

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

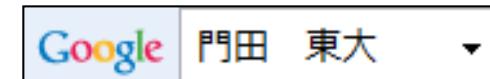
3. データ解析基礎 | 3-4. R Bioconductor I

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 I、2014/09/09 10:30–14:45、中級、実習
 - Tips : setwd関数を利用した効率的な作業ディレクトリの変更
 - データの型1 : translate関数の入力情報(AAStringSet)
 - Tips : オブジェクトの消去
 - データの型2 : 翻訳配列取得のコードの中身を解説
 - バージョン情報把握とバージョンアップ
 - バージョン違いの影響
 - 過去から現在 : BiostingsパッケージのreadDNAStringSet関数
 - 現在から未来 : BSgenome.Hsapiens.UCSC.hg19パッケージでプロモーター配列取得
 - Bioconductor概観



- ・ イントロ | 一般 | 任意の長さの可能な全ての塩基配列を作成 (last modified 2013/06/14)
- ・ イントロ | 一般 | 任意の位置の塩基を置換 (last modified 2013/09/12)
- ・ イントロ | 一般 | 指定した範囲の配列を取得 (last modified 2014/03/08)
- ・ イントロ | 一般 | 指定したID(染色体やdescriptor)の配列を取得 (last modified 2014/03/10)
- ・ イントロ | 一般 | 翻訳配列(translate)を取得 (last modified 2013/06/14)
- ・ イントロ | 一般 | 相補鎖(complement)を取得 (last modified 2013/06/14)
- ・ イントロ | 一般 | 逆相補鎖(reverse complement)を取得 (last modified 2013/06/14)

- ・ イントロ | 一般 | 翻訳配列(translate)を取得

イントロ | 一般 | 翻訳配列(translate)を取得 NEW

塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。

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

1. FASTA形式ファイル(sample1.fasta)の

```
in_f <- "sample1.fasta"
out_f <- "hogel.fasta"

#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_)

#本番
hoge <- translate(fasta)
names(hoge) <- names(fasta)
fasta <- hoge
fasta

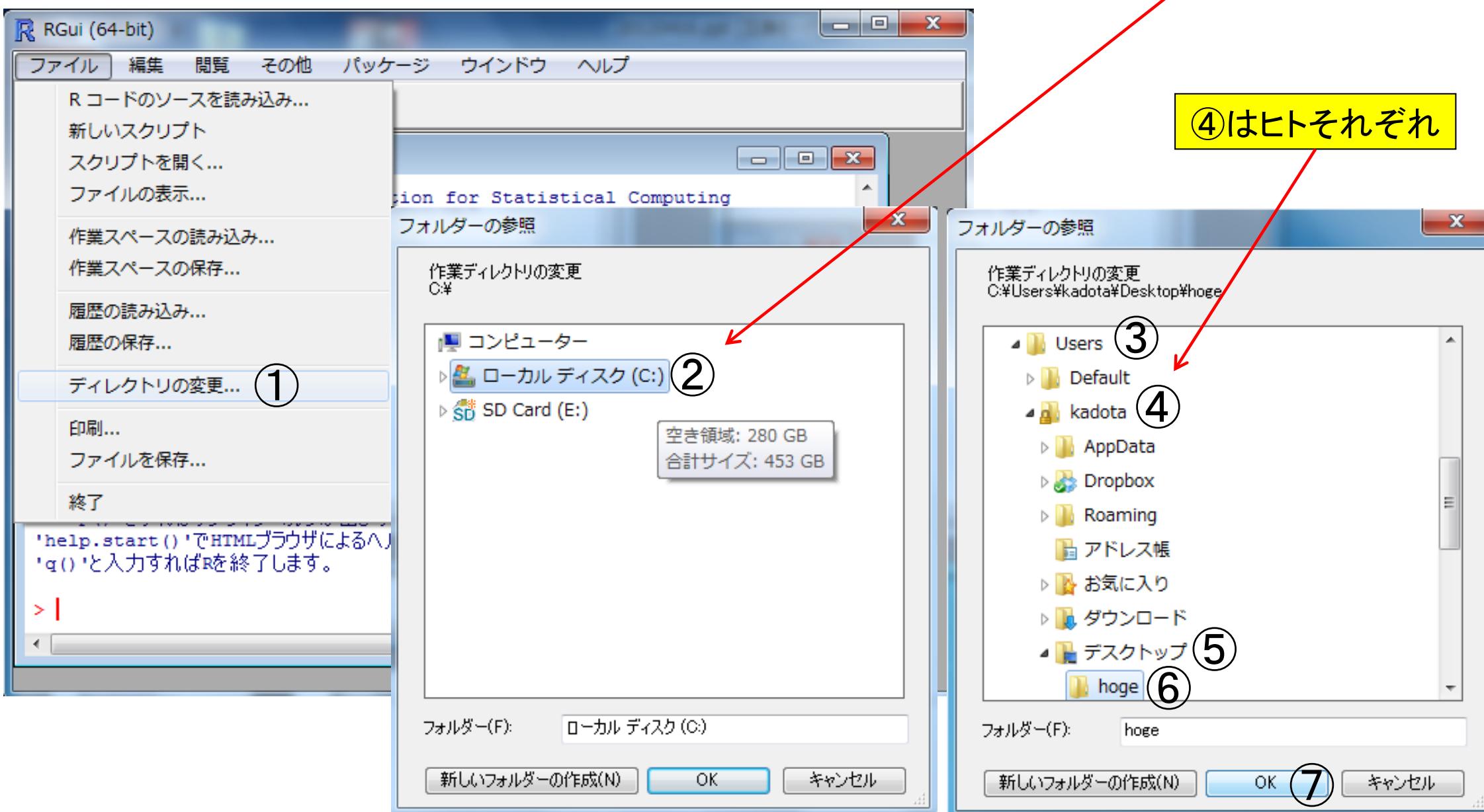
#ファイルに保存
writeXStringSet(fasta, file=o
```

```
rcode_translate.txt x
#####
### 翻訳配列取得
#####
in_f <- "sample1.fasta"
out_f <- "hogel.fasta"
#
#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, format="fasta")#in_fで指定したファイルの読み込み
#
#本番
hoge <- translate(fasta)
names(hoge) <- names(fasta)
fasta <- hoge
fasta
#
#ファイルに保存
writeXStringSet(fasta, file=out_f, format="fasta", width=50)#fastaの中身を指定
```

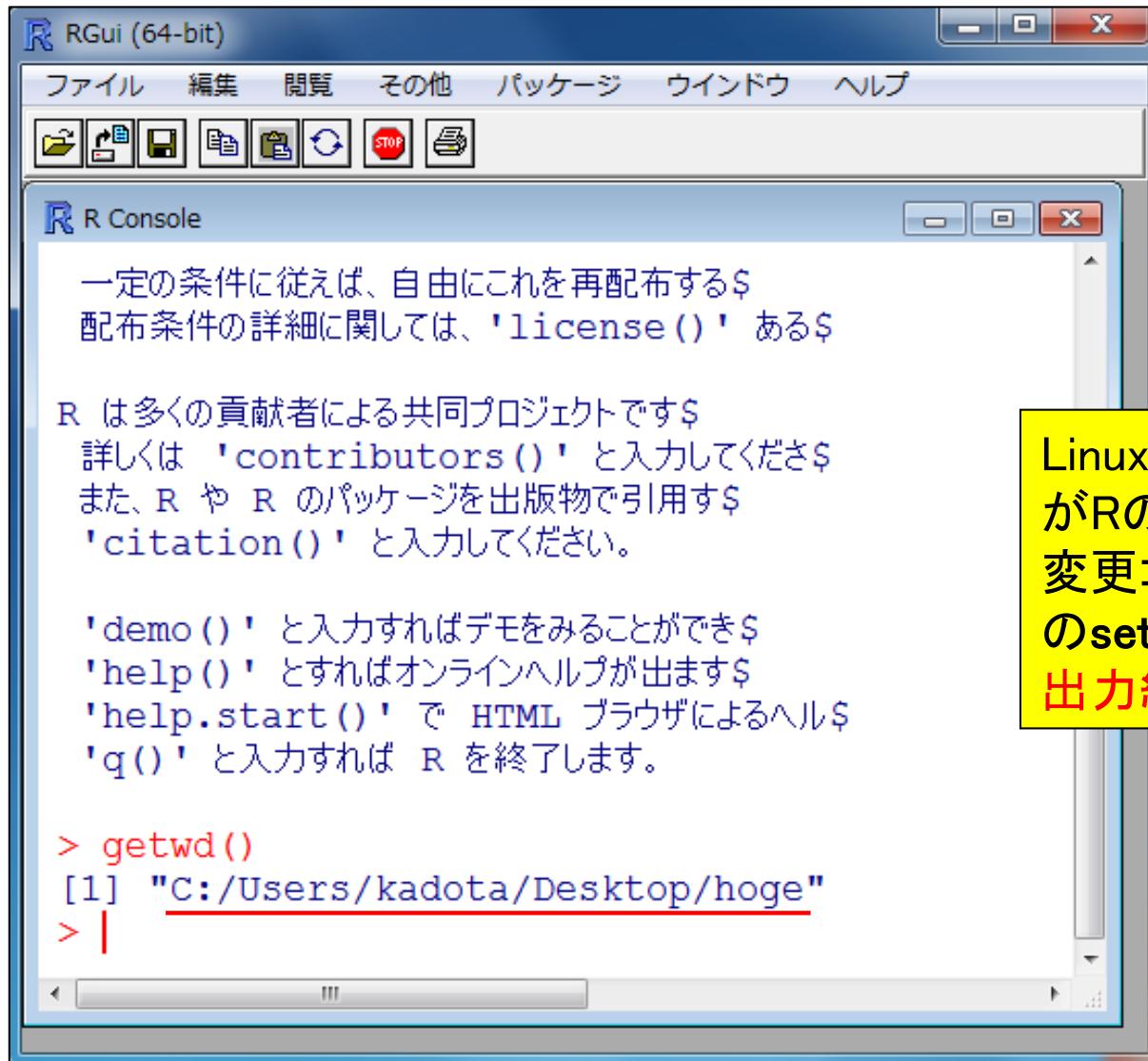
Rに慣れてくると、Gui画面上での煩雑な「ファイル - ディレクトリの変更」作業を効率的に行いたくなります。

