency(fasta)        #A,C,G,T,..の数を配列ごとにカウントした結果をhogeに
CG <- rowSums(hoge[,2:3])               #C,Gの総数を計算してCGに格納
ACGT <- rowSums(hoge[,1:4])             #A,C,G,Tの総数を計算してACGTに格納
GC_content <- sum(CG)/sum(ACGT)         #トータルのGC含量の情報を取得

#ファイルに保存
tmp <- NULL
tmp <- rbind(tmp, c("Total length (bp)", Total_len))
tmp <- rbind(tmp, c("Number of contigs", Number_of_contigs))
tmp <- rbind(tmp, c("Average length", Average_len))
tmp <- rbind(tmp, c("Median length", Median_len))
tmp <- rbind(tmp, c("Max length", Max_len))
tmp <- rbind(tmp, c("Min length", Min_len))
tmp <- rbind(tmp, c("N50", N50))
tmp <- rbind(tmp, c("GC content", GC_content))
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)#tmpの中身を指定したフ
```

ID	A	C	G	T	配列長	CG	ACGT	%GC含量
contig_1	4	9	7	4	24	16	24	66.6667
contig_2	20	34	31	18	103	65	103	63.1068
contig_3	16	13	20	16	65	33	65	50.7692
contig_4	14	15	10	10	49	25	49	51.0204
Total	54	71	68	48	241	139	241	57.6763

GC含量(CとGが含まれる割合)は、塩基の種類ごとのカウント数が分かれば四則演算で計算可能です。

1. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたmul

```
sorted <- rev(sort(width(fasta)))        #長さ情報を降順に
obj <- (cumsum(sorted) >= Total_len*0.5)#条件を満たすか
N50 <- sorted[obj][1]                    #objがTRUEとなる
```

```
#本番(GC含量情報取得)
hoge <- alphabetFrequency(fasta)         #A,C,G,T,..の数を
CG <- rowSums(hoge[,2:3])                #C,Gの総数を計算してCGに格納
ACGT <- rowSums(hoge[,1:4])              #A,C,G,Tの総数を計算してACGTに格納
GC_content <- sum(CG)/sum(ACGT)          #トータルのGC含量の情報を取得
```

```
#ファイルに保存
tmp <- NULL
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
write.table(t
```

ID	A	C	G	T	配列長	CG	ACGT	%GC含量
contig_1	4	9	7	4	24	16	24	66.6667
contig_2	20	34	31	18	103	65	103	63.1068
contig_3	16	13	20	16	65	33	65	50.7692
contig_4	14	15	10	10	49	25	49	51.0204
Total	54	71	68	48	241	139	241	57.6763

alphabetFrequency関数を実行すると、コンティグごとにA, C, G, Tなど塩基の種類ごとの出現数を数値行列として得ることができます。

R Console

```
> fasta
  A DNAStringSet instance of length 4
    width seq                                          names               $
[1]    24 CGGACAGCTCCTCGGCATCCGGAT                     contig_1
[2]   103 GTCTGCCTCAAGCGC...CCAGCACACCCTGTC            contig_2
[3]    65 TGTAGGAGAAGGGCG...GTATGAGGTCGGGCA            contig_3
[4]    49 CGTGCTGATTCCACA...CTACCTATGAACATG            contig_4
> alphabetFrequency(fasta)
     A  C  G  T M R W S Y K V H D B N - + .
[1,] 4  9  7  4 0 0 0 0 0 0 0 0 0 0 0 0 0
[2,] 20 34 31 18 0 0 0 0 0 0 0 0 0 0 0 0 0
[3,] 16 13 20 16 0 0 0 0 0 0 0 0 0 0 0 0 0
[4,] 14 15 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0
> |
```

1. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたmul

```
sorted <- rev(sort(width(fasta)))        #長さ情報を降順に
obj <- (cumsum(sorted) >= Total_len*0.5)#条件を満たすか
N50 <- sorted[obj][1]                    #objがTRUEとなる
```

```
#本番(GC含量情報取得)
hoge <- alphabetFrequency(fasta)         #A,C,G,T,..の数を
CG <- rowSums(hoge[,2:3])                #C,Gの総数を計算してCGに格納
ACGT <- rowSums(hoge[,1:4])              #A,C,G,Tの総数を計算してACGTに格納
GC_content <- sum(CG)/sum(ACGT)          #トータルのGC含量の情報を取得
```

```
#ファイルに保存
tmp <- NULL
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
write.table(t
```

ID	A	C	G	T	配列長	CG	ACGT	%GC含量
contig_1	4	9	7	4	24	16	24	66.6667
contig_2	20	34	31	18	103	65	103	63.1068
contig_3	16	13	20	16	65	33	65	50.7692
contig_4	14	15	10	10	49	25	49	51.0204
Total	54	71	68	48	241	139	241	57.6763

dim関数は行列hogeの行数と列数を表示。hogeオブジェクトは4行×18列からなると解釈する。Macintoshのヒトで4行×17列の結果になった経験あり。理由不明。

R Console

```
> hoge <- alphabetFrequency(fasta)
> hoge
     A  C  G  T M R W S Y K V H D B N - + .
[1,] 4  9  7  4 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[2,] 20 34 31 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[3,] 16 13 20 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[4,] 14 15 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0
> dim(hoge)
[1]  4 18
> hoge[3,]
 A  C  G  T  M  R  W  S  Y  K  V  H  D  B  N  -  +  .
16 13 20 16  0  0  0  0  0  0  0  0  0  0  0  0  0  0
>
```

1. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたmul

```
sorted <- rev(sort(width(fasta)))        #長さ情報を降順に
obj <- (cumsum(sorted) >= Total_len*0.5)#条件を満たすか
N50 <- sorted[obj][1]                    #objがTRUEとなる
```

```
#本番(GC含量情報取得)
hoge <- alphabetFrequency(fasta)         #A,C,G,T,..の数を
CG <- rowSums(hoge[,2:3])                #C,Gの総数を計算してCGに格納
ACGT <- rowSums(hoge[,1:4])              #A,C,G,Tの総数を計算してACGTに格納
GC_content <- sum(CG)/sum(ACGT)          #トータルのGC含量の情報を取得
```

```
#ファイルに保存
tmp <- NULL
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
write.table(t
```

ID	A	C	G	T	配列長	CG	ACGT	%GC含量
contig_1	4	9	7	4	24	16	24	66.6667
contig_2	20	34	31	18	103	65	103	63.1068
contig_3	16	13	20	16	65	33	65	50.7692
contig_4	14	15	10	10	49	25	49	51.0204
Total	54	71	68	48	241	139	241	57.6763

行列要素の抽出は[行, 列]。例えば、3番目のコンティグの結果のみの抽出は、行列hogeの3行目の抽出に相当する。

R Console

```
> hoge <- alphabetFrequency(fasta)
> hoge
      A  C  G  T M R W S Y K V H D B N - + .
[1,]  4  9  7  4 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[2,] 20 34 31 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[3,] 16 13 20 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[4,] 14 15 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0
> dim(hoge)
[1]  4 18
> hoge[3,]
 A  C  G  T  M  R  W  S  Y  K  V  H  D  B  N  -  +  .
16 13 20 16  0  0  0  0  0  0  0  0  0  0  0  0  0  0
>
```

1. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたmul

```
sorted <- rev(sort(width(fasta)))        #長さ情報を降順に
obj <- (cumsum(sorted) >= Total_len*0.5)#条件を満たすか
N50 <- sorted[obj][1]                    #objがTRUEとなる
```

```
#本番(GC含量情報取得)
hoge <- alphabetFrequency(fasta)         #A,C,G,T,..の数を
CG <- rowSums(hoge[,2:3])                #C,Gの総数を計算してCGに格納
ACGT <- rowSums(hoge[,1:4])              #A,C,G,Tの総数を計算してACGTに格納
GC_content <- sum(CG)/sum(ACGT)          #トータルのGC含量の情報を取得
```

```
#ファイルに保存
tmp <- NULL
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
write.table(t
```

ID	A	C	G	T	配列長	CG	ACGT	%GC含量
contig_1	4	9	7	4	24	16	24	66.6667
contig_2	20	34	31	18	103	65	103	63.1068
contig_3	16	13	20	16	65	33	65	50.7692
contig_4	14	15	10	10	49	25	49	51.0204
Total	54	71	68	48	241	139	241	57.6763

行列hogeを眺めることで1列目がAの出現数、2列目がC、3列目がGというのを確認しておくことがその後の解析に重要。

R Console

```
> hoge
     A  C  G  T M R W S Y K V H D B N - + .
[1,]  4  9  7  4 0 0 0 0 0 0 0 0 0 0 0 0 0
[2,] 20 34 31 18 0 0 0 0 0 0 0 0 0 0 0 0 0
[3,] 16 13 20 16 0 0 0 0 0 0 0 0 0 0 0 0 0
[4,] 14 15 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0
> hoge[,1:3]
     A  C  G
[1,]  4  9  7
[2,] 20 34 31
[3,] 16 13 20
[4,] 14 15 10
> hoge[2, 1:3]
 A  C  G
20 34 31
>
```

1. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたmul

```
sorted <- rev(sort(width(fasta)))        #長さ情報を降順に
obj <- (cumsum(sorted) >= Total_len*0.5)#条件を満たすか
N50 <- sorted[obj][1]                    #objがTRUEとなる
```

```
#本番(GC含量情報取得)
hoge <- alphabetFrequency(fasta)         #A,C,G,T,..の数を
CG <- rowSums(hoge[,2:3])                #C,Gの総数を計算してCGに格納
ACGT <- rowSums(hoge[,1:4])              #A,C,G,Tの総数を計算してACGTに格納
GC_content <- sum(CG)/sum(ACGT)          #トータルのGC含量の情報を取得
```

```
#ファイルに保存
tmp <- NULL
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
write.table(t
```

ID	A	C	G	T	配列長	CG	ACGT	%GC含量
contig_1	4	9	7	4	24	16	24	66.6667
contig_2	20	34	31	18	103	65	103	63.1068
contig_3	16	13	20	16	65	33	65	50.7692
contig_4	14	15	10	10	49	25	49	51.0204
Total	54	71	68	48	241	139	241	57.6763

2列目と4列目の情報のみ抽出することは
CとTのみの出現数を得ることに相当する。

R Console

```
> hoge
     A  C  G  T M R W S Y K V H D B N - + .
[1,]  4  9  7  4 0 0 0 0 0 0 0 0 0 0 0 0 0
[2,] 20 34 31 18 0 0 0 0 0 0 0 0 0 0 0 0 0
[3,] 16 13 20 16 0 0 0 0 0 0 0 0 0 0 0 0 0
[4,] 14 15 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0
> hoge[, c(2, 4)]
     C  T
[1,]  9  4
[2,] 34 18
[3,] 13 16
[4,] 15 10
> c(2, 4)
[1] 2 4
>
```

1. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたmul

```
sorted <- rev(sort(width(fasta)))        #長さ情報を降順に
obj <- (cumsum(sorted) >= Total_len*0.5)#条件を満たすか
N50 <- sorted[obj][1]                    #objがTRUEとなる
```

```
#本番(GC含量情報取得)
hoge <- alphabetFrequency(fasta)         #A,C,G,T,..の数を
CG <- rowSums(hoge[,2:3])                #C,Gの総数を計算してCGに格納
ACGT <- rowSums(hoge[,1:4])              #A,C,G,Tの総数を計算してACGTに格納
GC_content <- sum(CG)/sum(ACGT)          #トータルのGC含量の情報を取得
```

```
#ファイルに保存
tmp <- NULL
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
write.table(t
```

ID	A	C	G	T	配列長	CG	ACGT	%GC含量
contig_1	4	9	7	4	24	16	24	66.6667
contig_2	20	34	31	18	103	65	103	63.1068
contig_3	16	13	20	16	65	33	65	50.7692
contig_4	14	15	10	10	49	25	49	51.0204
Total	54	71	68	48	241	139	241	57.6763

rowSums関数は、行(row)ごとの総和(sum)を計算する。colSums関数は、列(column)ごとの総和(sum)を計算する。

R Console

```
> hoge
     A  C  G  T M R W S Y K V H D B N - + .
[1,] 4  9  7  4 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[2,] 20 34 31 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[3,] 16 13 20 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[4,] 14 15 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0
> apply(hoge, 1, sum)
[1]  24 103  65  49
> rowSums(hoge)
[1]  24 103  65  49
> apply(hoge, 2, sum)
 A  C  G  T  M  R  W  S  Y  K  V  H  D  B  N  -  +  .
54 71 68 48  0  0  0  0  0  0  0  0  0  0  0  0  0  0
> colSums(hoge)
 A  C  G  T  M  R  W  S  Y  K  V  H  D  B  N  -  +  .
54 71 68 48  0  0  0  0  0  0  0  0  0  0  0  0  0  0
>
```

1. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたmul

```
sorted <- rev(sort(width(fasta)))        #長さ情報を降順に
obj <- (cumsum(sorted) >= Total_len*0.5)#条件を満たすか
N50 <- sorted[obj][1]                    #objがTRUEとなる
```

```
#本番(GC含量情報取得)
hoge <- alphabetFrequency(fasta)         #A,C,G,T,..の数を
CG <- rowSums(hoge[,2:3])                #C,Gの総数を計算してCGに格納
ACGT <- rowSums(hoge[,1:4])              #A,C,G,Tの総数を計算してACGTに格納
GC_content <- sum(CG)/sum(ACGT)          #トータルのGC含量の情報を取得
```

```
#ファイルに保存
tmp <- NULL
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
write.table(t
```

ID	A	C	G	T	配列長	CG	ACGT	%GC含量
contig_1	4	9	7	4	24	16	24	66.6667
contig_2	20	34	31	18	103	65	103	63.1068
contig_3	16	13	20	16	65	33	65	50.7692
contig_4	14	15	10	10	49	25	49	51.0204
Total	54	71	68	48	241	139	241	57.6763

A, C, G, Tのみのサブセットhoge[,1:4]に対して同様な計算を行っている。

R Console