作業ディレクトリの変更

「Windows(C:)」となっている場合もあるが、気にしない



作業ディレクトリの変更



Linuxのpwdコマンドに相当する
のがRのgetwd()。Linuxのディレクトリ
変更コマンドcdに相当するのがR
のsetwd()。重要なのは、getwd()の
出力結果を把握しておくことです。

デスクトップのhogeと決めるなら…

ファイル名 : rcode_20140909.txt

```
#####
### Tips (作業ディレクトリの変更)↓
### Windows PCでユーザがkadotaかiuの場合↓
#####
```

```
getwd()          # R起動直後の作業ディレクトリを表示↓
setwd("C:/Users/kadota/Desktop/hoge") # デスクトップ上のhogeフォルダにしたい場合↓
getwd()          # 作業ディレクトリが変更できているか確認↓
```

```
↓
getwd()          # R起動直後の作業ディレクトリを表示↓
setwd("C:/Users/iu/Desktop/hoge") # デスクトップ上のhogeフォルダにしたい場合↓
getwd()          # 作業ディレクトリが変更できているか確認↓
```

```
↓
setwd("C:/Users/kadota/Desktop")
setwd("C:/Users/kadota/Documents")
↓
```

Rを再起動してコピペ。setwd関数実行によって、作業ディレクトリの変更がコピペで完了していることが分かる。

The screenshot shows the R Console window with the following text:

```
R Console
#
# 'help.start()' で HTML ブラウザによるヘルプがみられます。
'q()' と入力すれば R を終了します。

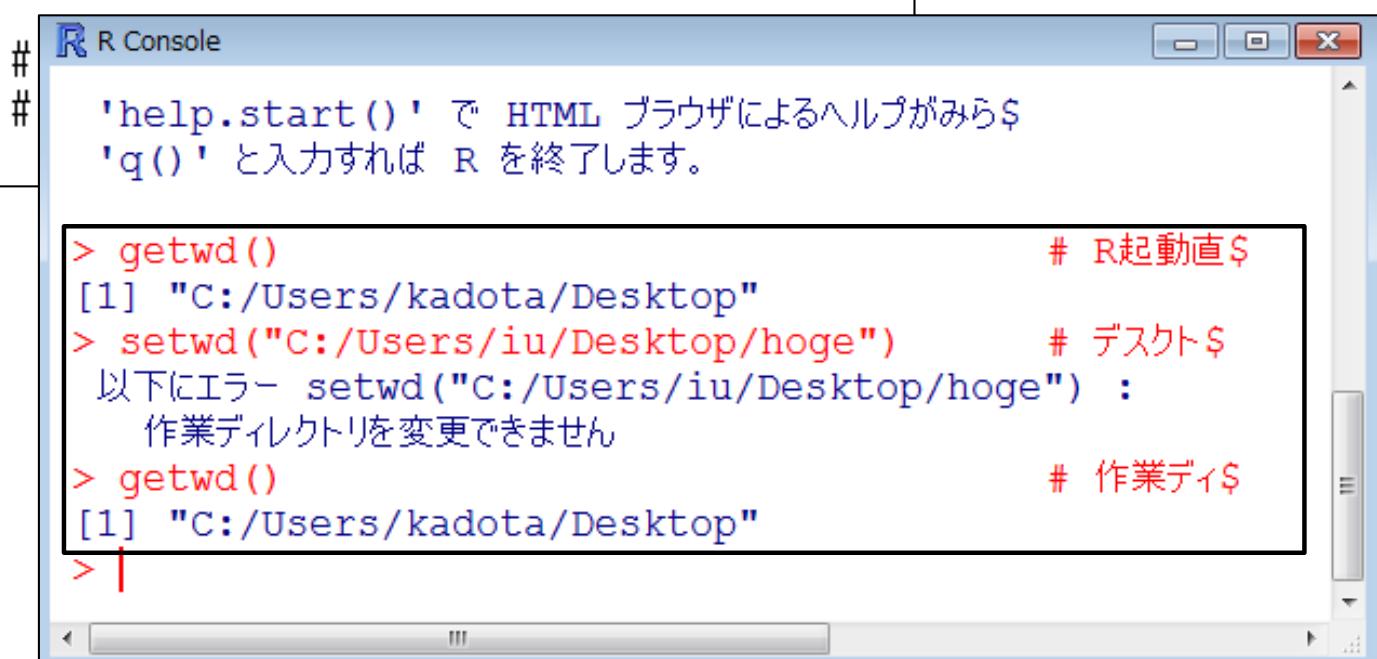
> getwd()          # R起動直後
[1] "C:/Users/kadota/Desktop"
> setwd("C:/Users/kadota/Desktop/hoge") # デスクトップ上のhogeフォルダにしたい場合
> getwd()          # 作業ディレクトリが変更できているか確認
[1] "C:/Users/kadota/Desktop/hoge"
>
```

デスクトップのhogeと決めるなら…

ファイル名 : rcode_20140909.txt

```
#####
### Tips (作業ディレクトリの変更)↓
### Windows PCでユーザがkadotaかiuの場合↓
#####
getwd()          # R起動直後の作業ディレクトリを表示↓
setwd("C:/Users/kadota/Desktop/hoge") # デスクトップ上のhogeフォルダにしたい場合↓
getwd()          # 作業ディレクトリが変更できているか確認↓
↓
getwd()          # R起動直後の作業ディレクトリを表示↓
setwd("C:/Users/iu/Desktop/hoge")    # デスクトップ上のhogeフォルダにしたい場合↓
getwd()          # 作業ディレクトリが変更できているか確認↓
↓
setwd("C:/Users/kadota/Desktop")
setwd("C:/Users/kadota/Documents")
```

アグリバイオ貸与PCはユーザ名がiu
なのでエラーが出ないと思います。



R Console

```
# 'help.start()' で HTML ブラウザによるヘルプがみら$
# 'q()' と入力すれば R を終了します。
```

```
> getwd()          # R起動直$
[1] "C:/Users/kadota/Desktop"
> setwd("C:/Users/iu/Desktop/hoge")      # デスクト$
以下にエラー setwd("C:/Users/iu/Desktop/hoge") :
  作業ディレクトリを変更できません
> getwd()          # 作業ディレクトリを表示$
[1] "C:/Users/kadota/Desktop"
> |
```

自分で解析していた頃のコード例

イントロ | 一般 | 翻訳配列(translate)を取得 NEW

塩基配列を読み込んでアミノ酸配列に翻訳するや
「ファイル」-「ディレクトリの変更」で解析したいファ

1. FASTA形式ファイル([sample1.fasta](#))の場合:

```
in_f <- "sample1.fasta"
out_f <- "hoge1.fasta"

#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, form="fasta")

#本番
hoge <- translate(fasta)
names(hoge) <- names(fasta)
fasta <- hoge
fasta

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

ファイル名 : rcode_translate2.txt

```
#####
### 作業ディレクトリの変更↓
#####
setwd("C:/Users/kadota/Desktop/hoge") # デスクトップ上のhogeフォルダにしたい場合↓
#setwd("C:/Users/kadota/Desktop")      # 「デスクトップ」にしたい場合↓
#setwd("C:/Users/kadota/Documents")    # 「マイドキュメント」にしたい場合↓
↓
#####
### 翻訳配列取得↓
#####
in_f <- "sample1.fasta"                  # 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)
↓
#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 <- translate(fasta)
names(hoge) <- names(fasta)
fasta <- hoge
fasta

#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↓
```

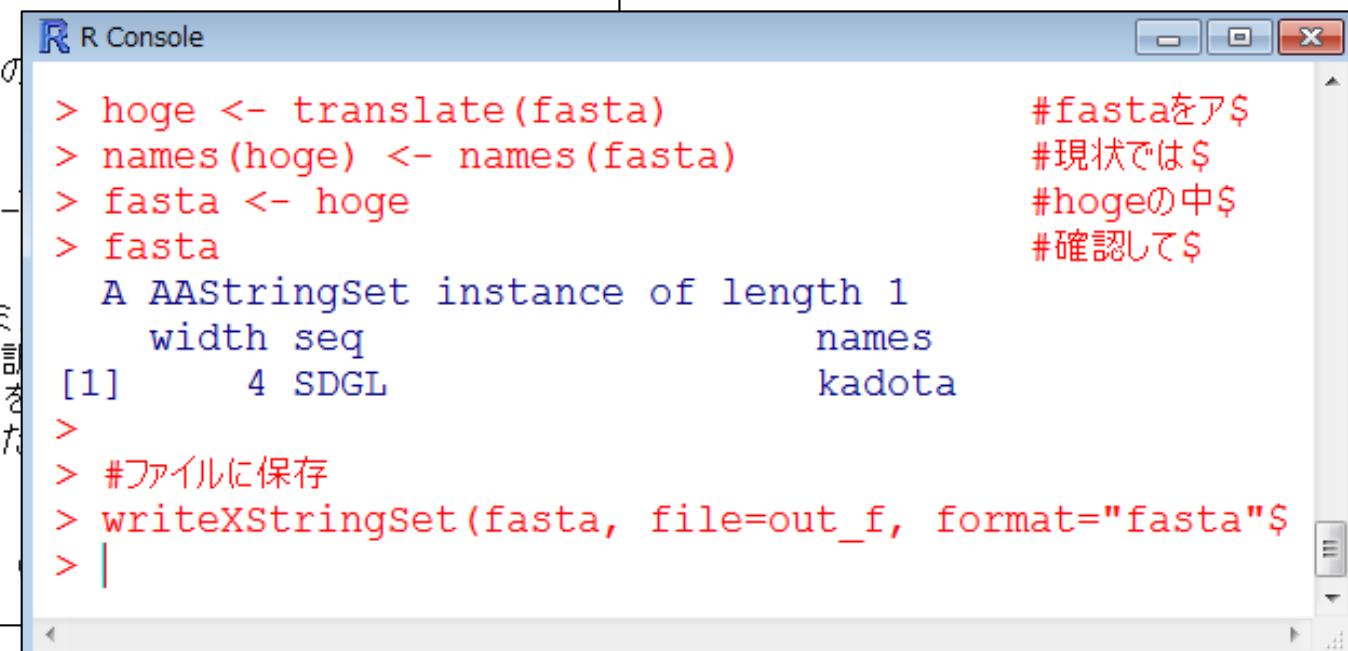
作業ディレクトリ変更を含んだ一連のコードを用意しておき、Rを起動直後にコピペ

自分で解析していた頃のコード例

ファイル名 : rcode_translate2.txt