```
> hoge
      A  C  G  T M R W S Y K V H D B N - + .
[1,]  4  9  7  4 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[2,] 20 34 31 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[3,] 16 13 20 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[4,] 14 15 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0
> apply(hoge[,1:4], 1, sum)
[1]  24 103  65  49
> rowSums(hoge[,1:4])
[1]  24 103  65  49
> apply(hoge[,1:4], 2, sum)
 A  C  G  T
54 71 68 48
> colSums(hoge[,1:4])
 A  C  G  T
54 71 68 48
>
```

1. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたmul

```
sorted <- rev(sort(width(fasta)))        #長さ情報を降順に
obj <- (cumsum(sorted) >= Total_len*0.5)#条件を満たすか
N50 <- sorted[obj][1]                    #objがTRUEとなる
```

```
#本番(GC含量情報取得)
hoge <- alphabetFrequency(fasta)         #A,C,G,T,..の数を
CG <- rowSums(hoge[,2:3])                #C,Gの総数を計算してCGに格納
ACGT <- rowSums(hoge[,1:4])              #A,C,G,Tの総数を計算してACGTに格納
GC_content <- sum(CG)/sum(ACGT)          #トータルのGC含量の情報を取得
```

```
#ファイルに保存
tmp <- NULL
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
write.table(t
```

ID	A	C	G	T	配列長	CG	ACGT	%GC含量
contig_1	4	9	7	4	24	16	24	66.6667
contig_2	20	34	31	18	103	65	103	63.1068
contig_3	16	13	20	16	65	33	65	50.7692
contig_4	14	15	10	10	49	25	49	51.0204
Total	54	71	68	48	241	139	241	57.6763

apply関数は、MARGINオプションの数値を変更することで行(=1)ごとや列(=2)ごとの総和(sum)を計算する。オプション名は省略可能です。

R Console

```
> hoge
      A  C  G  T M R W S Y K V H D B
[1,]  4  9  7  4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[2,] 20 34 31 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[3,] 16 13 20 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[4,] 14 15 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
> apply(hoge[,1:4], 1, sum)
[1]  24 103  65  49
> rowSums(hoge[,1:4])
[1]  24 103  65  49
> apply(hoge[,1:4], 2, sum)
 A  C  G  T
54 71 68 48
> colSums(hoge[,1:4])
 A  C  G  T
54 71 68 48
> |
```

1. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたmul

```
sorted <- rev(sort(width(fasta)))        #長さ情報を降順に
obj <- (cumsum(sorted) >= Total_len*0.5)#条件を満たすか
N50 <- sorted[obj][1]                    #objがTRUEとなる
```

```
#本番(GC含量情報取得)
hoge <- alphabetFrequency(fasta)         #A,C,G,T,..の数を
CG <- rowSums(hoge[,2:3])                #C,Gの総数を計算してCGに格納
ACGT <- rowSums(hoge[,1:4])              #A,C,G,Tの総数を計算してACGTに格納
GC_content <- sum(CG)/sum(ACGT)          #トータルのGC含量の情報を取得
```

```
#ファイルに保存
tmp <- NULL
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
write.table(t
```

ID	A	C	G	T	配列長	CG	ACGT	%GC含量
contig_1	4	9	7	4	24	16	24	66.6667
contig_2	20	34	31	18	103	65	103	63.1068
contig_3	16	13	20	16	65	33	65	50.7692
contig_4	14	15	10	10	49	25	49	51.0204
Total	54	71	68	48	241	139	241	57.6763

R Console

```
> hoge
     A  C  G  T M R W S Y K V H D B N - + .
[1,] 4  9  7  4 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[2,] 20 34 31 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[3,] 16 13 20 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[4,] 14 15 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0
> apply(hoge[,1:4], MARGIN=1, sum)
[1]  24 103  65  49
> apply(hoge[,1:4], MARGIN=1, FUN=sum)
[1]  24 103  65  49
> apply(hoge[,1:4], MARGIN=2, sum)
 A  C  G  T
54 71 68 48
> apply(hoge[,1:4], MARGIN=2, FUN=sum)
 A  C  G  T
54 71 68 48
>
```

オプション名を明示するとこんな感じ。関数名FUNはsum以外にmean, medianなど任意に指定できます。

1. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたmul

```
sorted <- rev(sort(width(fasta)))        #長さ情報を降順に
obj <- (cumsum(sorted) >= Total_len*0.5)#条件を満たすか
N50 <- sorted[obj][1]                    #objがTRUEとなる

#本番(GC含量情報取得)
hoge <- alphabetFrequency(fasta)         #A,C,G,T,..の数を
CG <- rowSums(hoge[,2:3])                #C,Gの総数を計算してCGに格納
ACGT <- rowSums(hoge[,1:4])              #A,C,G,Tの総数を計算してACGTに格納
GC_content <- sum(CG)/sum(ACGT)          #トータルのGC含量の情報を取得

#ファイルに保存
tmp <- NULL
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
tmp <- rbind(
write.table(t
```

ID	A	C	G	T	配列長	CG	ACGT	%GC含量
contig_1	4	9	7	4	24	16	24	66.6667
contig_2	20	34	31	18	103	65	103	63.1068
contig_3	16	13	20	16	65	33	65	50.7692
contig_4	14	15	10	10	49	25	49	51.0204
Total	54	71	68	48	241	139	241	57.6763

全コンティグを合わせたGC含量は139/241=0.5767と計算される。コンティグごとのGC含量も簡単に得ることができる。

R Console

```
> hoge
      A  C  G  T M R W S Y K V H D B N - + .
[1,]  4  9  7  4 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[2,] 20 34 31 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[3,] 16 13 20 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[4,] 14 15 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0
> rowSums(hoge[,2:3])
[1] 16 65 33 25
> sum(CG)/sum(ACGT)
[1] 0.5767635
> CG/ACGT
[1] 0.6666667 0.6310680 0.5076923 0.5102041
>
```

multi-FASTAファイルからの各種情報抽出

- 解析 | 一般 | アラインメント(ペアワイズ; 応用編) (last modified 2010/6/8)
- 解析 | 一般 | パターンマッチング (last modified 2013/06/19)
- 解析 | 一般 | GC含量(GC contents) (last modified 2014/05/01)
- 解析 | 一般 | Sequence logos(Schneider, 1990) (last modified 2014/07/23) NEW
- 解析 | 一般 | 上流配列解析 | LDSS(Yamamoto, 2007) (last modified 2012/07/17)

コンティグごとのGC含量も簡単に得ることができる。(Rで)**塩基配列解析**は、ごく一部の変更で作成した別項目も多数あり。

解析 | 一般 | GC含量 (GC contents)

multi-FASTA形式ファイルを読み込んで配列ごとのGC含量 (GC contents)を出力するやり方を示します。出力ファイルは、「description」「CGの総数」「ACGTの総数」「配列長」「%GC含量」としています。尚、%GC含量は「CGの総数/ACGTの総数」で計算しています。

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

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

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

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

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

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

#ファイルに保存
tmp <- cbind(names(fasta), CG, ACGT, width(fasta), GC_content)#保存したい情報をtmpに格納
colnames(tmp) <- c("description", "CG", "ACGT", "Length", "%GC_contents")#列名を付与
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.names=T)#tmpの中身を指定したファ
```

Contents

- 3-4. R Bioconductor II、2014/09/09 15:00-18:15、中級、実習
  - multi-FASTAファイルからの情報抽出（コンティグ数、総塩基数、N50、GC含量）
    - GC含量計算の詳細説明。alphabetFrequency, apply関数、数値行列計算の基本
    - コンティグごとのGC含量計算
  - **FASTQ形式ファイルの読み込み**
  - ファイル形式の変換：FASTQ → FASTA
  - クオリティチェック(クオリティコントロール；QC)
  - フィルタリング
    - クオリティスコア、N、配列長など
    - 動作確認用のサブセット作成
  - その他（FASTA/FASTQファイルのdescription行を整形）

FASTQ形式ファイルの読み込み

- イントロ | NGS | 読み込み | FASTA形式 | 基本情報を取得 (last modified 2014/05/29)
- イントロ | NGS | 読み込み | FASTA形式 | description行の記述を整形 (last modified 2014/04/
- イントロ | NGS | 読み込み | FASTQ形式 | modified 2014/07/17)
- イントロ | NGS | 読み込み | FASTQ形式 | description行の記述を整形 (last modified 2013/06/
- イントロ | NGS | 読み込み | Illuminaの* seq.txt (last modified 2013/06/13)
- イントロ | NGS | 読み込み | Illuminaの* qseq.txt (last modified 2013/06/17)
- イントロ | ファイル形式の変換 | について (last modified 2014/06/09)
- イントロ | ファイル形式の変換 | BAM --> BED (last modified 2014/06/21)
- イントロ | ファイル
- イントロ | ファイル
- イントロ | ファイル
- イントロ | ファイル
- イントロ | ファイル
- 前処理 | クオリティ
- 前処理 | クオリティ
- 前処理 | クオリティ
- 前処理 | クオリティ
- 前処理 | フィルタリ

イントロ | NGS | 読み込み | FASTQ形式

Sanger FASTQ形式ファイルを読み込むやり方を示します。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

1. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., BMC Bioinformatics, 2010)。
quality情報を除く塩基配列情報のみ読み込むやり方です。配列長が同じ場合のみ読み込めます。

```
in_f <- "SRR037439.fastq"                    #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="fastq")#in_fで指定したファイルの読み込み
fasta                                        #確認してるだけです
```

(Rで)塩基配列解析は沢山の項目があるが、ごく一部の変更もあり。FASTQ形式ファイルもreadDNAStringSet関数のformatオプションを変更するだけで読み込むことができます。

FASTQ形式ファイルの読み込み

イントロ | NGS | 読み込み | FASTQ形式

Sanger FASTQ形式ファイルを読み込むやり方を示します。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

1. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., BMC Bioinformatics, 2010)。
quality情報を除く塩基配列情報のみ読み込むやり方です。配列長が同じ場合のみ読み込めます。

```
in_f <- "SRR037439.fastq"                    #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)

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

```
@SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35↓
NNNNNNNNNNNNNNCTACCCCCCCAGCCGCCGCA↓
+SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35↓
!!!!!!!!!!!!!!!"""""""""""""""""""#"""↓
@SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35↓
NNNNNNNNNNNNNNAGACAGTTGATTTAGCATAG↓
+SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35↓
!!!!!!!!!!!!!!!"+"""""""""""""""""&↓
@SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35↓
NNNNNNNNNNNNNNGGGTGGGGCGTTTGTTCTTG↓
+SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35↓
!!!!!!!!!!!!!!!/5$$&"""%""""""""""↓
@SRR037439.4 HWI-E4_6_30ACL:2:1:0:1349 length=35↓
```

FASTQ形式は、4行で1つのリード情報を表すものです。

1. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., BMC Bioinformatics, 2010)。
quality情報を除く塩基配列情報のみ読み込むやり方です。配列長が同じ場合のみ読み込めます。

```
in_f <- "SRR037439.fastq"                    #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="fastq")#in_fで指定したファ
fasta                                        #確認してるだけです
```

FASTA形式ファイル分の情報(リード塩基配列とdescription情報)のみ読み込むやり方。これは全部で500リード、35塩基長からなるファイルです

R Console

```
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fastq")#in_fで指定$
> fasta                                        #確認してるだけです
  A DNAStringSet instance of length 500
      width seq                                names
  [1]    35 NNNNNNNNNN...AGCCGCCGCA SRR037439.1 HWI-E...
  [2]    35 NNNNNNNNNN...TTTAGCATAG SRR037439.2 HWI-E...
  [3]    35 NNNNNNNNNN...TTTGTTCTTG SRR037439.3 HWI-E...
  [4]    35 NNNNNNNNNN...CCCCTCCCTC SRR037439.4 HWI-E...
  [5]    35 NNNNNNNNNN...GACGCCCACA SRR037439.5 HWI-E...
  ...   ... ...
[496]    35 CTGGACGGCC...CACCCCCCCC SRR037439.496 HWI...
[497]    35 TGTCACTTGT...CACGGGGCTT SRR037439.497 HWI...
[498]    35 CCGCCCTTTT...CACAAAAAAA SRR037439.498 HWI...
[499]    35 CGTTCTTGTT...GGGAAAAACC SRR037439.499 HWI...
[500]    35 GGAGCCTCCC...GGGGGGGGGC SRR037439.500 HWI...
>
```

2. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., BMC Bioinformatics, 2010)。
quality情報も読み込むやり方です。配列長が異なっていても読み込めます。

```
in_f <- "SRR037439.fastq"                    #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(ShortRead)                           #パッケージの読み込み
#入力ファイルの読み込み
fastq <- readFastq(in_f)                     #in_fで指定したファイルの
fastq                                        #確認してるだけです

showClass("ShortReadQ")
sread(fastq)
quality(fastq)
id(fastq)
```

クオリティ情報も読み込むやり方。fastqオブジェクトを表示しても一見わけがわからないが、ShortReadQ形式というクラスオブジェクトなんだろうと解釈する。

R Console

```
> in_f <- "SRR037439.fastq"                  #入力ファイル名を指$
>
> #必要なパッケージをロード
> library(ShortRead)                         #パッケージの読み込$
 要求されたパッケージ BiocParallel をロード中です
 要求されたパッケージ Rsamtools をロード中です
 要求されたパッケージ GenomicRanges をロード中です
 要求されたパッケージ GenomeInfoDb をロード中です
 要求されたパッケージ GenomicAlignments をロード中です
 要求されたパッケージ BSgenome をロード中です
>
> #入力ファイルの読み込み
> fastq <- readFastq(in_f)                   #in_fで指定したファ$
> fastq                                      #確認してるだけです
class: ShortReadQ
length: 500 reads; width: 35 cycles
>
```

2. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., BMC Bioinformatics, 2010)。
quality情報も読み込むやり方です。配列長が異なっていても読み込めます。

```
in_f <- "SRR037439.fastq"                    #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(ShortRead)                           #パッケージの読み込み

#入力ファイルの読み込み
fastq <- readFastq(in_f)                     #in_fで指定したファイルの読み込み
fastq                                        #確認してるだけです

showClass("ShortReadQ")
sread(fastq)
quality(fastq)
id(fastq)
```

実用上は、classとオブジェクト名が分かれば、showClass関数を実行することで情報の取り出し方のヒントをつかめます

R Console

```
> showClass("ShortReadQ")                    #ShortReadQ"という$
Class "ShortReadQ" [package "ShortRead"]

Slots:

Name:       quality        sread            id
Class: QualityScore DNAStringSet   BStringSet

Extends:
Class "ShortRead", directly
Class ".ShortReadBase", by class "ShortRead", distance 2

Known Subclasses: "AlignedRead"
>
```

2. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., BMC Bioinformatics, 2010)。
quality情報も読み込むやり方です。配列長が異なっていても読み込めます。

```
in_f <- "SRR037439.fastq"                    #入力ファイル名を指定し
#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
fastq <- readFastq(in_f)                     #in_fで指定したファイル
fastq                                        #確認してるだけです

showClass("ShortReadQ")
sread(fastq)
quality(fastq)
id(fastq)
```

例えばsread(fastq)はこのウェブページ上でfastaというオブジェクト名で取り扱うものと基本的に同じ。ただし、description情報は含まれてないことがわかる。

R Console

```
> sread(fastq)
  A DNAStringSet instance of length 500
      width seq
  [1]    35 NNNNNNNNNNNNNNNCTACCCCCCCAGCCGCCGCA
  [2]    35 NNNNNNNNNNNNNNNAGACAGTTGATTTAGCATAG
  [3]    35 NNNNNNNNNNNNNNNGGGTGGGGCGTTTGTTCTTG
  [4]    35 NNNNNNNNNNNNNNNCCCCGCCCCGCCCCTCCCTC
  [5]    35 NNNNNNNNNNNNNNNGGTCGCCCCCGACGCCCACA
  ...   ... ...
[496]    35 CTGGACGGCCCCCCCCCCACACACCCACCCCCCCC
[497]    35 TGTCACTTGTGCTTTGCTCTTGTCCCACGGGGCTT
[498]    35 CCGCCCTTTTTTCCAGAAATTTTCCGCACAAAAAAA
[499]    35 CGTTCTTGTTGCCCCCGGGGCGGGGGGGAAAAACC
[500]    35 GGAGCCTCCCCCCCCCCCCCAAGGGGGGGGGGGGGC
> |
```

Contents

- 3-4. R Bioconductor II、2014/09/09 15:00-18:15、中級、実習
  - multi-FASTAファイルからの情報抽出（コンティグ数、総塩基数、N50、GC含量）
    - GC含量計算の詳細説明。alphabetFrequency, apply関数、数値行列計算の基本
    - コンティグごとのGC含量計算
  - FASTQ形式ファイルの読み込み
  - **ファイル形式の変換：FASTQ → FASTA**
  - クオリティチェック(クオリティコントロール；QC)
  - フィルタリング
    - クオリティスコア、N、配列長など
    - 動作確認用のサブセット作成
  - その他（FASTA/FASTQファイルのdescription行を整形）

ファイル形式の変換(FASTQ → FASTA)

- イントロ | ファイル形式の変換 | について (last modified 2014/06/09)
- イントロ | ファイル形式の変換 | BAM --> BED (last modified 2014/06/21)
- イントロ | ファイル形式の変換 | FASTQ --> FASTA (last modified 2013/06/17)
- イントロ | ファイル形式の変換 | Genbank --> FASTA (last modified 2014/03/10)
- イントロ | ファイル形式の変換 | qseq --> FASTA (last modified 2013/06/17)

FASTQからFASTAへのファイル形式の変換も途中までは全く同じです。

イントロ | ファイル形式の変換 | FASTQ --> FASTA NEW

Sanger FASTQ形式ファイルを読み込んでFASTA形式で出力するやり方を示します。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

4. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., 2010)。配列長が同じであってもreadFastq関数で読み込むことができます。

```
in_f <- "SRR037439.fastq"                #入力ファイル名を指定してin_fに格納
out_f <- "hoge4.fasta"                   #出力ファイル名を指定してout_fに格納

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

#入力ファイルの読み込み
fastq <- readFastq(in_f)                 #in_fで指定したファイルの読み込み
sread(fastq)                             #配列情報を表示

#本番
fasta <- sread(fastq)                    #fastqの配列情報部分をfastaに格納
names(fasta) <- id(fastq)                #description情報部分をfastaに追加
fasta                                    #確認してるだけです

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

ファイル形式の変換(FASTQ → FASTA)

4. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデー
(Bullard et al., 2010)。配列長が同じであってもreadFastq関数で読み込むことができます。

```
in_f <- "SRR037439.fastq"                #入力ファイル名を指定してin_fl
out_f <- "hoge4.fasta"                   #出力ファイル名を指定してout_f

#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
fastq <- readFastq(in_f)
sread(fastq)

#本番
fasta <- sread(fastq)
names(fasta) <- id(fastq)
fasta

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

sread(fastq)はこのウェブページ上でfastaというオブジェクト名で取り扱うものと基本的に同じ。ただし、description情報は含まれてないことがわかる。

R Console

```
> sread(fastq)
  A DNAStringSet instance of length 500
      width seq
  [1]    35 NNNNNNNNNNNNNNNCTACCCCCCCAGCCGCCGCA
  [2]    35 NNNNNNNNNNNNNNNAGACAGTTGATTTAGCATAG
  [3]    35 NNNNNNNNNNNNNNNGGGTGGGGCGTTTGTTCTTG
  [4]    35 NNNNNNNNNNNNNNNCCCCGCCCCGCCCCTCCCTC
  [5]    35 NNNNNNNNNNNNNNNGGTCGCCCCCGACGCCCACA
  ...   ... ...
[496]    35 CTGGACGGCCCCCCCCCCACACACCCACCCCCCCC
[497]    35 TGTCACTTGTGCTTTGCTCTTGTCCCACGGGGCTT
[498]    35 CCGCCCTTTTTTCCAGAAATTTTCCGCACAAAAAAA
[499]    35 CGTTCTTGTTGCCCCCGGGGCGGGGGGGAAAAACC
[500]    35 GGAGCCTCCCCCCCCCCCCCAAGGGGGGGGGGGGGC
>
```

ファイル形式の変換(FASTQ → FASTA)

4. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータ
(Bullard et al., 2010)。配列長が同じであってもreadFastq関数で読み込むことができます。

```
in_f <- "SRR037439.fastq"                    #入力ファイル名を指定してin_fに
out_f <- "hoge4.fasta"                       #出力ファイル名を指定してout_fに格納

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

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

#本番
fasta <- sread(fastq)
names(fasta) <- id(fastq)
fasta

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

当たり前だが、fastaというオブジェクト名で取り扱えば自動的にdescription情報が追加されるわけではない。

R Console

```
> fasta <- sread(fastq)
> fasta
  A DNAStringSet instance of length 500
      width seq
  [1]    35 NNNNNNNNNNNNNNNCTACCCCCCCAGCCGCCGCA
  [2]    35 NNNNNNNNNNNNNNNAGACAGTTGATTTAGCATAG
  [3]    35 NNNNNNNNNNNNNNNGGGTGGGGCGTTTGTTCTTG
  [4]    35 NNNNNNNNNNNNNNNCCCCGCCCCGCCCCTCCCTC
  [5]    35 NNNNNNNNNNNNNNNGGTCGCCCCCGACGCCCACA
  ...   ... ...
[496]    35 CTGGACGGCCCCCCCCCCACACACCCACCCCCCCC
[497]    35 TGTCACTTGTGCTTTGCTCTTGTCCCACGGGGCTT
[498]    35 CCGCCCTTTTTCCAGAAATTTTCCGCACAAAAAAA
[499]    35 CGTTCTTGTTGCCCCCGGGGCGGGGGGGAAAAACC
[500]    35 GGAGCCTCCCCCCCCCCCCAAGGGGGGGGGGGGGC
>
```

ファイル形式の変換(FASTQ → FASTA)

4. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデー
(Bullard et al., 2010)。配列長が同じであってもreadFastq関数で読み込むことができます。

```
in_f <- "SRR037439.fastq"                    #入力ファイル名を指定してin_fl
out_f <- "hoge4.fasta"                       #出力ファイル名を指定してout_f

#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
fastq <- readFastq(in_f)
sread(fastq)

#本番
fasta <- sread(fastq)
names(fasta) <- id(fastq)
fasta

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

ShortReadQ形式のfastqオブジェクトをshowClass関数で眺めると、BStringSet形式のdescription情報をid(fastq)で取り出せることが経験的に分かってくる。

R Console

```
> fastq
class: ShortReadQ
length: 500 reads; width: 35 cycles
> showClass("ShortReadQ")
Class "ShortReadQ" [package "ShortRead"]

Slots:

Name:       quality        sread            id
Class: QualityScore DNAStringSet   BStringSet

Extends:
Class "ShortRead", directly
Class ".ShortReadBase", by class "ShortRead", distance 2

Known Subclasses: "AlignedRead"
>
```

ファイル形式の変換(FASTQ → FASTA)

4. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデー
(Bullard et al., 2010)。配列長が同じであってもreadFastq関数で読み込むことができます。

```
in_f <- "SRR037439.fastq"                #入力ファイル名を指定してin_fに
out_f <- "hoge4.fasta"                   #出力ファイル名を指定してout_f

#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
fastq <- readFastq(in_f)
sread(fastq)

#本番
fasta <- sread(fastq)
names(fasta) <- id(fastq)
fasta

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

ShortReadQ形式のfastqオブジェクトをshowClass関数で眺めると、BStringSet形式のdescription情報をid(fastq)で取り出せることが経験的に分かってくる。

R Console

```
> id(fastq)
  A BStringSet instance of length 500
      width seq
  [1]    46 SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35
  [2]    46 SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35
  [3]    47 SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35
  [4]    47 SRR037439.4 HWI-E4_6_30ACL:2:1:0:1349 length=35
  [5]    47 SRR037439.5 HWI-E4_6_30ACL:2:1:0:1669 length=35
  ...   ... ...
[496]    50 SRR037439.496 HWI-E4_6_...L:2:1:13:1279 length=35
[497]    49 SRR037439.497 HWI-E4_6_30ACL:2:1:13:508 length=35
[498]    49 SRR037439.498 HWI-E4_6_30ACL:2:1:13:760 length=35
[499]    50 SRR037439.499 HWI-E4_6_...L:2:1:13:1596 length=35
[500]    50 SRR037439.500 HWI-E4_6_...L:2:1:13:1034 length=35
>
```

頭の整理

4. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出
(Bullard et al., 2010)。配列長が同じであってもreadFastq関数で読み

```
in_f <- "SRR037439.fastq"                    #入力ファイル
out_f <- "hoge4.fasta"                       #出力ファイル

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

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

#本番
fasta <- sread(fastq)
names(fasta) <- id(fastq)
fasta

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

```
@SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35
NNNNNNNNNNNNNNCTACCCCCCCAGCCGCCGCA
+SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35
!!!!!!!!!!!!!!!"""""""""""""""""#"""
@SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35
NNNNNNNNNNNNNNAGACAGTTGATTTAGCATAG
+SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35
!!!!!!!!!!!!!!!"+"""""""""""""""""&
@SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35
NNNNNNNNNNNNNNGGGTGGGGCGTTTGTTCTTG
+SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35
!!!!!!!!!!!!!!!/5$$&"""%""""""""""
```

R Console

```
> id(fastq)
  A BStringSet instance of length 500
      width seq
  [1]    46 SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35
  [2]    46 SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35
  [3]    47 SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35
  [4]    47 SRR037439.4 HWI-E4_6_30ACL:2:1:0:1349 length=35
  [5]    47 SRR037439.5 HWI-E4_6_30ACL:2:1:0:1669 length=35
  ...   ... ...
[496]    50 SRR037439.496 HWI-E4_6_...L:2:1:13:1279 length=35
[497]    49 SRR037439.497 HWI-E4_6_30ACL:2:1:13:508 length=35
[498]    49 SRR037439.498 HWI-E4_6_30ACL:2:1:13:760 length=35
[499]    50 SRR037439.499 HWI-E4_6_...L:2:1:13:1596 length=35
[500]    50 SRR037439.500 HWI-E4_6_...L:2:1:13:1034 length=35
> |
```

width列の数値はdescription部分の文字数

ファイル形式の変換(FASTQ → FASTA)

4. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデー
(Bullard et al., 2010)。配列長が同じであってもreadFastq関数で読み込むことができます。

```
in_f <- "SRR037439.fastq"                    #入力ファイル名を指定してin_f
out_f <- "hoge4.fasta"                       #出力ファイル名を指定してout_f

#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
fastq <- readFastq(in_f)
sread(fastq)

#本番
fasta <- sread(fastq)
names(fasta) <- id(fastq)
fasta

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

description情報を含まないfastaオブジェクトのnames列に、description情報に相当するid(fastq)を代入し、代入後のfastaオブジェクトを表示している。

R Console

```
> names(fasta) <- id(fastq)                  #description情報部$
> fasta                                      #確認してるだけです
  A DNAStringSet instance of length 500
      width seq                                          names
  [1]    35 NNNNNNNNNNNNN...CCAGCCGCCGCA SRR037439.1 HWI-E...
  [2]    35 NNNNNNNNNNNNN...GATTTAGCATAG SRR037439.2 HWI-E...
  [3]    35 NNNNNNNNNNNNN...CGTTTGTTCTTG SRR037439.3 HWI-E...
  [4]    35 NNNNNNNNNNNNN...CGCCCCTCCCTC SRR037439.4 HWI-E...
  [5]    35 NNNNNNNNNNNNN...CCGACGCCCACA SRR037439.5 HWI-E...
  ...   ... ...
[496]    35 CTGGACGGCCCCC...CCCACCCCCCCC SRR037439.496 HWI...
[497]    35 TGTCACTTGTGCT...CCCACGGGGCTT SRR037439.497 HWI...
[498]    35 CCGCCCTTTTTCC...CGCACAAAAAAA SRR037439.498 HWI...
[499]    35 CGTTCTTGTTGCC...GGGGGAAAAACC SRR037439.499 HWI...
[500]    35 GGAGCCTCCCCCC...GGGGGGGGGGGC SRR037439.500 HWI...
>
```

ファイル形式の変換(FASTQ → FASTA)

4. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., 2010)。配列長が同じであってもreadFastq関数で読み込むことができます。

```
in_f <- "SRR037439.fastq"          #入力ファイル名を指定してin_fに格納
out_f <- "hoge4.fasta"             #出力ファイル名を指定してout_fに格納
```