```
#####
### 作業ディレクトリの変更↓
#####
setwd("C:/Users/kadota/Desktop/hoge") # テスクトップ上のhogeフォルダにしたい場合↓
#setwd("C:/Users/kadota/Desktop")      # 「デスクトップ」にしたい場合↓
#setwd("C:/Users/kadota/Documents")    # 「マイドキュメント」にしたい場合↓
↓
#####
### 翻訳配列取得↓
#####
in_f <- "sample1.fasta"                # 入力ファイル名を指定して in_f に格納↓
out_f <- "hogel.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)
↓
#入力ファイルの読み込み↓
fasta <- readDNAStringSet(in_f, format="fasta")#in_↓
#本番↓
hoge <- translate(fasta)               #fastaをアミ↑
names(hoge) <- names(fasta)            #現状では翻↑
fasta <- hoge                          #hogeの中身を↑
fasta                                     #確認してるた↑
↓
#ファイルに保存↓
writeXStringSet(fasta, file=out_f, format="fasta", ↓
```

いくつかのテンプレートとして使
う可能性のあるものを予め用意
しておき、実際に使うもののみ#
を外して利用していました。



R Console window showing the execution of the R code and its output.

```
R Console
> hoge <- translate(fasta)          #fastaをアミ↑
> names(hoge) <- names(fasta)       #現状では翻↑
> fasta <- hoge                   #hogeの中身を↑
> fasta                           #確認してるた↑
A AAStringSet instance of length 1
  width seq                         names
  [1] 4 SDGL                         kadota
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta")$#
> |
```

塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。

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

1. FASTA形式ファイル([sample1.fasta](#))の場合:

```

in_f <- "sample1.fasta"          #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

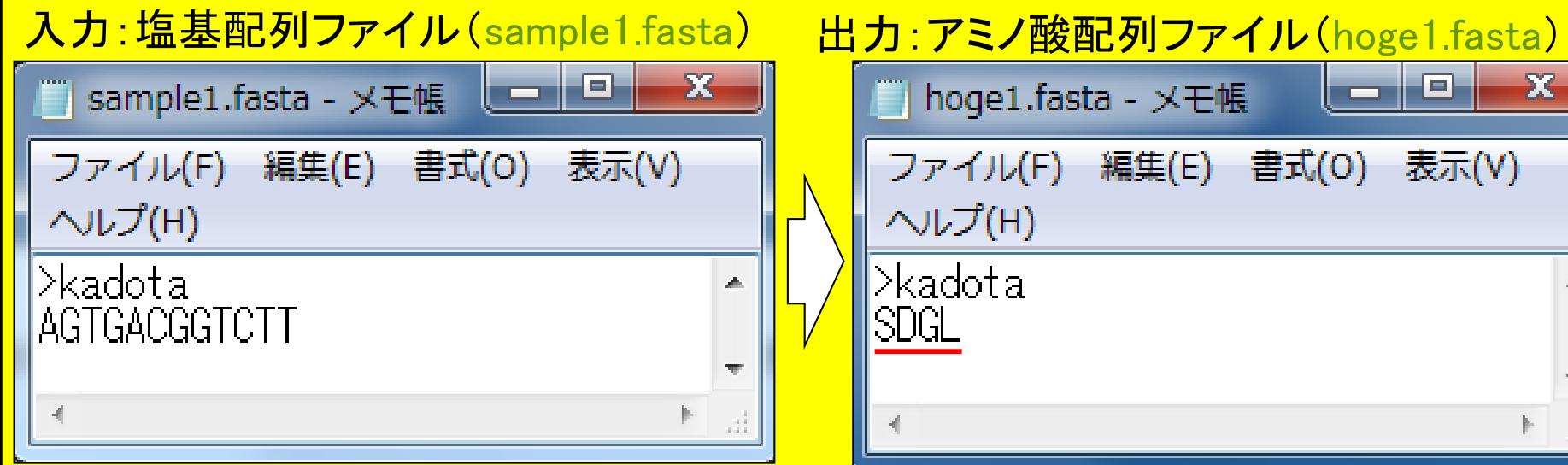
#必要なパッケージをロード
library(Biostrings)              #パッケージの読み込み

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

#本番
hoge <- translate(fasta)         #fastaをアミノ酸配列に翻訳した結果をhogeに格納
names(hoge) <- names(fasta)       #現状では翻訳した結果のオブジェクトhogeのdescriptor
fastaa <- hoge                   #hogeの中身をfastaaに格納
fastaa                                #確認してるだけです

#ファイルに保存
writeXStringSet(fasta, out_f)

```



Contents

- 3-4. R Bioconductor I、2014/09/09 10:30–14:45、中級、実習
 - Tips : setwd関数を利用した効率的な作業ディレクトリの変更
 - データの型1 : translate関数の入力情報(AAStringSet)
 - Tips : オブジェクトの消去
 - データの型2 : 翻訳配列取得のコードの中身を解説
 - バージョン情報把握とバージョンアップ
 - バージョン違いの影響
 - 過去から現在 : BiostingsパッケージのreadDNAStringSet関数
 - 現在から未来 : BSgenome.Hsapiens.UCSC.hg19パッケージでプロモーター配列取得
 - Bioconductor概観



データの型1

ファイル名 : rcode_translate2.txt

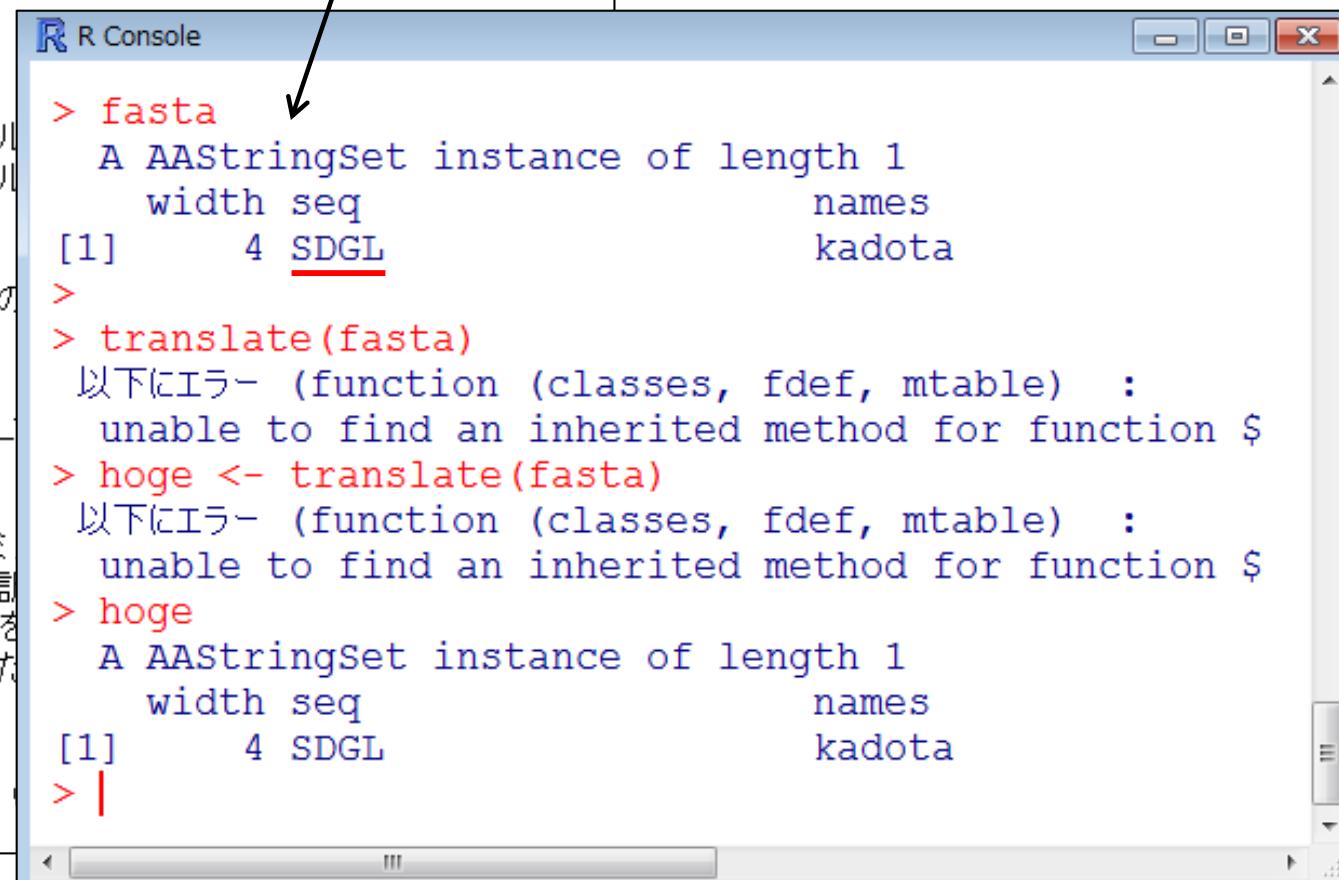
```
#####
### 作業ディレクトリの変更↓
#####
setwd("C:/Users/kadota/Desktop/hoge") # デスクトップ上のhogeフォルダにしたい場合↓
#setwd("C:/Users/kadota/Desktop")      # 「デスクトップ」にしたい場合↓
#setwd("C:/Users/kadota/Documents")    # 「マイドキュメント」にしたい場合↓