FASTQ形式からFASTA形式への変換がちゃんとできています。

入力:塩基配列ファイル(SRR037439.fastq)

```
@SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35
NNNNNNNNNNNNNNCTACCCCCCCCAGCCGCCGCA
+SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35
!!!!!!!!!!!!!!!"""""""""""""""""#"""
@SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35
NNNNNNNNNNNNNNAGACAGTTGATTTAGCATAG
+SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35
!!!!!!!!!!!!!!!"+"""""""""""""""""&
@SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35
NNNNNNNNNNNNNNGGGTGGGGCGTTTGTTCTTG
+SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35
!!!!!!!!!!!!!!!/5$$&"""""%""""""""""
@SRR037439.4 HWI-E4_6_30ACL:2:1:0:1349 length=35
NNNNNNNNNNNNNNCCCCGCCCCGCCCCTCCCTC
+SRR037439.4 HWI-E4_6_30ACL:2:1:0:1349 length=35
!!!!!!!!!!!!!!!"0"("""&""""$""""""""
@SRR037439.5 HWI-E4_6_30ACL:2:1:0:1669 length=35
NNNNNNNNNNNNNNGGTCGCCCCCGACGCCCACA
+SRR037439.5 HWI-E4_6_30ACL:2:1:0:1669 length=35
```

出力:アミノ酸配列ファイル(hoge4.fasta)

```
>SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35
NNNNNNNNNNNNNNCTACCCCCCCCAGCCGCCGCA
>SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35
NNNNNNNNNNNNNNAGACAGTTGATTTAGCATAG
>SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35
NNNNNNNNNNNNNNGGGTGGGGCGTTTGTTCTTG
>SRR037439.4 HWI-E4_6_30ACL:2:1:0:1349 length=35
NNNNNNNNNNNNNNCCCCGCCCCGCCCCTCCCTC
>SRR037439.5 HWI-E4_6_30ACL:2:1:0:1669 length=35
NNNNNNNNNNNNNNGGTCGCCCCCGACGCCCACA
>SRR037439.6 HWI-E4_6_30ACL:2:1:0:1719 length=35
NNNNNNNNNNNNNNACACGTCCCACCCCCCGCCC
>SRR037439.7 HWI-E4_6_30ACL:2:1:0:1942 length=35
```

ファイル形式の変換(FASTQ → FASTA)

1. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., 2010)。配列長が同じであることが既知の場合です。

```
in_f <- "SRR037439.fastq"                #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.fasta"                   #出力ファイル名を指定してout_fに格納

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

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

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

入力ファイルのリード長が同じ場合にはこちらのほうが簡単

R Console

```
> fasta <- readDNAStringSet(in_f, format="fastq")#in_fで指定$
> fasta                                    #確認してるだけです
  A DNAStringSet instance of length 500
      width seq                                    names
  [1]    35 NNNNNNNNNNNNN...CCAGCCGCCGCA SRR037439.1 HWI-E...
  [2]    35 NNNNNNNNNNNNN...GATTTAGCATAG SRR037439.2 HWI-E...
  [3]    35 NNNNNNNNNNNNN...CGTTTGTTCTTG SRR037439.3 HWI-E...
  [4]    35 NNNNNNNNNNNNN...CGCCCCTCCCTC SRR037439.4 HWI-E...
  [5]    35 NNNNNNNNNNNNN...CCGACGCCCACA SRR037439.5 HWI-E...
  ...   ... ...
[496]    35 CTGGACGGCCCCC...CCCACCCCCCCC SRR037439.496 HWI...
[497]    35 TGTCACTTGTGCT...CCCACGGGGCTT SRR037439.497 HWI...
[498]    35 CCGCCCTTTTTCC...CGCACAAAAAAA SRR037439.498 HWI...
[499]    35 CGTTCTTGTTGCC...GGGGGAAAAACC SRR037439.499 HWI...
[500]    35 GGAGCCTCCCCCC...GGGGGGGGGGGC SRR037439.500 HWI...
>
```

Contents

- 3-4. R Bioconductor II、2014/09/09 15:00-18:15、中級、実習
  - multi-FASTAファイルからの情報抽出（コンティグ数、総塩基数、N50、GC含量）
    - GC含量計算の詳細説明。alphabetFrequency, apply関数、数値行列計算の基本
    - コンティグごとのGC含量計算
  - FASTQ形式ファイルの読み込み
  - ファイル形式の変換：FASTQ → FASTA
  - **クオリティチェック(クオリティコントロール；QC)**
  - フィルタリング
    - クオリティスコア、N、配列長など
    - 動作確認用のサブセット作成
  - その他（FASTA/FASTQファイルのdescription行を整形）

NGSデータ解析とR

塩基配列データ取得
↓
QCやpreprocessing
↓
マッピング
↓
カウント情報取得
↓
発現変動解析

(Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランスクリプトーム、正規化、発現変動、統計、モデル、バイオインフォマティクス～
(last modified 2014/07/14, since 2010)

- イントロ | 一般 | 配列取得 | ゲノム配列 | 公共DBから (last modified 2014/05/28)
- イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome (last modified 2014/06/28) NEW
- イントロ | 一般 | 配列取得 | プロモーター配列 | 公共DBから (last modified 2014/04/02)
- イントロ | 一般 | 配列取得 | プロモーター配列 | BSgenome (last modified 2014/04/25)
- イントロ | 一般 | 配列取得 | プロモーター配列 | GenomicFeatures(Lawrence 2013) (last modifi
- イントロ | 一般 | 配列取得 | トランスクリプトーム配列 | 公共DBから (last modified 2014/04/02)
- イントロ | 一般 | 配列取得 | トランスクリプトーム配列 | biomaRt(Durinck 2009) (last modified
- イントロ | NGS | 様々なプラットフォーム (last modified 2014/06/10)
- イントロ | NGS | qPCRやmicroarrayなどとの比較 (last modified 2014/07/11) NEW
- イントロ | NGS | 可視化(ゲノムブラウザやViewer) (last modified 2014/06/25) NEW
- イントロ | NGS | 配列取得 | FASTQ or SRALite | 公共DBから (last modified 2014/06/28) N
- イントロ | NGS | 配列取得 | FASTQ or SRALite | SRAdb(Zhu 2013) (last modified 2014/06/
- イントロ | NGS | 配列取得 | シミュレーションデータ | について (last modified 2014/06/25) NEW
- イントロ | NGS | 配列取得 | シミュレーションデータ | ランダムな塩基配列の生成から (last mod
- イントロ | NGS | アノテーション情報取得 | について (last modified 2014/03/26)
- イントロ | NGS | アノテーション情報取得 | GFF/GTF形式ファイル (last modified 2014/04/11)
- イントロ | NGS | アノテーション情報取得 | refFlat形式ファイル (last modified 2013/09/25)
- イントロ | NGS | アノテーション情報取得 | biomaRt(Durinck 2009) (last modified 2013/09/26)
- イントロ | NGS | アノテーション情報取得 | TranscriptDb | について (last modified 2014/03/28)
- イントロ | NGS | アノテーション情報取得 | TranscriptDb | TxDb.*から (last modified 2013/10/08
- イントロ | NGS | アノテーション情報取得 | TranscriptDb | GenomicFeatures(Lawrence 2013) (la
- イントロ | NGS | アノテーション情報取得 | TranscriptDb | GFF/GTF形式ファイルから (last mod

ヒトやマウスなどのリファレンス配列、NGSデータ、アノテーション情報取得などもR経由で可能。wgetやftp周辺

NGSデータ解析とR

塩基配列データ取得
↓
QCやpreprocessing
↓
マッピング
↓
カウント情報取得
↓
発現変動解析

(Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランスクリプトーム、正規化、発現変動、統計、モデル、バイオインフォマティクス～
(last modified 2014/07/14, since 2010)

- イントロ | NGS | 読み込み | FASTA形式 | 基本情報を取得(last modified 2014/05/29)
- 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(last modified 2014/06/15)
- イントロ | NGS | 読み込み | FASTQ形式 | description行の記述を整形(last modified 2013/06/13)
- イントロ | NGS | 読み込み | Illuminaの* seq.txt(last modified 2013/06/13)
- イントロ | NGS | 読み込み | Illuminaの* qseq.txt(last modified 2013/06/17)
- イントロ | ファイル形式の変換 | について(last modified 2014/06/09)
- イントロ | ファイル形式の変換 | BAM --> BED(last modified 2014/06/21) NEW
- イントロ | ファイル形式の変換 | FASTQ --> FASTA(last modified 2013/06/17)
- イントロ | ファイル形式の変換 | Genbank --> FASTA(last modified 2014/03/10)
- イントロ | ファイル形式の変換 | qseq --> FASTA(last modified 2013/06/17)
- イントロ | ファイル形式の変換 | qseq --> Illumina FASTQ(last modified 2013/06/17)
- イントロ | ファイル形式の変換 | qseq --> Sanger FASTQ(last modified 2013/08/19)
- 前処理 | クオリティチェック | について (last modified 2014/06/30) NEW
- 前処理 | クオリティチェック | qrqc(last modified 2014/06/11)
- 前処理 | クオリティチェック | PHREDスコアに変換(last modified 2013/06/18)
- 前処理 | クオリティチェック | 配列長分布を調べる(last modified 2013/06/18)

FASTAやFASTQ形式ファイルの読み込み。ファイル形式の変換、Quality Control (QC)なども可能。SAMtoolsやFastQC周辺。

NGSデータ解析とR

(Rで)塩基配列解析

~NGS、RNA-seq、ゲノム、トランスクリプトーム、正規化、発現変動、統計、モデル、バイ
(last modified 2014/07/14, since 2010)

- イントロ | NGS | 読み込み | FASTA形式 | 基本情報を取得(last mod
- イントロ | NGS | 読み込み | FASTA形式 | description行の記述を整
- イントロ | NGS | 読み込み | FASTQ形式(last modified 2014/06/15)
- イントロ | NGS | 読み込み | FASTQ形式 | description行の記述を整
- イントロ | NGS | 読み込み | Illuminaの* seq.txt(last modified 2013/
- イントロ | NGS | 読み込み | Illuminaの* qseq.txt(last modified 2013
- イントロ | ファイル形式の変換 | について(last modified 2014/06/09)
- イントロ | ファイル形式の変換 | BAM --> BED(last modified 2014/0
- イントロ | ファイル形式の変換 | FASTQ --> FASTA(last modified 2(
- イントロ | ファイル形式の変換 | Genbank --> FASTA(last modified 2
- イントロ | ファイル形式の変換 | qseq --> FASTA(last modified 2013
- イントロ | ファイル形式の変換 | qseq --> Illumina FASTQ(last modi
- イントロ | ファイル形式の変換 | qseq --> Sanger FASTQ(last modifie
- 前処理 | クオリティチェック | について(last modified 2014/06/30) N
- 前処理 | クオリティチェック | qrqc(last modified 2014/06/11)
- 前処理 | クオリティチェック | PHREDスコアに変換(last modified 201
- 前処理 | クオリティチェック | 配列長分布を調べる(last modified 201

FastQCなどR以外のプログラムもリストアップしています

前処理 | クオリティチェック | について NEW

Quality Control (QC)を実行する様々な方法をリストアップします。Krakenなどアダプター配列除去などが行えるものも含みます。

R用:

- qrqc: 原著論文なし
- PIQA: Martinez-Alcantara et al., Bioinformatics, 2009
- ShortRead: Morgan et al., Bioinformatics, 2009
- girafe: Toedling et al., Bioinformatics, 2010
- QuasR: 原著論文なし

R以外:

- FastQC: 原著論文なし
- FASTX-ToolKit: 原著論文なし
- SolexaQA: Cox et al., BMC Bioinformatics, 2010
- Quake: Kelley et al., Genome Biol., 2010
- NGSQC: Dai et al., BMC Genomics, 2010
- Cutadapt: Martin, M., EMBnet.journal, 2011
- PRINSEQ: Schmieder and Edwards, Bioinformatics, 2011
- ECHO: Kao et al., Genome Res., 2011
- Btrim: Kong Y., Genomics, 2011
- Hammer: Medvedev et al., Bioinformatics, 2011
- ConDeTri: Smeds et al., PLoS One, 2011
- BIGpre: Zhang et al., Genomics Proteomics Bioinformatics, 2011
- NGS QC Toolkit: Patel et al., PLoS One, 2012
- RobiNA: Lohse et al., Nucleic Acids Res., 2012
- SEQuel: Ronen et al., Bioinformatics, 2012
- AdapterRemoval: Lindgreen S., BMC Res Notes, 2012
- Slim-Filter: Golovko et al., BMC Bioinformatics, 2012
- HTQC: Yang et al., BMC Bioinformatics, 2013
- QC-Chain: Zhou et al., PLoS One, 2013
- Kraken: Davis et al., Methods, 2013
- Skewer: Jiang et al., BMC Bioinformatics, 2014

クオリティチェック(QC)

- イントロ | ファイル形式の変換 | qseq --> Illumina FASTQ (last modified 2013/06/17)
- イントロ | ファイル形式の変換 | qseq --> Sanger FASTQ (last modified 2013/08/19)
- 前処理 | クオリティチェック | について (last modified 2014/06/30)
- 前処理 | クオリティチェック | qrqc (last modified 2014/07/17)
- 前処理 | クオリティチェック | PHRED スコアに変換 (last modified 2013/06/18)
- 前処理 | クオリティチェック | 配列長分布を調べる (last modified 2013/06/18)
- 前処理 | フィルタリ
- 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

FastQCと似たようなQCレポートを得ることができます

前処理 | クオリティチェック | qrqc

FastQCのR版のようなものです。Sanger FASTQ形式ファイルを読み込んで、positionごとの「クオリティスコア(quality score)」、「どんな塩基が使われているのか(base frequency and base proportion)」、「リード長の分布」、「GC含量」、「htmlレポート」などを出力してくれます。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

1. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., BMC Bioinformatics, 2010)。下記を実行すると「SRR037439-report」という名前のフォルダが作成されます。中にあるreport.htmlをダブルクリックするとhtmlレポートを見ることができます。

```
in_f <- "SRR037439.fastq"                    #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(qrqc)                                #パッケージの読み込み

#入力ファイルの読み込み
fastq <- readSeqFile(in_f, quality="sanger")#in_fで指定したファイルの読み込み

#本番
makeReport(fastq)                            #htmlレポートの作成
```

クオリティチェック(QC)

前処理 | クオリティチェック | qrqc

FastQCのR版のようなものです。Sanger FASTQ形式ファイルを読み込んで、positionごとの「クオリティスコア(quality score)」、「どんな塩基が使われているのか(base frequency and base proportion)」、「リード長の分布」、「GC含量」、「htmlレポート」などを出力してくれます。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

1. サンプルデータ7のFASTQ形式ファイル(SRF

SRR037439から得られるFASTQファイルの最初
(Bullard et al., BMC Bioinformatics, 2010)。下
ルダが作成されます。中にあるreport.htmlを

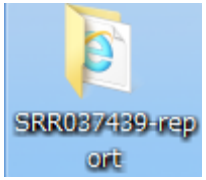
```
in_f <- "SRR037439.fastq"

#必要なパッケージをロード
library(qrqc)

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

#本番
makeReport(fastq)
```

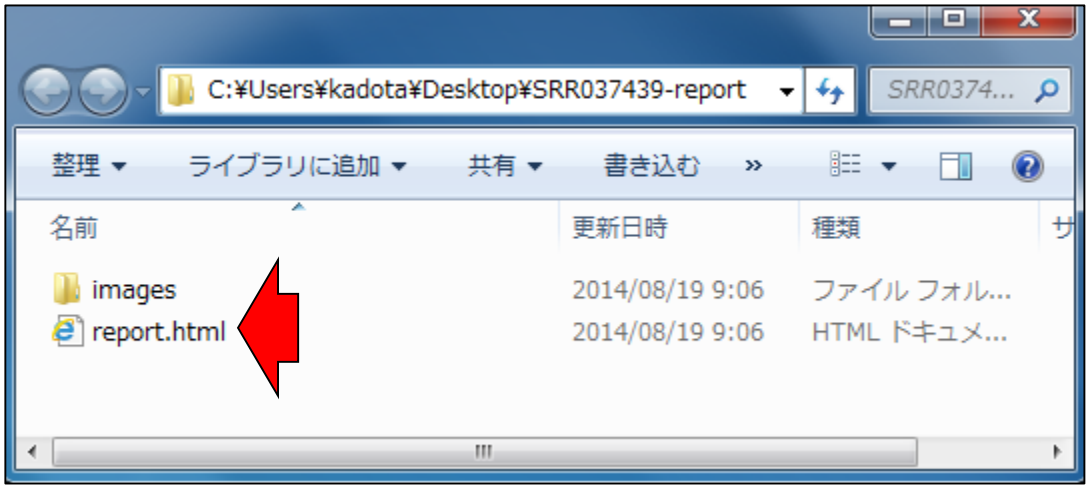
エラーメッセージは出ているが、一応最後までできている

R Console

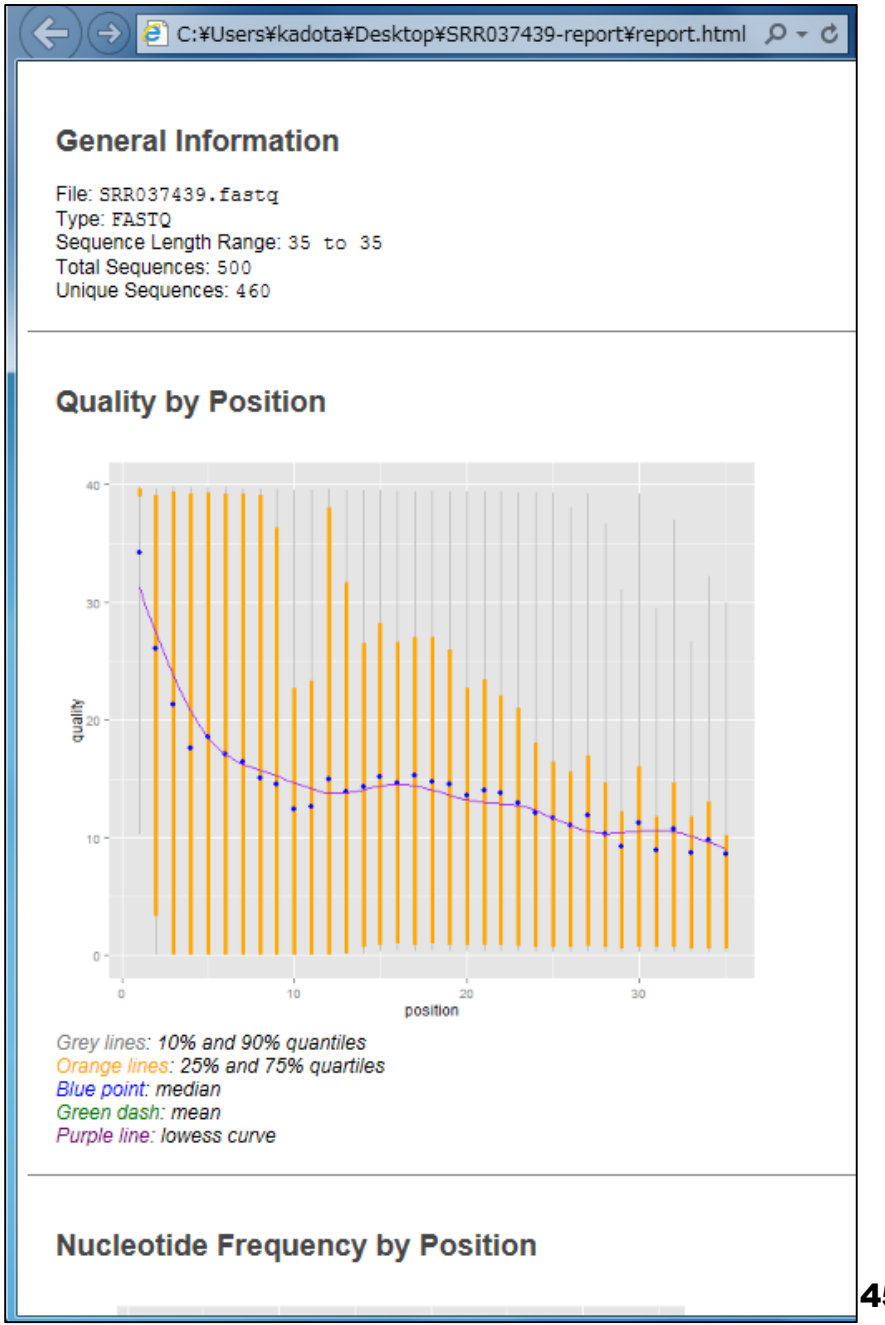
```
    compact

 要求されたパッケージ biovizBase をロード中です
 要求されたパッケージ brew をロード中です
 要求されたパッケージ xtable をロード中です
 要求されたパッケージ Rsamtools をロード中です
 要求されたパッケージ GenomicRanges をロード中です
 要求されたパッケージ GenomeInfoDb をロード中です
 要求されたパッケージ testthat をロード中です
>
> #入力ファイルの読み込み
> fastq <- readSeqFile(in_f, quality="sanger")#in_fで指定し$
>
> #本番
> makeReport(fastq)                          #htmlレポートの作成
geom_smooth: method="auto" and size of largest group is >=1$
Error : Insufficient values in manual scale. 2 needed but o$
Report written to directory './SRR037439-report'.
>
```

クオリティチェック(QC)





htmlファイルを開くと、塩基のポジションごとのクオリティスコア分布など全体像を眺めることができる。漫然と眺めるのではなく、フィルタリングの際に用いる閾値と得られる結果のイメージを掴むべし。



Contents

- 3-4. R Bioconductor II、2014/09/09 15:00-18:15、中級、実習
  - multi-FASTAファイルからの情報抽出（コンティグ数、総塩基数、N50、GC含量）
    - GC含量計算の詳細説明。alphabetFrequency, apply関数、数値行列計算の基本
    - コンティグごとのGC含量計算
  - FASTQ形式ファイルの読み込み
  - ファイル形式の変換：FASTQ → FASTA
  - クオリティチェック(クオリティコントロール；QC)
  - **フィルタリング**
    - **クオリティスコア、N、配列長など**
    - 動作確認用のサブセット作成
  - その他（FASTA/FASTQファイルのdescription行を整形）

フィルタリング

- イントロ | ファイル形式の変換 | qseq --> Sanger FASTQ (last modified 2013/08/19)
- 前処理 | クオリティチェック | について (last modified 2014/06/30)
- 前処理 | クオリティチェック | qrqc (last modified 2014/07/17)
- 前処理 | クオリティチェック | PHREDスコアに変換 (last modified 2013/06/18)
- 前処理 | クオリティチェック | 配列長分布を調べる (last modified 2013/06/18)
- 前処理 | フィルタリング | PHREDスコアが低い塩基をNに置換 (last modified 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 2014/03/03)
- 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) NEW
- 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 | AC
- 前処理 | フィルタリング | 重複
- 前処理 | フィルタリング | 指定
- 前処理 | フィルタリング | 任意
- 前処理 | フィルタリング | 指定
- 前処理 | フィルタリング | 任意
- 前処理 | フィルタリング | Illu
- 前処理 | フィルタリング | GF
- 前処理 | フィルタリング | 組合
- 前処理 | トリミング | ポリA配
- 前処理 | トリミング | アダプタ
- 前処理 | トリミング | アダプタ
- 前処理 | トリミング | アダプタ
- 前処理 | トリミング | アダプタ
- 前処理 | トリミング | 指定した
- アセンブル | について (last m
- アセンブル | ゲノム用 (last m
- アセンブル | トランスクリプト
- マッピング | について (last m

Phredスコア20未満の塩基が配列長の10%以上を占めるリードを除去するやり方です

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 NEW

Sanger FASTQ形式ファイルを読み込んで、PHREDスコアが低いリードを除去するやり方を紹介します。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

1. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., 2010)。PHREDスコアが20未満のものがリード長に占める割合が0.1以上のリードを除去するやり方です。(例題のファイル中のリードは全て35bpのリードである。その10%以上ということで実質的にPHREDスコアが閾値未満のものが4塩基以上あるリードはダメということ) writeFastq関数のデフォルトオプションはcompress=Tで、gzip圧縮ファイルを出力します。ここではcompress=Fとして非圧縮ファイルを出力しています。

```
in_f <- "SRR037439.fastq"                    #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.fastq"                       #出力ファイル名を指定してout_fに格納
param1 <- 20                                 #PHREDスコアの閾値を指定
param2 <- 0.1                                #指定した閾値未満のものが配列長に占める割合を指定

#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
fastq <- readFastq(in_f)                     #in_fで指定したファイルの読み込み
sread(fastq)                                 #配列情報を表示

#本番
hoge <- as(quality(fastq), "matrix")         #ASCIIコードのquality scoreをPHRED scoreに変換し、デー
```

フィルタリング

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

Sanger FASTQ形式ファイルを読み込んで、PHREDスコアが低いリードを除去するやり方を紹介します。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

1. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

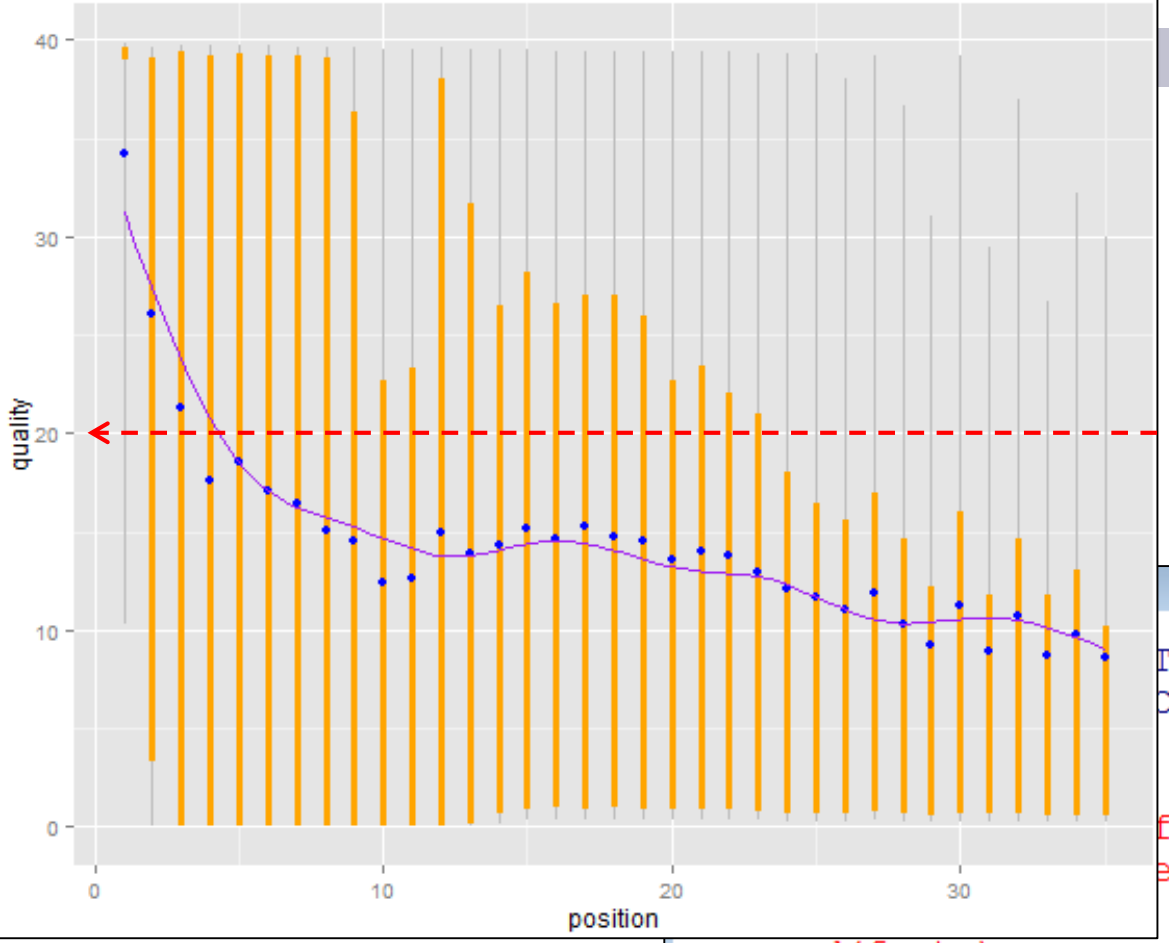
SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., 2010)。PHREDスコアが20未満のものがリード長に占める割合が0.1以上のリードを除去するやり方です。(例題のファイル中のリードは全て35bpのリードである。その10%以上ということで実質的にPHREDスコアが閾値未満のものが4塩基以上あるリードはダメということ) writeFastq関数のデフォルトオプションはcompress=Tで、gzip圧縮ファイルを出力します。ここではcompress=Fとして非圧縮ファイルを出力しています。