#####
### 翻訳配列取得↓
#####
in_f <- "sample1.fasta"
out_f <- "hogel.fasta"
#
#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, format="fasta")#in_
#
#本番↓
hoge <- translate(fasta)
names(hoge) <- names(fasta)
fasta <- hoge
fasta
#
#ファイルに保存↓
writeXStringSet(fasta, file=out_f, format="fasta",

```

#入力ファイル
#出力ファイル
#パッケージの
#fastaをアミ
#現状では翻訳
#hogeの中身を
#確認してた

左記のコード実行後のfastaオブジェクトにはアミノ酸配列情報が格納されていることが分かる。AAStringSetのAAはAmino Acids (アミノ酸)の略。AAStringSetクラスオブジェクトとかAAStringSetクラスなどと呼ばれるが、実用上はAAStringSetというのまたは型という程度の理解でよい。



The screenshot shows the R Console window with the following session:

```
R Console
> fasta
A AAStringSet instance of length 1
  width seq                               names
[1] 4 SDGL                                kadota
>
> translate(fasta)
以下にエラー (function (classes, fdef, mtable) :
  unable to find an inherited method for function $ 
> hoge <- translate(fasta)
以下にエラー (function (classes, fdef, mtable) :
  unable to find an inherited method for function $ 
> hoge
A AAStringSet instance of length 1
  width seq                               names
[1] 4 SDGL                                kadota
> |
```

データの型1

ファイル名 : rcode_translate2.txt

```
#####
### 作業ディレクトリの変更↓
#####
setwd("C:/Users/kadota/Desktop/hoge") # デスクトップ上のhogeフォルダにした
#setwd("C:/Users/kadota/Desktop")      # 「デスクトップ」にしたい場合↓
#setwd("C:/Users/kadota/Documents")    # 「マイドキュメント」にしたい場合↓
↓

#####
### 翻訳配列取得↓
#####
in_f <- "sample1.fasta"
out_f <- "hogel.fasta"
↓
#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, format="fasta")#in_
↓
#本番↓
hoge <- translate(fasta)
names(hoge) <- names(fasta)
fasta <- hoge
fasta
↓
#ファイルに保存↓
writeXStringSet(fasta, file=out_f, format="fasta",
↓
```

#入力ファイル
#出力ファイル
#パッケージの
#fastaをアミ
#現状では翻訳
#hogeの中身を
#確認してた

fastaオブジェクトを入力としてtranslate関数を実行してもエラーが出る。この理由は明確。translate関数は塩基配列を入力としてアミノ酸配列を出力するものであり、アミノ酸配列情報からなるfastaオブジェクトを入力とするのは想定外だから。

```
R Console
> fasta
A AAStringSet instance of length 1
  width seq
[1] 4 SDGL
names
kadota
>
> translate(fasta)
以下にエラー (function (classes, fdef, mtable) :
  unable to find an inherited method for function $ 
> hoge <- translate(fasta)
以下にエラー (function (classes, fdef, mtable) :
  unable to find an inherited method for function $ 
> hoge
A AAStringSet instance of length 1
  width seq
[1] 4 SDGL
names
kadota
> |
```

データの型1

ファイル名 : rcode_translate2.txt

```
#####
### 作業ディレクトリの変更↓
#####
setwd("C:/Users/kadota/Desktop/hoge") # デスクトップ上のhogeフォルダにしたい場合↓
#setwd("C:/Users/kadota/Desktop") # 「デスクトップ」にしたい場合↓
#setwd("C:/Users/kadota/Documents") # 「マイドキュメント」にしたい場合↓
#
#####
### 翻訳配列取得↓
#####
in_f <- "sample1.fasta"
out_f <- "hogel.fasta"
#
#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, format="fasta")#in_
#
#本番↓
hoge <- translate(fasta)
names(hoge) <- names(fasta)
fasta <- hoge
fasta
#
#ファイルに保存↓
writeXStringSet(fasta, file=out_f, format="fasta",

```

#入力ファイル
#出力ファイル
#パッケージの
#fastaをアミ
#現状では翻訳
#hogeの中身を
#確認してた

塩基配列情報からなるfasta
オブジェクトを入力としたとき
には正常に動作していた。

当然ながら、アミノ酸配列情報からなるfastaオブジェクトを入力としている以上、それと全く同じコードに変更してもエラーが出る。

```
R Console
> fasta
A AAStringSet instance of length 1
  width seq
  [1] 4 SDGL
  names
  kadota
>
> translate(fasta)
以下にエラー (function (classes, fdef, mtable) :
  unable to find an inherited method for function $ 
> hoge <- translate(fasta)
以下にエラー (function (classes, fdef, mtable) :
  unable to find an inherited method for function $ 
> hoge
A AAStringSet instance of length 1
  width seq
  [1] 4 SDGL
  names
  kadota
> |
```

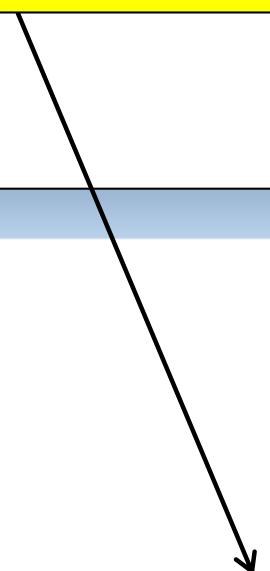
データの型1

ファイル名 : rcode_translate2.txt

```
#####
#### 作業ディレクトリの変更↓
#####
setwd("C:/Users/kadota/Desktop/hoge") # デスクトップ上のhogeフォルダにしたい場合↓
#setwd("C:/Users/kadota/Desktop")      # 「デスクトップ」にしたい場合↓
#setwd("C:/Users/kadota/Documents")    # 「マイドキュメント」にしたい場合↓
↓
#####

```

エラーメッセージの全貌。translate関数の基本動作が分かっている状態で眺め、「AAStringSetというアミノ酸配列情報を入力しているからダメ」とと言われているのだろう、という対策方針を立てるべし



R Console

```

> fasta
A AAStringSet instance of length 1
  width seq                 names
[1]     4 SDGL               kadota
>
> translate(fasta)
以下にエラー (function (classes, fdef, mtable) :
  unable to find an inherited method for function 'translate' for signature '"AAStringSet"'
> hoge <- translate(fasta)
以下にエラー (function (classes, fdef, mtable) :
  unable to find an inherited method for function 'translate' for signature '"AAStringSet"'
> hoge
A AAStringSet instance of length 1
  width seq                 names
[1]     4 SDGL               kadota
> |

```

エラーなのに正しい結果が得られている?!

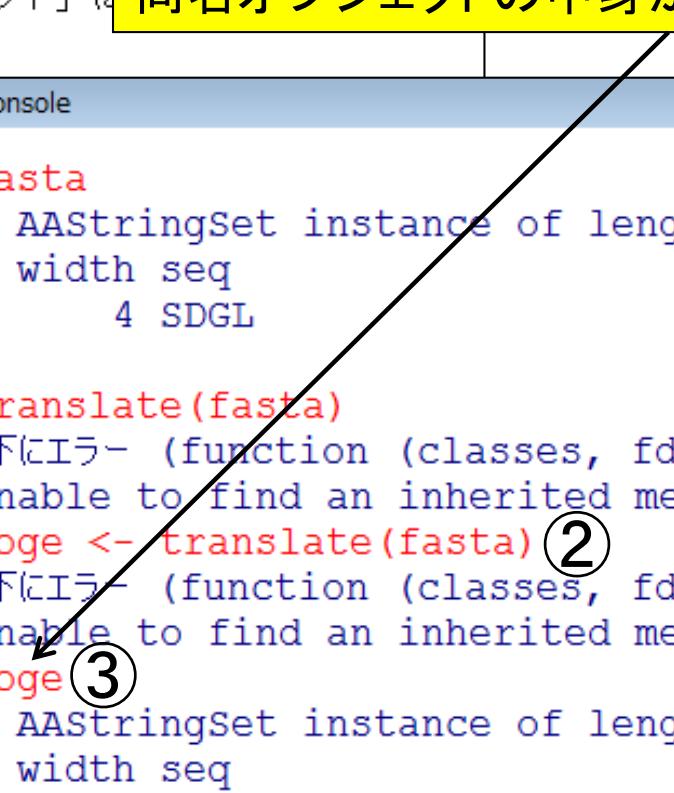
ファイル名 : rcode_translate2.txt

```
#####
### 作業ディレクトリの変更↓
#####
setwd("C:/Users/kadota/Desktop/hoge") # デスクトップ上のhogeフォルダ
#setwd("C:/Users/kadota/Desktop")      # 「デスクトップ」にした
#setwd("C:/Users/kadota/Documents")    # 「マイドキュメント」にした
#
#####
### 翻訳配列取得↓
#####
in_f <- "sample1.fasta"
out_f <- "hoge1.fasta"
#
#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, format="fasta")#in_f
#
#本番↓
hoge <- translate(fasta) ①
names(hoge) <- names(fasta)
fasta <- hoge
fasta
#
#ファイルに保存↓
writeXStringSet(fasta, file=out_f, format="fasta",

```

#入力ファイル
#出力ファイル
#パッケージの読み込み
#fastaをアミノ酸配列に変換する
#現状では翻訳が実行されない
#hogeの中身を確認してみる

③のhogeは、②の実行結果として得られたhogeではなく、①の実行結果として得られたhogeです。エラーが出ているのになぜかうまくいっている、というわけではありません。単純に以前作成した同名オブジェクトの中身が残っていただけ。



R Console