```
in_f <- "SRR037439.fastq"
out_f <- "hoge1.fastq"
param1 <- 20
param2 <- 0.1

#必要なパッケージをロード
library(ShortRead)

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

#本番
hoge <- as(quality(fastq), "matrix")
obj <- (rowSums(hoge < param1) <= width
fastq <- fastq[obj]
sread(fastq)

#ファイルに保存
writeFastq(fastq, out_f, compress=F)
```

Phredスコア20未満の塩基が配列長の10%以上を占めるリードを除去すると、たった5リードしか残らないという結果!

R Console

```
[499]     35 CGTTCTTGTTGCCCCCGGGGCGGGGGGGAAAAACC
[500]     35 GGAGCCTCCCCCCCCCCCCAAGGGGGGGGGGGGGC
>
> #本番
> hoge <- as(quality(fastq), "matrix")    #ASCIIコードのqual$
> obj <- (rowSums(hoge < param1) <= width(fastq)*param2)#条$
> fastq <- fastq[obj]                     #objがTRUEとなる要$
> sread(fastq)                            #配列情報を表示
  A DNAStringSet instance of length 5
    width seq
[1]     35 GGGACGGGGGGGGGGGGGGGGGGGCCCGGGGGG
[2]     35 TTTTTACGTATGCATATATATGATTATTCTTTTGG
[3]     35 GAACCATGTGCTAATTTTTCACAATTATTGACCTG
[4]     35 GATAGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
[5]     35 CGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
>
> #ファイルに保存
> writeFastq(fastq, out_f, compress=F)    #fastqの中身を指定$
>
```



• 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

QC段階で、特に3’側ではPhredスコア20未満の塩基だらけであることが読み取れるのでこの結果は極めて妥当

```
TGCCCCCGGGGCGGGGGGGAAAAACC
CCCCCCCCCCAAGGGGGGGGGGGGC


fastq), "matrix")    #ASCIIコードのqual$
e < param1) <= width(fastq)*param2)#条$
                     #objがTRUEとなる要$
                     #配列情報を表示
> sread(fastq)
  A DNAStringSet instance of length 5
    width seq
[1]    35 GGGACGGGGGGGGGGGGGGGGGGGCCCGGGGGG
[2]    35 TTTTTACGTATGCATATATATGATTATTCTTTTGG
[3]    35 GAACCATGTGCTAATTTTTCACAATTATTGACCTG
[4]    35 GATAGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
[5]    35 CGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
>
> #ファイルに保存
> writeFastq(fastq, out_f, compress=F)    #fastqの中身を指定$
>
```

フィルタリング

- イントロ | ファイル形式の変換 | qseq --> Sanger FASTQ (last modified 2013/08/19)
- 前処理 | クオリティチェック | について (last modified 2014/06/30)
- 前処理 | クオリティチェック | qrqc (last modified 2014/07/17)
- 前処理 | クオリティチェック | PHREDスコアに変換 (last modified 2013/06/18)
- 前処理 | クオリティチェック | 配列長分布を調べる (last modified 2013/06/18)
- 前処理 | フィルタリング | PHREDスコアが低い塩基をNに置換 (last modified 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 2014/03/03)
- 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) NEW
- 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 2014/07/17)
- 前処理 | フィルタリング | 指定した長さの範囲の配列を抽出 (last modified 2013/06/18)
- 前処理 | フィルタリング | 任意のIDを含む配列を抽出 (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 | 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) | 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(Lerch 20XX) (last modified 2014/06/13)
- 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)推奨
- 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 2014/08/08) NEW

Nを含まないリードのみ、許容するNの数を指定するやり方

配列長(リード長)でのフィルタリング

Contents

- 3-4. R Bioconductor II、2014/09/09 15:00-18:15、中級、実習
  - multi-FASTAファイルからの情報抽出（コンティグ数、総塩基数、N50、GC含量）
    - GC含量計算の詳細説明。alphabetFrequency, apply関数、数値行列計算の基本
    - コンティグごとのGC含量計算
  - FASTQ形式ファイルの読み込み
  - ファイル形式の変換：FASTQ → FASTA
  - クオリティチェック(クオリティコントロール；QC)
  - フィルタリング
    - クオリティスコア、N、配列長など
    - **動作確認用のサブセット作成**
  - その他（FASTA/FASTQファイルのdescription行を整形）

フィルタリング：サブセット作成

- イントロ | ファイル形式の変換 | qseq --> Sanger FASTQ (last modified 2013/08/19)
- 前処理 | クオリティチェック | について (last modified 2014/06/30)
- 前処理 | クオリティチェック | qrqc (last modified 2014/07/17)
- 前処理 | クオリティチェック | PHREDスコアに変換 (last modified 2013/06/18)
- 前処理 | クオリティチェック | 配列長分布を調べる (last modified 2013/06/18)
- 前処理 | フィルタリング | PHREDスコアが低い塩基をNに置換 (last modified 2014/0
- 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 201
- 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
- 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 modifi
- 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 2014/07/17)
- 前処理 | フィルタリング | 指定した長さの範囲の配列を抽出 (last modified 2013/06/18)
- 前処理 | フィルタリング | 任意のIDを含む配列を抽出 (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 | 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) | 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(Lerch 20XX) (last modified 2014/06/13)
- 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)推奨
- 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 2014/08/08) NEW

一般に数千～数億リードからなるNGSデータをテストなしにいきなり解析することはありません。本番前の動作確認を主目的として、最初の十万リードとか、ランダムに総リード数のx%分のサブセットなどを用いてpre解析を行うのがおそらく一般的です。サブセットの作成法を伝授します。

フィルタリング：サブセット作成

前処理 | フィルタリング | 任意のリード(サブセット)を抽出

FASTA形式やFASTQ形式ファイルを入力として、任意の配列(リード)を抽出するやり方を示します。ここのコードをテンプレートにして、マッピングなどを行う際に動作確認用として指定したリード数からなるサブセットを作成できます。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

最初のx個など、一定範囲を抽出するやり方の基本形

1. multi-FASTA

イントロ | 一般 | ...

```
in_f <- "hoge
out_f <- "hog
param <- 3

#必要なパッケー
library(Biost

#入力ファイルの
fasta <- read
fasta

#本番
obj <- 1:para
fasta <- fast
fasta

#ファイルに保存
writeXStringS
```

10. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., 2010)。(全部で500リードからなることが既知という前提で)5-20番目のリードを抽出するやり方です。

```
in_f <- "SRR037439.fastq"                  #入力ファイル名を指定してin_fに格納
out_f <- "hoge10.fastq"                    #出力ファイル名を指定してout_fに格納
param_range <- 5:20                        #抽出したいリードの範囲を指定

#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
fastq <- readFastq(in_f)                   #in_fで指定したファイルの読み込み
sread(fastq)                               #確認してるだけです(塩基配列情報を表示)

#本番
obj <- param_range                         #抽出したいリードの位置情報をobjに格納
fastq <- fastq[obj]                        #objがTRUEとなる要素のみ抽出した結果をfastqに格納
sread(fastq)                               #確認してるだけです(塩基配列情報を表示)

#ファイルに保存
writeFastq(fastq, out_f, compress=F)       #fastqの中身を指定したファイル名で保存
```

フィルタリング：サブセット作成

10. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., 2010)。(全部で500リードからなることが既知という前提で)5-20番目のリードを抽出するやり方です。

```
in_f <- "SRR037439.fastq"                #入力ファイル名を指定してin_fに格納
out_f <- "hoge10.fastq"                  #出力ファイル名を指定してout_fに格納
param_range <- 5:20                      #抽出したい
                                         #
#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
fastq <- readFastq(in_f)                 #in_fで指定
sread(fastq)                             #確認してる

#本番
obj <- param_range                       #抽出したい
fastq <- fastq[obj]                      #objがTRUE
sread(fastq)                             #確認してる

#ファイルに保存
writeFastq(fastq, out_f, compress=F)     #fastqの中
```

5-20番目の計16リードからなるサブセットになっており妥当

R Console

```
> #本番
> obj <- param_range                                    #抽出し$
> fastq <- fastq[obj]                                   #objがT$
> sread(fastq)                                          #確認し$
  A DNAStringSet instance of length 16
      width seq
 [1]     35 NNNNNNNNNNNNNNNGGTCGCCCCCGACGCCCACA
 [2]     35 NNNNNNNNNNNNNNNNACACGTCCCACCCCCCGCCC
 [3]     35 NNNNNNNNNNNNNNNNACATATCTGACCCTGTGCCG
 [4]     35 NNNNNNNNNNNNNNNNGGTGGGCCCCTAAGGACCTG
 [5]     35 NNNNNNNNNNNNNNNNACTCAACCCTCCCCGCCCCA
  ...    ... ...
[12]     35 NNNNNNNNNNNNNNNACTGGCGCGCGACTGTACAGC
[13]     35 NNNNNNNNNNNNNNNTTTTTCTTTTACACAGTGTTT
[14]     35 NNNNNNNNNNNNNNNGGGGGGGGGGGGGGGGGGGG
[15]     35 GNNNNNNNNNNNNNNGGGGGGGGGCGAGGGGGGCGG
[16]     35 CNNNNNNNNNNNNNNCCCCCGTCTACGCACCCCGAC
>
> #ファイルに保存
> writeFastq(fastq, out_f, compress=F)    #fastq$
>
```

フィルタリング：サブセット作成

10. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., 2010)。(全部で500リードからなることが既知という前提で)5-20番目のリードを抽出するやり方です。

```
in_f <- "SRR037439.fastq"          #入力ファイル名を指定してin_fに格納
out_f <- "hoge10.fastq"            #出力ファイル名を指定してout_fに格納
param_range <- 5:20                #抽出したいリードの範囲を指定
```

5-20番目の計16リードからなるサブセットになっており妥当

入力：SRR037439.fastq（500リード）

```
@SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35
NNNNNNNNNNNNNNCTACCCCCCCAGCCGCCGCA
+SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35
!!!!!!!!!!!!!!!"""""""""""""""#"""
@SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35
NNNNNNNNNNNNNNAGACAGTTGATTTAGCATAG
+SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35
!!!!!!!!!!!!!!!"+"""""""""""""""""&
@SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35
NNNNNNNNNNNNNNGGGTGGGGCGTTTGTTCTTG
+SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35
!!!!!!!!!!!!!!!/5$$&"""""%""""""""""
@SRR037439.4 HWI-E4_6_30ACL:2:1:0:1349 length=35
NNNNNNNNNNNNNNCCCCGCCCCGCCCCTCCCTC
+SRR037439.4 HWI-E4_6_30ACL:2:1:0:1349 length=35
!!!!!!!!!!!!!!!"0"("""&""""$""""""""
@SRR037439.5 HWI-E4_6_30ACL:2:1:0:1669 length=35
NNNNNNNNNNNNNNGGTCGCCCCCGACGCCCACA
+SRR037439.5 HWI-E4_6_30ACL:2:1:0:1669 length=35
```

出力：hoge10.fastq（16リード）

```
@SRR037439.5 HWI-E4_6_30ACL:2:1:0:1669 length=35
NNNNNNNNNNNNNNGGTCGCCCCCGACGCCCACA
+
!!!!!!!!!!!!!!!",""""#""""""""&"""&"
@SRR037439.6 HWI-E4_6_30ACL:2:1:0:1719 length=35
NNNNNNNNNNNNNNACACGTCCCACCCCCCGCCC
+
!!!!!!!!!!!!!!!"2%8""46%""*-*%%"'""
@SRR037439.7 HWI-E4_6_30ACL:2:1:0:1942 length=35
NNNNNNNNNNNNNNACATATCTGACCCTGTGCCG
+
!!!!!!!!!!!!!!!&.""/""+")'"("%""$)"
@SRR037439.8 HWI-E4_6_30ACL:2:1:0:2001 length=35
NNNNNNNNNNNNNNGGTGGGCCCCTAAGGACCTG
+
!!!!!!!!!!!!!!!IH8"'"""#""""""&""""")
@SRR037439.9 HWI-E4_6_30ACL:2:1:0:2032 length=35
```

フィルタリング：サブセット作成

11. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., 2010)。(全部で500リードからなることが既知という前提で)30リード分をランダムに非復元抽出するやり方です。writeFastq関数実行時にcompress=Fとしてgzip圧縮前のファイルを出力しています

```
in_f <- "SRR037439.fastq"                #入力ファイル名を指定してin_fに格納
out_f <- "hoge11.fastq"                  #出力ファイル名を指定してout_fに格納
param <- 30                              #ランダム抽

#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
fastq <- readFastq(in_f)                 #in_fで指定
sread(fastq)                             #確認してる

#本番
set.seed(1010)                           #おまじない
obj <- sample(1:length(fastq), param, replace=F)#
fastq <- fastq[sort(obj)]                #objで指定し
sread(fastq)                             #確認してる

#ファイルに保存
writeFastq(fastq, out_f, compress=F)     #fastqの中身
```

paramで指定したリード数をランダム抽出するやり方です

R Console

```
> set.seed(1010)                                  #おまじ$
> obj <- sample(1:length(fastq), param, replace=$
> fastq <- fastq[sort(obj)]                       #objで$
> sread(fastq)                                    #確認し$
  A DNAStringSet instance of length 30
      width seq
 [1]     35 NNNNNNNNNNNNNNAGACAGTTGATTTAGCATAG
 [2]     35 NNNNNNNNNNNNNNGGTCGCCCCCGACGCCCACA
 [3]     35 TNNNNNNNNNNNNNTTTTTATTTTTTTTTTTTTT
 [4]     35 CNNNNNNNNNNNNTGGCGTTGCTGTGGCGGGCGCG
 [5]     35 CNNNNNNNNNNNNCCCCAAAAAAAACAAAAAAAAC
 ...    ... ...
[26]     35 TGGGACGGACCCCCCCCACAAAAAAAACAGAGAGA
[27]     35 CATTCAGGGCAAGAACTTTTTTTGGGGGGGGGGGG
[28]     35 CAAAGGTTGGCGTGCCTGAGACAATATTTTTTGGT
[29]     35 CGGATCTTTTTTTTTTGTTTTCTCCCTTTTCCCCC
[30]     35 GTCAAAGATGAGGGGGGGGTTGGGGGGGGGGGAC
>
> #ファイルに保存
> writeFastq(fastq, out_f, compress=F)    #fastq$
>
```

フィルタリング：サブセット作成

11. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです
2010)。(全部で500リードからなることが既知という前提で)30リード分をランダムに非復元抽出す
writeFastq関数実行時にcompress=Fとしてgzip圧縮前のファイルを出力しています

```
in_f <- "SRR037439.fastq"          #入力ファイル名を指定してin_fに格納
out_f <- "hoge11.fastq"            #出力ファイル名を指定してout_fに格納
param <- 30                        #ランダム抽出したいリード数を指定
```

paramで指定した30リードをランダム抽出するやり方です。ランダム抽出であるにも関わらず、ほとんどのヒトが2, 5, 28, 36, 39, …と同じリードとなっているはずです。

入力：SRR037439.fastq（500リード）

```
@SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35
NNNNNNNNNNNNNNCTACCCCCCCAGCCGCCGCA
+SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35
!!!!!!!!!!!!!!!"""""""""""""""""#"""
@SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35
NNNNNNNNNNNNNNAGACAGTTGATTTAGCATAG
+SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35
!!!!!!!!!!!!!!!"+"""""""""""""""""&
@SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35
NNNNNNNNNNNNNNGGGTGGGGCGTTTGTTCTTG
+SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35
!!!!!!!!!!!!!!!/5$$&"""""%""""""""""
@SRR037439.4 HWI-E4_6_30ACL:2:1:0:1349 length=35
NNNNNNNNNNNNNNCCCCGCCCCGCCCCTCCCTC
+SRR037439.4 HWI-E4_6_30ACL:2:1:0:1349 length=35
!!!!!!!!!!!!!!!"0"("""&""""$""""""""
@SRR037439.5 HWI-E4_6_30ACL:2:1:0:1669 length=35
NNNNNNNNNNNNNNGGTCGCCCCCGACGCCCACA
+SRR037439.5 HWI-E4_6_30ACL:2:1:0:1669 length=35
```

出力：hoge11.fastq（30リード）

```
@SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35
NNNNNNNNNNNNNNAGACAGTTGATTTAGCATAG
+
!!!!!!!!!!!!!!"+"""""""""""""""""&
@SRR037439.5 HWI-E4_6_30ACL:2:1:0:1669 length=35
NNNNNNNNNNNNNNGGTCGCCCCCGACGCCCACA
+
!!!!!!!!!!!!!!",""""#""""""&"""&"
@SRR037439.28 HWI-E4_6_30ACL:2:1:1:799 length=35
TNNNNNNNNNNNNTTTTTATTTTTTTTTTTTTT
+
!!!!!!!!!!!!!"#"$/"0"/"*%,*,)//>+.
@SRR037439.36 HWI-E4_6_30ACL:2:1:1:1360 length=35
CNNNNNNNNNNNTGGCGTTGCTGTGGCGGGCGCG
+
5!!!!!!!!!!!"##""""""""""#""""""""
@SRR037439.39 HWI-E4_6_30ACL:2:1:1:1533 length=35
```

11. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., 2010)。(全部で500リードからなることが既知という前提で)30リード分をランダムに非復元抽出するやり方です。writeFastq関数実行時にcompress=Fとしてgzip圧縮前のファイルを出力しています

```
in_f <- "SRR037439.fastq"                #入力ファイル名を指定してin_fに格納
out_f <- "hoge11.fastq"                  #出力ファイル名を指定してout_fに格納
param <- 30                              #ランダム抽出したいリード数を指定

#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
fastq <- readFastq(in_f)                 #in_fで指定したファイルの読み込み
sread(fastq)                             #確認してるだけです(塩基配列情報を表示)

#本番
set.seed(1010)                           #おまじない(同じ乱数になるようにするため)
obj <- sample(1:length(fastq), param, replace=F)#リード数の数値の中からparamで指定した数
fastq <- fastq[sort(obj)]                #objで指定したリードのみソートして抽出した結果をfa
sread(fastq)                             #確認してるだけです(塩基配列情報を表示)

#ファイルに保存
writeFastq(fastq, out_f, compress=F)     #fastqの中
```

sample関数を用いて発生させた乱数からなる数値ベクトルobjそのままよりは、(必須ではないが)ソートさせたものを出力させようという思想。

R Console

```
> obj
 [1] 277  96  92 337 453 407 328 482 191 393 448
[12] 406 476 267   5  39 134 436 142  28 203 395
[23]  98 122  36 147 223   2 362 242
> sort(obj)
 [1]   2   5  28  36  39  92  96  98 122 134 142
[12] 147 191 203 223 242 267 277 328 337 362 393
[23] 395 406 407 436 448 453 476 482
> sort(obj, decreasing=T)
 [1] 482 476 453 448 436 407 406 395 393 362 337
[12] 328 277 267 242 223 203 191 147 142 134 122
[23]  98  96  92  39  36  28   5   2
>
```

11. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., 2010)。(全部で500リードからなることが既知という前提で)30リード分をランダムに非復元抽出するやり方です。writeFastq関数実行時にcompress=Fとしてgzip圧縮前のファイルを出力しています

処理 | フィルタリング | 任意のリード(サブセット)を抽出

```
in_f <- "SRR037439.fastq"                #入力ファイル名を指定してin_fに格納
out_f <- "hoge11.fastq"                  #出力ファイル名を指定してout_fに格納
param <- 30                              #ランダム抽出したいリード数を指定

#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
fastq <- readFastq(in_f)                 #in_fで指定したファイルの読み込み
sread(fastq)                             #確認してるだけです(塩基配列情報を表示)

#本番
set.seed(1010)                           #おまじない(同じ乱数になるようにするため)
obj <- sample(1:length(fastq), param, replace=F)#リード数の数値の中からparamで指定した数
fastq <- fastq[sort(obj)]                #objで指定したリードのみソートして抽出した結果をfa
sread(fastq)                             #確認してるだけです(塩基配列情報を表示)

#ファイルに保存
writeFastq(fastq, out_f, compress=F)     #fastqの中
```

(説明のため)黒枠部分を一旦コピペ

R Console

```
  [3]     35 NNNNNNNNNNNNNNNNGGGTGGGGCGTTTGTTCTTG
  [4]     35 NNNNNNNNNNNNNNNNCCCCGCCCCGCCCCTCCCTC
  [5]     35 NNNNNNNNNNNNNNNNGGTCGCCCCCGACGCCCACA
  ...    ... ...
[496]     35 CTGGACGGCCCCCCCCCCACACACCCACCCCCCCC
[497]     35 TGTCACTTGTGCTTTGCTCTTGTCCCACGGGGCTT
[498]     35 CCGCCCTTTTTCCAGAAATTTTCCGCACAAAAAAA
[499]     35 CGTTCTTGTTGCCCCCGGGGCGGGGGGGAAAAACC
[500]     35 GGAGCCTCCCCCCCCCCCCAAGGGGGGGGGGGGGC
>
> #本番
> set.seed(1010)                                #おまじ$
>
```

11. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., 2010)。(全部で500リードからなることが既知という前提で)30リード分をランダムに非復元抽出するやり方です。writeFastq関数実行時にcompress=Fとしてgzip圧縮前のファイルを出力しています

```
in_f <- "SRR037439.fastq"                #入力ファイル名を指定してin_fに格納
out_f <- "hoge11.fastq"                  #出力ファイル名を指定してout_fに格納
param <- 30                              #ランダム抽出したいリード数を指定

#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
fastq <- readFastq(in_f)                 #in_fで指定したファイルの読み込み
sread(fastq)                             #確認してるだけです(塩基配列情報を表示)

#本番
set.seed(1010)                           #おまじない(同じ乱数になるようにするため)
obj <- sample(1:length(fastq), param, replace=F)#リード数の数値の中からparamで指定した数
fastq <- fastq[sort(obj)]                #objで指定
sread(fastq)                             #確認してる

#ファイルに保存
writeFastq(fastq, out_f, compress=F)     #fastqの中
```

処理 | フィルタリング | 任意のリード(サブセット)を抽出

乱数発生部分の説明。1～500の整数の範囲からparamで指定した数だけ非復元抽出すべく、リード数に相当するlength(fastq)を利用している

R Console

```
> #本番
> set.seed(1010)                                    #おまじ$
> fastq
class: ShortReadQ
length:   500 reads; width:  35 cycles
> length(fastq)
[1] 500
> sample(1:500, 30, replace=F)
 [1] 277  96  92 337 453 407 328 482 191 393 448
[12] 406 476 267   5  39 134 436 142  28 203 395
[23]  98 122  36 147 223   2 362 242
> sample(1:500, 30, replace=F)
 [1] 108 212 103 418 289  26  85 313  83 154 262
[12] 197 373 292  82 187 329 312 119   7 370 436
[23] 180  61  70 106 184 356  16  41
>
```

rcode_20140909.txt

11. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., 2010)。(全部で500リードからなることが既知という前提で)30リード分をランダムに非復元抽出するやり方です。writeFastq関数実行時にcompress=Fとしてgzip圧縮前のファイルを出力しています

```
in_f <- "SRR037439.fastq"                #入力ファイル名を指定してin_fに格納
out_f <- "hoge11.fastq"                  #出力ファイル名を指定してout_fに格納
param <- 30                              #ランダム抽出したいリード数を指定

#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
fastq <- readFastq(in_f)                 #in_fで指定したファイルの読み込み
sread(fastq)                             #確認してるだけです(塩基配列情報を表示)

#本番
set.seed(1010)                           #おまじない(同じ乱数になるようにするため)
obj <- sample(1:length(fastq), param, replace=F)#リード数の数値の中からparamで指定した数
fastq <- fastq[sort(obj)]                #objで指定し
sread(fastq)                             #確認してる

#ファイルに保存
writeFastq(fastq, out_f, compress=F)     #fastqの中身
```

再現性のある乱数になっているのは、set.seed関数で任意の乱数のタネ番号(=1010)を指定しているから。この数値は1でも500でも1000でもなんでもよい。

R Console

```
> #本番
> set.seed(1010)                                    #おまじ$
> fastq
class: ShortReadQ
length: 500 reads; width: 35 cycles
> length(fastq)
[1] 500
> sample(1:500, 30, replace=F)
 [1] 277  96  92 337 453 407 328 482 191 393 448
[12] 406 476 267   5  39 134 436 142  28 203 395
[23]  98 122  36 147 223   2 362 242
> sample(1:500, 30, replace=F)
 [1] 108 212 103 418 289  26  85 313  83 154 262
[12] 197 373 292  82 187 329 312 119   7 370 436
[23] 180  61  70 106 184 356  16  41
>
```

rcode_20140909.txt

11. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., 2010)。(全部で500リードからなることが既知という前提で)30リード分をランダムに非復元抽出するやり方です。writeFastq関数実行時にcompress=Fとしてgzip圧縮前のファイルを出力しています

```
in_f <- "SRR037439.fastq"                #入力ファイル名を指定してin_fに格納
out_f <- "hoge11.fastq"                  #出力ファイル名を指定してout_fに格納
param <- 30                              #ランダム抽出したいリード数を指定

#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
fastq <- readFastq(in_f)                 #in_fで指定したファイルの読み込み
sread(fastq)                             #確認してるだけです(塩基配列情報を表示)

#本番
set.seed(1010)                           #おまじない(同じ乱数になるようにするため)
obj <- sample(1:length(fastq), param, replace=F)#リード数の数値の中からparamで指定した数
fastq <- fastq[sort(obj)]                #objで指定し
sread(fastq)                             #確認してる

#ファイルに保存
writeFastq(fastq, out_f, compress=F)     #fastqの中身
```

タネ番号を9にしてset.seed関数を実行したのち、乱数を発生。

R Console

```
> set.seed(9)
> sample(1:500, 30, replace=F)
 [1] 111  13 104 108 221  67 193 183 329 488  58
[12]   5 432 147 240 243 195 472 173 237 491  10
[23] 153  54 252 434 188 181 457 400
> sample(1:500, 30, replace=F)
 [1] 101 486 234 448   3 435 402 465 118 371 471
[12] 353 280  88 192  98 399 328 492 449 479 257
[23] 282 428 115 433 314 355 305  68
> sample(1:500, 30, replace=F)
 [1]  91 160 188  34 424 128 280 162 253 391  77
[12]  94 317 407 101  59 311 483 259 179 486 209
[23]  11 275 110 204 308  52 453 421
>
```

rcode_20140909.txt

11. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., 2010)。(全部で500リードからなることが既知という前提で)30リード分をランダムに非復元抽出するやり方です。writeFastq関数実行時にcompress=Fとしてgzip圧縮前のファイルを出力しています

```
in_f <- "SRR037439.fastq"                #入力ファイル名を指定してin_fに格納
out_f <- "hoge11.fastq"                  #出力ファイル名を指定してout_fに格納
param <- 30                              #ランダム抽出したいリード数を指定

#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
fastq <- readFastq(in_f)                 #in_fで指定したファイルの読み込み
sread(fastq)                             #確認してるだけです(塩基配列情報を表示)

#本番
set.seed(1010)                           #おまじない(同じ乱数になるようにするため)
obj <- sample(1:length(fastq), param, replace=F)#リード数の数値の中からparamで指定した数
fastq <- fastq[sort(obj)]                #objで指定し
sread(fastq)                             #確認してる

#ファイルに保存
writeFastq(fastq, out_f, compress=F)     #fastqの中身
```

タネ番号を9にしてset.seed関数を実行したのち、乱数を発生。set.seedはTCCなどBioconductorのパッケージでもしばしば目にするので覚えておくとよい。

R Console