```
> fasta
A AAStringSet instance of length 1
  width seq
[1]     4 SDGL
names
kadota
>
> translate(fasta)
以下にエラー (function (classes, fdef, mtable) :
  unable to find an inherited method for function $ ...
> hoge <- translate(fasta) ②
以下にエラー (function (classes, fdef, mtable) :
  unable to find an inherited method for function $ ...
> hoge ③
A AAStringSet instance of length 1
  width seq
[1]     4 SDGL
names
kadota
> |
```

Contents

- 3-4. R Bioconductor I、2014/09/09 10:30–14:45、中級、実習
 - Tips: setwd関数を利用した効率的な作業ディレクトリの変更
 - データの型1:translate関数の入力情報(AAStringSet)
 - Tips: オブジェクトの消去
 - データの型2:翻訳配列取得のコードの中身を解説
 - バージョン情報把握とバージョンアップ
 - バージョン違いの影響
 - 過去から現在: BiostingsパッケージのreadDNAStringSet関数
 - 現在から未来: BSgenome.Hsapiens.UCSC.hg19パッケージでプロモーター配列取得
 - Bioconductor概観



オブジェクトの消去はrm関数を利用する

ファイル名 : rcode_translate2.txt

```
#####
#### 作業ディレクトリの変更↓
#####
setwd("C:/Users/kadota/Desktop/hoge") # デスクトップ上のhogeフォルダ
#setwd("C:/Users/kadota/Desktop")      # 「デスクトップ」にしたい場合↓
#setwd("C:/Users/kadota/Documents")    # 「マイドキュメント」にしたい場合↓
↓

#####
#### 翻訳配列取得↓
#####
in_f <- "sample1.fasta"
out_f <- "hogel.fasta"
↓
#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, format="fasta")#in_
↓
#本番↓
hoge <- translate(fasta)
names(hoge) <- names(fasta)
fasta <- hoge
fasta
↓
#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
```

#入力ファイル
#出力ファイル
#パッケージの読み込み
#fastaをアミノ酸配列に変換
#現状では翻訳が失敗する
#hogeの中身を確認するため

以前作成したhogeオブジェクトの中身を消去すべく、rm関数を用いてhogeオブジェクトを消去。消去後に、再びtranslate関数を実行した結果をhogelに格納しようとしてエラーが出ることを再確認。

R Console window showing the following session:

```
> rm(hoge)
> hoge
エラー: オブジェクト 'hoge' がありません
> hoge <- translate(fasta)
以下にエラー (function (classes, fdef, mtable) :
  unable to find an inherited method for function $ ...
> hoge
エラー: オブジェクト 'hoge' がありません
> ls()
[1] "fasta" "in_f"  "out_f"
> objects()
[1] "fasta" "in_f"  "out_f"
> |
```

オブジェクトの消去はrm関数を利用する

ファイル名 : rcode_translate2.txt

```
#####
### 作業ディレクトリの変更↓
#####
setwd("C:/Users/kadota/Desktop/hoge") # デスクトップ上のhogeフォルダ
#setwd("C:/Users/kadota/Desktop")      # 「デスクトップ」にした
#setwd("C:/Users/kadota/Documents")    # 「マイドキュメント」にした
#
#####
### 翻訳配列取得↓
#####
in_f <- "sample1.fasta"
out_f <- "hogel.fasta"
#
#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, format="fasta")#in_
#
#本番↓
hoge <- translate(fasta)
names(hoge) <- names(fasta)
fasta <- hoge
fasta
#
#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
```

translate関数実行結果をhogeに格納しようとしてエラーが出た場合は、本来hogeが生成されないことを確認しているだけです。ls関数やobjects関数は、現在作成されているオブジェクトをリストアップ。確かにhogelは存在しないことが分かる。

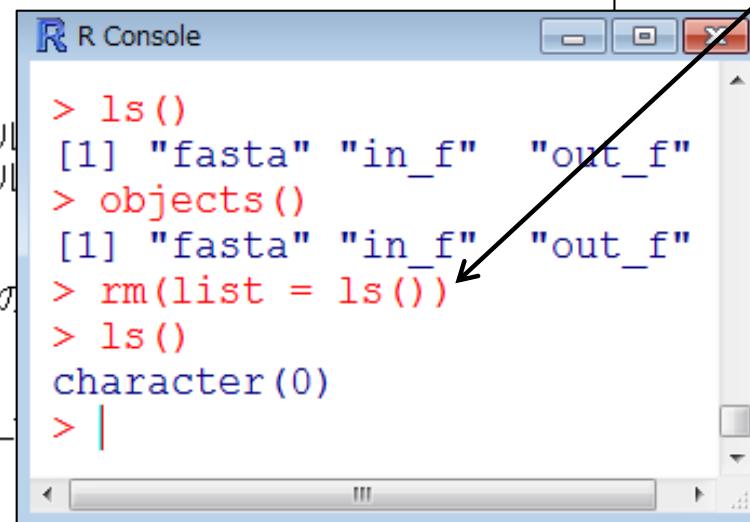
```
R Console
> rm(hoge)
> hoge
エラー: オブジェクト 'hoge' がありません
> hoge <- translate(fasta)
以下にエラー (function (classes, fdef, mtable) :
  unable to find an inherited method for function $ 
> hoge
エラー: オブジェクト 'hoge' がありません
> ls()
[1] "fasta" "in_f"  "out_f"
> objects()
[1] "fasta" "in_f"  "out_f"
>
```

オブジェクトの消去はrm関数を利用する

ファイル名 : rcode_translate2.txt

```
#####
### 作業ディレクトリの変更↓
#####
setwd("C:/Users/kadota/Desktop/hoge") # デスクトップ上のhogeフォルダ
#setwd("C:/Users/kadota/Desktop")      # 「デスクトップ」にした
#setwd("C:/Users/kadota/Documents")    # 「マイドキュメント」にした
#
#####
### 翻訳配列取得↓
#####
in_f <- "sample1.fasta"
out_f <- "hogel.fasta"
#
#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, format="fasta")#in_f
#
#本番↓
hoge <- translate(fasta)
names(hoge) <- names(fasta)
fasta <- hoge
fasta
#
#ファイルに保存↓
writeXStringSet(fasta, file=out_f, format="fasta", width=50)#fastaの中身を指定したファイル
```

現在作成されている全オブジェクトを全消去するやり方はrm(list = ls())。消去後に再びls関数を用いて利用可能なオブジェクトを表示させようしても何もないという結果が得られる。character(0)は何もないという意味です。



#fastaをアミノ酸配列に翻訳した結果をhogeに格納↓
#現状では翻訳した結果のオブジェクトhogeのdescriptionを表示↓
#hogeの中身をfastaに格納↓
#確認してるだけです↓

ファイル名 : rcode_translate3.txt

```

#####
### 作業ディレクトリの変更↓
#####
setwd("C:/Users/kadota/Desktop/hoge") # デスクトップ上のhogeフォルダにしたい場合
#setwd("C:/Users/kadota/Desktop")      # 「デスクトップ」にしたい場合↓
#setwd("C:/Users/kadota/Documents")    # 「マイドキュメント」にしたい場合↓
↓
#####
### オブジェクトの消去↓
#####
rm(list = ls())←
↓
#####
### 翻訳配列取得↓
#####
in_f <- "sample1.fasta"                #入力ファイル名を指定してin_fに格納↓
out_f <- "hogel.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="fasta")#in_fで指定したファイルの読み込み↓
↓
#本番↓
hoge <- translate(fasta)              #fastaをアミノ酸配列に翻訳した結果をhogeに
names(hoge) <- names(fasta)           #現状では翻訳した結果のオブジェクトhogeの中身をfast
fasta <- hoge                         #hogeの中身をfastaに格納↓
fasta
↓
#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
↓

```

初級者: 変なことが起こったらRを一旦終了してから再起動
 中級者: Rの再起動はやらず、オブジェクトの全消去で対応

データの型

ファイル名 : rcode_translate2.txt

```
#####
#### 作業ディレクトリの変更↓
#####
setwd("C:/Users/kadota/Desktop/hoge") # デスクトップ上のhogeフォルダにしたい場合↓
#setwd("C:/Users/kadota/Desktop")      # 「デスクトップ」にしたい場合↓
#setwd("C:/Users/kadota/Documents")    # 「マイドキュメント」にしたい場合↓
#
#####
#
```

「?translate」または「help(translate)」でも translate関数が受け付けるものの中に AAStringSetが含まれないと予想。入力として与えるfastaオブジェクトやオプションのことを引数(Arguments)といいます。

```
R Console
#####
#> 
#> > fasta
#>   A AAStringSet instance of length 1
#>     width seq           names
#>     [1] 4 SDGL          kadota
#> 
#> > translate(fasta)
#>   以下にエラー (function (classes, fdef, mtable) :
#>     unable to find an inherited method for function 'translate' for signature '"AAStringSet"'
#> > hoge <- translate(fasta)
#>   以下にエラー (function (classes, fdef, mtable) :
#>     unable to find an inherited method for function 'translate' for signature '"AAStringSet"'
#> > hoge
#>   A AAStringSet instance of length 1
#>     width seq           names
#>     [1] 4 SDGL          kadota
#> > |
```

```
> ?translate
starting httpd help server ... done
> |
```

translate {Biostrings}

Translating DNA/RNA sequences

Description

Functions for translating DNA or RNA sequences into amino acid sequences.

Usage

```
## Translating DNA/RNA:
translate(x, genetic.code=GENETIC_CODE, if.fuzzy.codon="error")

## Extracting codons without translating them:
codons(x)
```

Arguments

x A [DNAStringSet](#), [RNAStringSet](#), [DNAString](#), [RNAString](#), [MaskedDNAString](#) or [MaskedRNAString](#) object for translate.

A [DNAString](#), [RNAString](#), [MaskedDNAString](#) or [MaskedRNAString](#) object for codons.

genetic.code The genetic code to use for the translation of codons into Amino Acid letters. It must be represented as a named character vector of length 64 similar to predefined constant `GENETIC_CODE` i.e. it must contain 1-letter strings in the Amino Acid alphabet and its names must be identical to `names(GENETIC_CODE)`. The default value for `genetic.code` is `GENETIC_CODE` which represents The Standard Genetic Code. See `?AA_ALPHABET` for the Amino Acid alphabet and `?GENETIC_CODE` for The Standard Genetic Code and its known variants.

if.fuzzy.codon How fuzzy codons (i.e codon with IUPAC ambiguities) should be handled. Accepted values are:

translate関数が入力として受け付けるものはDNAStringSetやRNAStringSetなどであり、AAStringSetという記述がないことが分かります。UsageとArgumentsの記述でだいたい分かります。また、これらの記述を眺め、codons関数の存在に気づくことを通じて、幅を広げていきます。

Contents

- 3-4. R Bioconductor I、2014/09/09 10:30–14:45、中級、実習
 - Tips : setwd関数を利用した効率的な作業ディレクトリの変更
 - データの型1 : translate関数の入力情報(AAStringSet)
 - Tips : オブジェクトの消去
 - データの型2 : 翻訳配列取得のコードの中身を解説
 - バージョン情報把握とバージョンアップ
 - バージョン違いの影響
 - 過去から現在 : BiostingsパッケージのreadDNAStringSet関数
 - 現在から未来 : BSgenome.Hsapiens.UCSC.hg19パッケージでプロモーター配列取得
 - Bioconductor概観



イントロ | 一般 | 翻訳配列(translate)を取得 NEW

塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。

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

1. FASTA形式ファイル(sample1.fasta)の場合:

```
in_f <- "sample1.fasta"          #入力ファイル名を指定してin_fに格納  
out_f <- "hogel.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="fasta") #in_fで指定したファイルの読み込み  
  
#本番  
hoge <- translate(fasta)        #fastaをアミノ酸配列に翻訳した結果をhogeに格納  
names(hoge) <- names(fasta)      #現状では翻訳した結果のオブジェクトhogeのdescriptor  
fastal <- hoge                  #hogeの中身をfastalに格納  
#確認してみるだけです  
  
#ファイルに保存  
writeXStringSet(fasta, file=out_f, format="fasta")
```

・ イントロ | 一般 | 翻訳配列(translate)を取得

>kadota
AGTGACGGTCTT

in_fで指定した入力ファイルをreadDNAStringSet関数を用いて読み込んだ後のfastaオブジェクトは、translate関数が入力として受け付けるDNAStringSet形式であることが分かる

R Console

```
> in_f <- "sample1.fasta"          #入力$  
> out_f <- "hogel.fasta"          #出力$  
>  
> #必要なパッケージをロード  
> library(Biostrings)              #パッ$  
>  
> #入力ファイルの読み込み  
> fasta <- readDNAStringSet(in_f, format="fast$  
> fasta  
A DNAStringSet instance of length 1  
width seq                                names  
[1] 12 AGTGACGGTCTT                      kadota  
> |
```

イントロ | 一般 | 翻訳配列(translate)を取得 NEW

・イントロ | 一般 | 翻訳配列(translate)を取得

塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。

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

1. FASTA形式ファイル([sample1.fasta](#))の場合:

```
in_f <- "sample1.fasta"          #入力ファイル名を指定してin_fに格納  
out_f <- "hogel.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="fasta") #in_fで指定したファイルの読み込み  
  
#本番  
hoge <- translate(fasta)         #fastaをアミノ酸配列に翻訳した結果をhogeに格納  
names(hoge) <- names(fasta)       #現状では翻訳した結果のオブジェクトhogeのdescription  
fastal <- hoge                   #hogeの中身をfastalに格納  
#確認してるだけです  
  
#ファイルに保存  
writeXStringSet(fasta, file=out_f, format="fasta")
```

>kadota
AGTGACGGTCTT

入力ファイルのdescription行の記述がnamesという場所、配列長がwidthという場所に格納されていることもわかる。

R Console

```
> in_f <- "sample1.fasta"          #入力$  
> out_f <- "hogel.fasta"          #出力$  
>  
> #必要なパッケージをロード      #パッ$  
> library(Biostrings)  
>  
> #入力ファイルの読み込み  
> fasta <- readDNAStringSet(in_f, format="fast$  
> fasta  
A DNAStringSet instance of length 1  
  width seq  
[1]    12 AGTGACGGTCTT  
  names  
     kadota  
> |
```

イントロ | 一般 | 翻訳配列(translate)を取得 NEW

塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。

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

1. FASTA形式ファイル(sample1.fasta)の場合:

```
in_f <- "sample1.fasta"          #入力ファイル名を指定してin_fに格納
out_f <- "hogel.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="fasta") #in_fで指定したファイルの読み込み

#本番
hoge <- translate(fasta)         #fastaをアミノ酸配列に翻訳した結果をhogeに格納
names(hoge) <- names(fasta)       #現状では翻訳した結果のオブジェクトhogeのdescriptionをnamesに格納
fastaa <- hoge                   #hogeの中身をfastaaに格納
#確認しているだけです

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

・ イントロ | 一般 | 翻訳配列(translate)を取得

>kadota
AGTGACGGTCTT

入力ファイルのdescription行の記述がnamesという関数、配列長がwidthという関数で取り出せることが分かる。ただしseq関数は、存在はするがイメージと異なり塩基配列は返さない…。

R Console

```
> fasta
A DNAStringSet instance of length 1
  width seq
[1] 12 AGTGACGGTCTT
> names(fasta)
[1] "kadota"
> width(fasta)
[1] 12
> as.character(fasta)
  kadota
  "AGTGACGGTCTT"
> seq(fasta)
[1] 1
> |
```

イントロ | 一般 | 翻訳配列(translate)を取得 NEW

・ イントロ | 一般 | 翻訳配列(translate)を取得

塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。

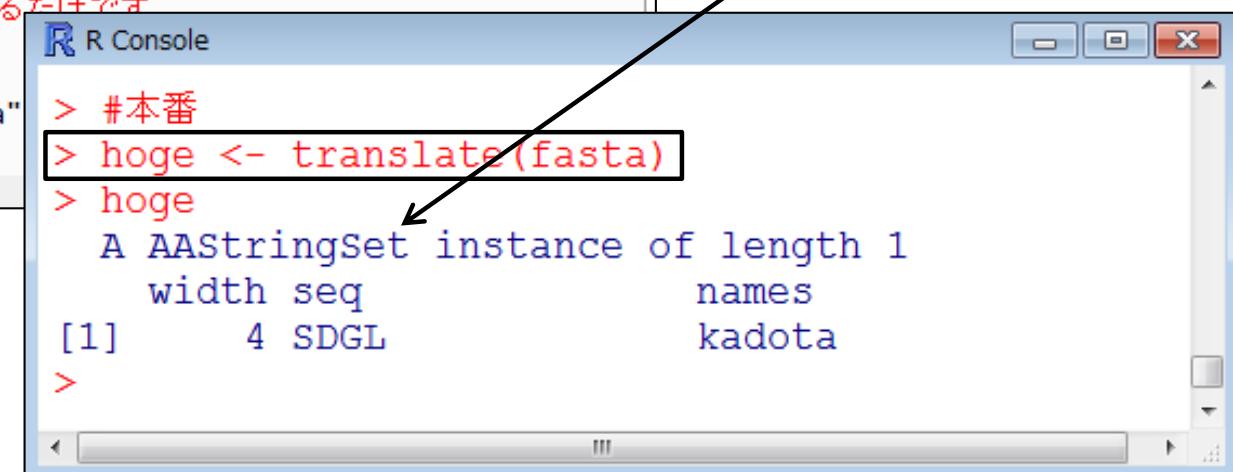
「ファイル」→「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

1. FASTA形式ファイル([sample1.fasta](#))の場合:

```
in_f <- "sample1.fasta"          #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="fasta") #in_fで指定したファイルの読み込み  
  
#本番  
hoge <- translate(fasta)        #fastaをアミノ酸配列に翻訳した結果をhogeに格納  
names(hoge) <- names(fasta)      #現状では翻訳した結果のオブジェクトhogeのdescriptor  
fasta <- hoge                   #hogeの中身をfastaに格納  
fasta  
  
#ファイルに保存  
writeXStringSet(fasta, file=out_f, format="fasta")
```

>kadota
AGTGACGGTCTT

translate関数実行後のhogeオブジェクトは、AAStringSet形式に変わっていることがわかる



R Console

```
> #本番  
> hoge <- translate(fasta)  
> hoge  
A AAStringSet instance of length 1  
  width seq  
  [1] 4 SDGL  
  names  
  [1] kadota  
>
```

イントロ | 一般 | 翻訳配列(translate)を取得 NEW

塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。

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

1. FASTA形式ファイル(sample1.fasta)の場合:

```
in_f <- "sample1.fasta"          #入力ファイル名を指定してin_fに格納
out_f <- "hogel.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="fasta") #in_fで指定したファイルの読み込み

#本番
hoge <- translate(fasta)         #fastaをアミノ酸配列に翻訳した結果をhogeに格納
names(hoge) <- names(fasta)       #現状では翻訳した結果のオブジェクトhogeのdescriptor
fasta <- hoge                     #hogeの中身をfastaに格納
fasta                                #確認しているだけです

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

・ イントロ | 一般 | 翻訳配列(translate)を取得

>kadota
AGTGACGGTCTT

少なくともR ver. 2.15.2までは必要だったが、講義資料作成時のR環境(R ver. 3.1.0とBioconductor ver. 2.14)では、この作業はもはや必要ない。

R Console