```
> set.seed(1010)
> sample(1:500, 30, replace=F)
 [1] 277  96  92 337 453 407 328 482 191 393 448
[12] 406 476 267   5  39 134 436 142  28 203 395
[23]  98 122  36 147 223   2 362 242
> sample(1:500, 30, replace=F)
 [1] 108 212 103 418 289  26  85 313  83 154 262
[12] 197 373 292  82 187 329 312 119   7 370 436
[23] 180  61  70 106 184 356  16  41
> sample(1:500, 30, replace=F)
 [1]  75 357 485 420 219 486 311 356  74 358 313
[12] 483 192  32 170  66 138 246  78 155 169 367
[23] 316 324 222 175 491 221 116 476
>
```

rcode_20140909.txt

11. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., 2010)。(全部で500リードからなることが既知という前提で)30リード分をランダムに非復元抽出するやり方です。writeFastq関数実行時にcompress=Fとしてgzip圧縮前のファイルを出力しています

```
in_f <- "SRR037439.fastq"                #入力ファイル名を指定してin_fに格納
out_f <- "hoge11.fastq"                  #出力ファイル名を指定してout_fに格納
param <- 30                              #ランダム抽出したいリード数を指定

#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
fastq <- readFastq(in_f)                 #in_fで指定したファイルの読み込み
sread(fastq)                             #確認してるだけです(塩基配列情報を表示)

#本番
set.seed(1010)                           #おまじない(同じ乱数になるようにするため)
obj <- sample(1:length(fastq), param, replace=F)#リード数の数値の中からparamで指定した数
fastq <- fastq[sort(obj)]                #obj
sread(fastq)                             #確認

#ファイルに保存
writeFastq(fastq, out_f, compress=F)     #fast
```

原因既知状態でエラーを発生させている。500個の要素しかない状態での非復元抽出では、501個目のサンプリングは当然不可能。

R Console

```
> set.seed(1010)
> sample(1:500, 501, replace=F)
 以下にエラー sample.int(length(x), size, replace, prob) :
   'replace = FALSE' なので、母集団以上の大きさの標本は取ることができません
> sample(1:10, 5, replace=F)
[1]  6  2  9  5 10
> sample(1:10, 10, replace=F)
 [1]  9  6  8  3  5 10  4  7  2  1
> sample(1:10, 11, replace=F)
 以下にエラー sample.int(length(x), size, replace, prob) :
   'replace = FALSE' なので、母集団以上の大きさの標本は取ることができません
> sample(1:10, 11, replace=T)
 [1]  1  3 10  3  1  5  9  3  3  1  4
>
```

フィルタリング：サブセット作成

前処理 | フィルタリング | 任意のリード(サブセット)を抽出

FASTA形式やFASTQ形式ファイルを入力として、任意の配列(リード)を抽出するやり方を示します。
レートにして、マッピングなどを行う際に動作確認用として指定したリード数からなるサブセットを作成
「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

例えば例題2.をテンプレートにすることで、ヒトゲノムファイルを読み込んで特定の染色体のみ取り出すことができます。

1. multi-FASTAファイル(hoge4.fa)の場合:

イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたものです。最初の3リードを抽出するやり方です。

```
in_f <- "hoge4
out_f <- "hoge
param <- 3

#必要なパッケージを
library(Biostring

#入力ファイルの読み
fasta <- readDNAS
fasta

#本番
obj <- 1:param
fasta <- fasta[ob
fasta

#ファイルに保存
writeXStringSet(f
```

2. multi-FASTAファイル(hoge4.fa)の場合:

(全部で4リードからなることが既知という前提で)2-4番目のリードを抽出するやり方です。

```
in_f <- "hoge4.fa"                          #入力ファイル名を指定してin_fに格納
out_f <- "hoge2.fasta"                      #出力ファイル名を指定してout_fに格納
param_range <- 2:4                          #抽出したいリードの範囲を指定

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

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

#本番
obj <- param_range                          #抽出したいリードの位置情報をobjに格納
fasta <- fasta[obj]                         #objがTRUEとなる要素のみ抽出した結果をfastaに格納
fasta                                       #確認してるだけです

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

フィルタリング : サブセット作成

前処理 | フィルタリング | 任意のリード(サブセット)を抽出

FASTA形式やFASTQ形式ファイルを入力として、任意の配列(リード)を抽出するやり方を示します。
レートにして、マッピングなどを行う際に動作確認用として指定したリード数からなるサブセットを作
「ファイル」-「

8. small RNA-seqのgzip圧縮FASTQ形式ファイル(SRR609266.fa

イントロ | NGS | 配列取得 | FASTQ or SRALite | SRAdb(Zhu_20
タ(Nie et al., BMC Genomics, 2013)です。入力ファイルサイズは4
リード分をランダムに非復元抽出した結果をgzip圧縮なしで出力
同じもの(100000リード; 約16MB)になります。Macintoshではうま

```
in_f <- "SRR609266.fastq.gz"          #入力ファイル名を指定してin_fに格納
out_f <- "hoge8.fastq"                #出力ファイル名を指定してout_fに格納
param <- 100000                       #ランダム抽出したいリード数を指定

#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
fastq <- readFastq(in_f)              #in_fで指定したファイルの読み込み
id(fastq)                             #確認してるだけです(description情報を表示)

#本番
set.seed(1010)                        #おまじない(同じ乱数になるようにするため)
obj <- sample(1:length(fastq), param, replace=F)#リード数の数値の中からparamで指定した数だけ
fastq <- fastq[sort(obj)]             #objで指定したリードのみソートして抽出した結果をfastqに
id(fastq)                             #確認してるだけです(description情報を表示)

#ファイルに保存
writeFastq(fastq, out_f, compress=F)  #fastqの中身を指定したファイル名で保存
```

gzip圧縮ファイルを入力とすることもできる。writeFastq関数実行時にcompress=Tとすることでgzip圧縮ファイルで出力することも可能(同時に拡張子の追加や変更も忘れずに!)。Windowsではうまくいくが Macintoshではうまくいかないようです。

Contents

- 3-4. R Bioconductor II、2014/09/09 15:00-18:15、中級、実習
  - multi-FASTAファイルからの情報抽出（コンティグ数、総塩基数、N50、GC含量）
    - GC含量計算の詳細説明。alphabetFrequency, apply関数、数値行列計算の基本
    - コンティグごとのGC含量計算
  - FASTQ形式ファイルの読み込み
  - ファイル形式の変換：FASTQ → FASTA
  - クオリティチェック(クオリティコントロール；QC)
  - フィルタリング
    - クオリティスコア、N、配列長など
    - 動作確認用のサブセット作成
  - **その他（FASTA/FASTQファイルのdescription行を整形）**

その他

- イントロ | NGS | アノテーション情報取得 | TranscriptDb | について (last modified 2014/03/28)
- イントロ | NGS | アノテーション情報取得 | TranscriptDb | TxDb.*から (last modified 2013/10/08)
- イントロ | NGS | アノテーション情報取得 | TranscriptDb | GenomicFeatures(Lawrence 2013) (last
- イントロ | NGS | アノテーション情報取得 | TranscriptDb | GFF/GTF形式ファイルから (last modifi
- イントロ | NGS | 読み込み | FASTA形式 | 基本情報を取得 (last modified 2014/08/18) NEW
- 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 (last modified 2014/07/17)
- イントロ | NGS | 読み込み | FASTQ形式 | description行の記述を整形 modified 2013/06/13)
- イントロ | NGS | 読み込み | Illuminaの* seq.txt (last modified 2013/06/17)
- イントロ | NGS | 読み込み | Illuminaの* qseq.txt (last modified 2013/06/17)
- イントロ | ファイル形式の変換 | につ
- イントロ | ファイル形式の変換 | BAM
- イントロ | ファイル形式の変換 | FAS
- イントロ | ファイル形式の変換 | Gen
- イントロ | ファイル形式の変換 | qseq
- イントロ | ファイル形式の変換 | qseq
- イントロ | ファイル形式の変換 | qseq
- 前処理 | クオリティチェック | につい
- 前処理 | クオリティチェック | qrqc (la
- 前処理 | クオリティチェック | PHRED
- 前処理 | クオリティチェック | 配列長

一般にPerlというプログラミング言語で習うようなテクニックもRで可能です。

イントロ | NGS | 読み込み | FASTQ形式 | description行の記述を整形 NEW

FASTQ形式ファイルに対して、「目的の記述部分のみ抽出し、それを新たなdescription」としてFASTQ形式で保存するやり方などを示します。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

1. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., 2010)。
抽出例:「SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35」-->「SRR037439.1」
戦略: description行の" "を区切り文字として分割("SRR037439.1"と"HWI-E4_6_30ACL:2:1:0:176"と"length=35")し、分割後の1つ目の要素を抽出
writeFastq関数実行時にcompress=Fとして非圧縮FASTQ形式ファイルを出力しています。

```
in_f <- "SRR037439.fastq"                #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.fastq"                   #出力ファイル名を指定してout_fに格納
param1 <- " "                            #区切り文字を指定
param2 <- 1                              #分割後に抽出したい要素番号を指定

#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
fastq <- readFastq(in_f)                 #in_fで指定したファイルの読み込み
id(fastq)                                #description情報を表示
```

1. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., 2010)。
抽出例:「SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35」-->「SRR037439.1」
戦略: description行の" "を区切り文字として分割("SRR037439.1"と"HWI-E4_6_30ACL:2:1:0:176"と"length=35")し、分割後の1つ目の要素を抽出
writeFastq関数実行時にcompress=Fとして非圧縮FASTQ形式ファイルを出力しています。

```
in_f <- "SRR037439.fastq"          #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.fastq"             #出力ファイル名を指定してout_fに格納
param1 <- " "                      #区切り文字を指定
param2 <- 1                        #分割後に抽出したい要素番号を指定
```

マッピング結果のファイルサイズ削減に効果的

入力:SRR037439.fastq

```
@SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35
NNNNNNNNNNNNNNCTACCCCCCCCAGCCGCCGCA
+SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35
!!!!!!!!!!!!!!!"""""""""""""""""#"""
@SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35
NNNNNNNNNNNNNNAGACAGTTGATTTAGCATAG
+SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35
!!!!!!!!!!!!!!!"+"""""""""""""""""&
@SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35
NNNNNNNNNNNNNNGGGTGGGGCGTTTGTTCTTG
+SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35
!!!!!!!!!!!!!!!/5$$&"""""%""""""""""
@SRR037439.4 HWI-E4_6_30ACL:2:1:0:1349 length=35
NNNNNNNNNNNNNNCCCCGCCCCGCCCCTCCCTC
+SRR037439.4 HWI-E4_6_30ACL:2:1:0:1349 length=35
!!!!!!!!!!!!!!!"0"("""&""""$""""""""""
@SRR037439.5 HWI-E4_6_30ACL:2:1:0:1669 length=35
```

出力:hoge1.fastq

```
@SRR037439.1
NNNNNNNNNNNNNNCTACCCCCCCCAGCCGCCGCA
+
!!!!!!!!!!!!!!!"""""""""""""""""#"""
@SRR037439.2
NNNNNNNNNNNNNNAGACAGTTGATTTAGCATAG
+
!!!!!!!!!!!!!!!"+"""""""""""""""""&
@SRR037439.3
NNNNNNNNNNNNNNGGGTGGGGCGTTTGTTCTTG
+
!!!!!!!!!!!!!!!/5$$&"""""%""""""""""
@SRR037439.4
NNNNNNNNNNNNNNCCCCGCCCCGCCCCTCCCTC
+
!!!!!!!!!!!!!!!"0"("""&""""$""""""""""
@SRR037439.5
```

5. サンプルデータ7のFASTQ形式ファイル(SRR037439.fastq)の場合:

SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです(Bullard et al., 2010)。
抽出例:「SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35」-->「SRR037439.1」
戦略: description行の" "を区切り文字として分割("SRR037439.1"と"HWI-E4_6_30ACL:2:1:0:176"と"length=35")し、分割後の1つ目の要素を抽出。次に、"."を区切り文字として分割("SRR037439"と"1")し、分割後の2つ目の要素を抽出
writeFastq関数実行時にcompress=Fとして非圧縮FASTQ形式ファイルを出力しています。

```
in_f <- "SRR037439.fastq"          #入力ファイル名を指定してin_fに格納
out_f <- "hoge5.fastq"             #出力ファイル名を指定してout_fに格納
param1 <- " "                      #区切り文字を指定
param2 <- 1                        #分割後に抽出したい要素番号を指定
param3 <- "."                      #区切り文字を指定
param4 <- 2                        #分割後に抽出したい要素番号を指定
```

リードのシリアル番号のみにすることも可能。このテクニックは、RefSeqなどのトランスクリプトーム配列からバージョン情報を除いたRefSeq ID部分のみの抽出などに転用可能。

入力:SRR037439.fastq

```
@SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35
NNNNNNNNNNNNNNCTACCCCCCCCAGCCGCCGCA
+SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35
!!!!!!!!!!!!!!!"""""""""""""""""#"""
@SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35
NNNNNNNNNNNNNNAGACAGTTGATTTAGCATAG
+SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35
!!!!!!!!!!!!!!!"+"""""""""""""""""&
@SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35
NNNNNNNNNNNNNNGGGTGGGGCGTTTGTTCTTG
+SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35
!!!!!!!!!!!!!!!/5$$&"""%""""""""""
@SRR037439.4 HWI-E4_6_30ACL:2:1:0:1349 length=35
NNNNNNNNNNNNNNCCCCGCCCCGCCCCTCCCTC
+SRR037439.4 HWI-E4_6_30ACL:2:1:0:1349 length=35
!!!!!!!!!!!!!!!"0"(""&""""$""""""""
@SRR037439.5 HWI-E4_6_30ACL:2:1:0:1669 length=35
```

出力:hoge5.fastq

```
@1
NNNNNNNNNNNNNNCTACCCCCCCCAGCCGCCGCA
+
!!!!!!!!!!!!!!!"""""""""""""""""#"""
@2
NNNNNNNNNNNNNNAGACAGTTGATTTAGCATAG
+
!!!!!!!!!!!!!!!"+"""""""""""""""""&
@3
NNNNNNNNNNNNNNGGGTGGGGCGTTTGTTCTTG
+
!!!!!!!!!!!!!!!/5$$&"""%""""""""""
@4
NNNNNNNNNNNNNNCCCCGCCCCGCCCCTCCCTC
+
!!!!!!!!!!!!!!!"0"(""&""""$""""""""
@5
```