```
> #本番
> hoge <- translate(fasta)
> hoge
A AAStringSet instance of length 1
  width seq
[1] 4 SDGL
>
> names(hoge) <- names(fasta)      #現状$names$hoge
> fasta <- hoge                   #hoge$names
> fasta                            #確認$hoge$names
```

A AAStringSet instance of length 1
 width seq
[1] 4 SDGL
>

イントロ | 一般 | 翻訳配列(translate)を取得 NEW

・イントロ | 一般 | 翻訳配列(translate)を取得

塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。

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

1. FASTA形式ファイル(sample1.fasta)の場合:

```
in_f <- "sample1.fasta"          #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="fasta") #in_fで指定したファイルの読み込み  
  
#本番  
hoge <- translate(fasta)        #fastaをアミノ酸配列に翻訳した結果をhogeに格納  
names(hoge) <- names(fasta)      #現状では翻訳した結果のオブジェクトhogeのdescriptor  
fastal <- hoge                  #hogeの中身をfastalに格納  
fastal  
  
#ファイルに保存  
writeXStringSet(fasta, file=out_f)
```

```
>kadota  
AGTGACGGTCTT
```

NGS速習コース終了後に関連個所をこのように修正するか、バージョンの違いの影響を説明する資料として残すかは考え中

6. FASTA形式ファイル(sample1.fasta)の場合:

1.と基本的に同じ結果が得られます BUT AAStringSetオブジェクトでdescription行が消えてしまうバグが修正されたようなので、すっきりさせたものです。

```
in_f <- "sample1.fasta"          #入力ファイル名を指定してin_fに格納  
out_f <- "hoge6.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="fasta") #in_fで指定したファイルの読み込み  
  
#本番  
fastal <- translate(fasta)       #fastaをアミノ酸配列に翻訳した結果をfastalに格納  
fastal  
  
#ファイルに保存  
writeXStringSet(fasta, file=out_f, format="fasta", width=50) #fastalの中身を指定したファ
```

イントロ | 一般 | 翻訳配列(translate)を取得 NEW

・ イントロ | 一般 | 翻訳配列(translate)を取得

塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。

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

1. FASTA形式ファイル([sample1.fasta](#))の場合:

```
in_f <- "sample1.fasta"          #入力ファイル名を指定してin_fに格納  
out_f <- "hogel.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="fasta") #in_fで指定したファイルの読み込み  
  
#本番  
hoge <- translate(fasta)        #fastaをアミノ酸配列に翻訳した結果をhogeに格納  
names(hoge) <- names(fasta)      #現状では翻訳した結果のオブジェクトhogeのdescriptor  
fasta <- hoge                   #hogeの中身をfastaに格納  
hoge  
  
#ファイルに保存  
writeXStringSet(fasta, file=out_f, format="fasta")
```

>kadota
AGTGACGGTCTT

講義資料作成時のR環境(R ver. 3.1.0とBioconductor ver. 2.14)では、この作業はもはや必要ないと2014/08/13に気づいたが、一連の手順作成当時はこうする必要があったのです。自分のR環境を把握しバージョンの違いに気をつけるべし!

R Console

```
> #本番  
> hoge <- translate(fasta)  
> hoge  
A AAStringSet instance of length 1  
  width seq  
[1] 4 SDGL  
  names  
    kadota  
>  
> names(hoge) <- names(fasta) #現状$  
> fasta <- hoge #hoge$  
> fasta #確認$  
A AAStringSet instance of length 1  
  width seq  
[1] 4 SDGL  
  names  
    kadota  
> |
```

Contents

- 3-4. R Bioconductor I、2014/09/09 10:30–14:45、中級、実習
 - Tips : setwd関数を利用した効率的な作業ディレクトリの変更
 - データの型1 : translate関数の入力情報(AAStringSet)
 - Tips : オブジェクトの消去
 - データの型2 : 翻訳配列取得のコードの中身を解説
 - バージョン情報把握とバージョンアップ
 - バージョン違いの影響
 - 過去から現在 : BiostingsパッケージのreadDNAStringSet関数
 - 現在から未来 : BSgenome.Hsapiens.UCSC.hg19パッケージでプロモーター配列取得
 - Bioconductor概観



sessionInfo()で自分のR環境を把握

```
R R Console
> sessionInfo()
R version 3.1.0 (2014-04-10)
Platform: x86_64-w64-mingw32/x64 (64-bit)

locale:
[1] LC_COLLATE=Japanese_Japan.932
[2] LC_CTYPE=Japanese_Japan.932
[3] LC_MONETARY=Japanese_Japan.932
[4] LC_NUMERIC=C
[5] LC_TIME=Japanese_Japan.932

attached base packages:
[1] parallel stats      graphics grDevices utils
[6] datasets methods    base

other attached packages:
[1] Biostrings_2.32.0   XVector_0.4.0
[3] IRanges_1.22.3     BiocGenerics_0.10.0

loaded via a namespace (and not attached):
[1] stats4_3.1.0       zlibbioc_1.10.0
> |
```

門田の解析環境は、R ver. 3.1.0であることがわかる。また、Biostringsパッケージのバージョンは2.32.0と読み取る。

投稿論文へのバージョン情報記載例

BMC Bioinformatics. 2013 Jul 9;14:219. doi: 10.1186/1471-2105-14-219.

TCC: an R package for comparing tag count data with robust normalization strategies.

Sun J¹, Nishiyama T, Shimizu K, Kadota K.

Author information

Abstract

BACKGROUND: Differential expression analysis based on "next-generation" sequencing technologies is a fundamental means of studying RNA expression. We recently developed a multi-step normalization method (called TbT) for two-group RNA-seq data with replicates and demonstrated that the statistical methods available in four R packages (*edgeR*, *DESeq*, *baySeq*, and *NBPSeq*) together with TbT can produce a well-ranked gene list in which true differentially expressed genes (DEGs) are top-ranked and non-DEGs are bottom ranked. However, the advantages of the current TbT method come at the cost of a huge computation time. R packages did not have normalization methods based on such a multi-step strategy.

RESULTS: TCC (an acronym for Tag Count Comparison) is an R package that provides functions for differential expression analysis of tag count data. The package includes normalization methods, whose strategy is to remove potential DEGs before performing normalization. The normalization function based on this DEG elimination strategy (i) the original TbT method based on DEGES for two-group data with or without replicates, faster methods for two-group data with or without replicates, and (iii) methods for comparison. TCC provides a simple unified interface to perform such analyses with the functions provided by *edgeR*, *DESeq*, and *baySeq*. Additionally, a function for generating data under various conditions and alternative DEGES procedures consisting of functions existing packages are provided. Bioinformatics scientists can use TCC to evaluate what is done in other R packages and biologists familiar with other R packages can easily learn what is done in TCC.

CONCLUSION: DEGES in TCC is essential for accurate normalization of tag count data when up- and down-regulated DEGs in one of the samples are extremely biased. TCC is useful for analyzing tag count data in various scenarios ranging from unbiased to biased differential expression. TCC is available at <http://www.iu.a.u-tokyo.ac.jp/~kadota/TCC/> and will appear in Bioconductor (<http://bioconductor.org/>) from ver. 2.13.

R ver. 3.1.0の場合は、ここを
ver. 3.1.0に変更すればよい

the functions implemented in *edgeR*, *DESeq*, and *baySeq*. Here, we demonstrate that the DEGES-based normalization methods are more effective than the methods implemented in the other packages. All analyses were performed using R (ver. 2.15.2) and Bioconductor [20]. Execution times were measured on a Linux system (CentOS release 6.2 (Final), Intel® Xeon® E5-4617 (2.9 GHz) 24 CPU, and 512 GB memory). The versions of major R libraries were TCC ver. 1.1.99, *edgeR* ver. 3.0.4, *DESeq* ver. 1.10.1, and *baySeq* ver. 1.12.0.

投稿論文へのバージョン情報記載例

BMC Bioinformatics. 2013 Jul 9;14:219. doi: 10.1186/1471-2105-14-219.

TCC: an R package for comparing tag count data with robust normalization strategies.

Sun J¹, Nishiyama T, Shimizu K, Kadota K.

Author information

Abstract

BACKGROUND: Differential expression analysis based on "next-generation" sequencing technologies is a fundamental means of studying RNA expression. We recently developed a multi-step normalization method (called TbT) for two-group RNA-seq data with replicates and demonstrated that the statistical methods available in four R packages (*edgeR*, *DESeq*, *baySeq*, and *NBPSeq*) together with TbT can produce a well-ranked gene list in which true differentially expressed genes (DEGs) are top-ranked and non-DEGs are bottom ranked. However, the advantages of the current TbT method come at the cost of a huge computation time. R packages did not have normalization methods based on such a multi-step strategy.

RESULTS: TCC (an acronym for Tag Count Comparison) is an R package that provides functions for differential expression analysis of tag count data. The package includes normalization methods, whose strategy is to remove potential DEGs before performing normalization. The normalization function based on this DEG elimination strategy (i) the original TbT method based on DEGES for two-group data with or without replicates, faster methods for two-group data with or without replicates, and (iii) methods for comparison. TCC provides a simple unified interface to perform such analyses with the functions provided by *edgeR*, *DESeq*, and *baySeq*. Additionally, a function for generating data under various conditions and alternative DEGES procedures consisting of functions from existing packages are provided. Bioinformatics scientists can use TCC to evaluate what is done in other R packages and biologists familiar with other R packages can easily learn what is done in TCC.

CONCLUSION: DEGES in TCC is essential for accurate normalization of tag count data when up- and down-regulated DEGs in one of the samples are extremely biased. TCC is useful for analyzing tag count data in various scenarios ranging from unbiased to biased differential expression. TCC is available at <http://www.iu.a.u-tokyo.ac.jp/~kadota/TCC/> and will appear in Bioconductor (<http://bioconductor.org/>) from ver. 2.13.

「R and Bioconductor」や「R/Bioconductor」など、2大リポジトリの一つである Bioconductorがセットで記述される場合が多いのは、TCCやBiostringsパッケージを含むバイオインフォ系パッケージの多くが Bioconductorから配布されているから。

the functions implemented in *edgeR*, *DESeq*, and *baySeq*. Here, we demonstrate that the DEGES-based normalization methods are more effective than the methods implemented in the other packages. All analyses were performed using R (ver. 2.15.2) and Bioconductor [20]. Execution times were measured on a Linux system (CentOS release 6.2 (Final), Intel® Xeon® E5-4617 (2.9 GHz) 24 CPU, and 512 GB memory). The versions of major R libraries were TCC ver. 1.1.99, *edgeR* ver. 3.0.4, *DESeq* ver. 1.10.1, and *baySeq* ver. 1.12.0.

定期的なバージョンアップをお勧め

■ R本体もBioconductorも定期的にバージョンアップがなされている

- R (<http://www.r-project.org/>)
 - 2014-07-10にver. 3.1.1をリリース
 - 2014-04-10にver. 3.1.0をリリース
 - 2014-03-06にver. 3.0.3をリリース
 - ...
 - 2012-03-30にver. 2.15.0をリリース
 - ...

2014年8月14日現在のR本体の最新バージョンは3.1.1、Biostrings最新バージョンは2.32.1。しかし、R ver. 2.15.0など本体のバージョンが古いと、2014年8月14日にBiostringsパッケージを個別にインストールしてもver. 2.26.3と昔のバージョンがインストールされる点に注意が必要です。

- Bioconductor (<http://bioconductor.org/>)は半年ごとにリリース

- 2014-04にver. 2.14をリリース (R ver. 3.1.0で動作確認)、提供パッケージ数: 824
 - 2013-10にver. 2.13をリリース (R ver. 3.0で動作確認)、提供パッケージ数: 750
 - 2013-04にver. 2.12をリリース (R ver. 3.0で動作確認)、提供パッケージ数: 672
 - 2012-10にver. 2.11をリリース (R ver. 2.15.1で動作確認)、提供パッケージ数: 608
 - 2012-04にver. 2.10をリリース (R ver. 2.15.0で動作確認)、提供パッケージ数: 553
 - 2011-11にver. 2.9をリリース (R ver. 2.14.0で動作確認)、提供パッケージ数: 517
 - 2011-04にver. 2.8をリリース (R ver. 2.13.0で動作確認)、提供パッケージ数: 466



What's new?

- 門田幸二著シリーズ Useful R 第7巻トランスクリプトーム解析刊行(共立出版や、ROKU法(Kadota et al., 2006)、WAD法(Kadota et al., 2008)などについてトランスクリプトーム解析部分のRコード)については、このページの「書籍|トランスクリプトーム」
- お知らせは主に[\(Rで\)塩基配列解析](#)で行っておりますのでそちらをご覧ください。また、[\(Rで\)塩基配列解析](#)中の参考資料(講義、講習会、本など)から辿れます。

- [はじめに](#) (last modified 2014/05/14)
- [過去のお知らせ](#) (last modified 2014/03/03)
- [Rのインストールと起動](#) (last modified 2014/05/14)
- [Rの昔のバージョンのインストール](#) (last modified 2012/04/07)
- [使用例\(初心者向け\)](#) (last modified 2011/09/15)
- [サンプルデータ](#) (last modified 2014/05/02)
- ...

Rの昔のバージョンのインストール

DFW (Chen et al., Bioinformatics, 2007)というAffymetrixの"階層的クラスタリング"結果を導くような遺伝子データ用の正規化法(嫌味ではなくいい方法なんですが、R2.7.2あたりだと正常に動作していましたが、よく動いてくれません。そのような場合でも、Rの昔のバージョンをインストールするやり方をR2.7.2の

1. [ここ](#)をクリックして、任意のRのバージョンの名前をクリック
2. R-X.Y.Z-win32.exe(例えば[R-2.7.2-win32.exe](#))がままに押す

Previous Releases of R for Windows

This directory contains previous binary releases of R to run on Windows 95, 98, ME, NT4.0, 2000 and XP or later on Intel/clone chips.

The current release, and links to development snapshots, are available [here](#). Source code for these releases and others is available through [the main CRAN page](#).

In this directory:

- [R 3.1.0](#) (April, 2014)
- [R 3.0.3](#) (March, 2014)
- [R 3.0.2](#) (September, 2013)
- [R 3.0.1](#) (May, 2013)
- [R 3.0.0](#) (April, 2013)
- [R 2.15.3](#) (March, 2013)
- [R 2.15.2](#) (October, 2012)
- [R 2.15.1](#) (June, 2012)
- [R 2.15.0](#) (March, 2012)
- [R 2.14.2](#) (February, 2012)
- [R 2.14.1](#) (December, 2011)
- [R 2.14.0](#) (November, 2011)
- [R 2.13.2](#) (September, 2011)
- [R 2.13.1](#) (July, 2011)
- [R 2.13.0](#) (April, 2011)
- [R 2.12.2](#) (February, 2011)
- [R 2.12.1](#) (December, 2010)
- [R 2.12.0](#) (October, 2010)
- [R 2.11.1](#) (May, 2010)
- [R 2.11.0](#) (April, 2010)
- [R 2.10.1](#) (December, 2009)
- [R 2.10.0](#) (October, 2009)

Rの昔のバージョンのインストール手順。Windows版のver. 2.15.0の場合。

```
R version 2.15.0 (2012-03-30)
Copyright (C) 2012 The R Foundation for Statistical Co
ISBN 3-900051-07-0
Platform: x86_64-pc-mingw32/x64 (64-bit)
```

Rは、自由なソフトウェアであり、「完全に無保証」です。

一定の条件に従えば、自由にこれを再配布することができます

配布条件の詳細に関しては、「`license()`」あるいは「`licence()`」

Rは多くの貢献者による共同プロジェクトです。
 詳しくは「`contributors()`」と入力してください。
 また、RやRのパッケージを出版物で引用する際の形式
 「`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
 「`q()`」と入力すればRを終了します。

R ver. 2.15.0上で、Biostringsパッケージのみを通常手順でインストール。
 Bioconductor ver. 2.11でリリースされたバージョンのBiostringsがインストールされていることが分かる。

```
R Console
> source("http://bioconductor.org/biocLite.R")
A new version of Bioconductor is available after ins$ recent version of R; see http://bioconductor.org/i$ install.packages("BiocInstaller", repos = a["BioCso$ 'lib' = "C:/Program Files/R/R-2.15.0/library" is no$ URL 'http://www.bioconductor.org/packages/2.11/bioc$ Content type 'application/zip' length 43242 bytes (4$ 開かれた URL
downloaded 42 Kb

package 'BiocInstaller' successfully unpacked and $

The downloaded binary packages are in
      C:\Users\kadota\AppData\Local\Temp\RtmpY50wu$ Bioconductor version 2.11 (BiocInstaller 1.8.3), ?bi$ > biocLite("Biostrings")
BioC_mirror: http://bioconductor.org
Using Bioconductor version 2.11 (BiocInstaller 1.8.3$ Installing package(s) 'Biostrings'
警告: unable to access index for repository http:$ also installing the dependencies 'BiocGenerics', $
```

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

次のパッケージを付け加えます: ''BiocGenerics''

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

  xtabs

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

  anyDuplicated, cbind, colnames, duplicated, eval,
  get, intersect, lapply, Map, mapply, mget, order, pmax.int, pmin, pmin.int, Position, rbind, Reduce, rownames, sapply, setdiff, table, tapply, union, union.int

要求されたパッケージ IRanges をロード中です
警告メッセージ:
1: パッケージ ''Biostrings'' はバージョン 2.15.2 の R の下で造られました
2: パッケージ ''BiocGenerics'' はバージョン 2.15.1 の R の下で造られました
3: パッケージ ''IRanges'' はバージョン 2.15.2 の R の下で造られました
```

R ver. 2.15.0上で、Biostringsパッケージのみを通常手順でインストール。無事インストールできたようなので、library関数を用いてBiostringsパッケージを読み込んでいるところ。警告メッセージは「あなたの環境はR ver. 2.15.0だけど、インストールしたBiostringsパッケージはR ver. 2.15.2環境下で作られたものです」と言っています。library(Biostrings)でパッケージのロードに失敗する場合には本気で対処する必要がありますが、警告メッセージが出ているだけですので、私は特に気にしていません。

R R Console

```
> sessionInfo()
R version 2.15.0 (2012-03-30)
Platform: x86_64-pc-mingw32/x64 (64-bit)

locale:
[1] LC_COLLATE=Japanese_Japan.932
[2] LC_CTYPE=Japanese_Japan.932
[3] LC_MONETARY=Japanese_Japan.932
[4] LC_NUMERIC=C
[5] LC_TIME=Japanese_Japan.932

attached base packages:
[1] stats      graphics   grDevices utils      datasets 
[6] methods    base

other attached packages:
[1] Biostrings_2.26.3    IRanges_1.16.6
[3] BiocGenerics_0.4.0   BiocInstaller_1.8.3

loaded via a namespace (and not attached):
[1] parallel_2.15.0 stats4_2.15.0   tools_2.15.0
> date()
[1] "Thu Aug 14 14:38:54 2014"
> |
```

- [Rの昔のバージョンのインストール](#)

2014年8月14日現在のR本体の最新バージョンは3.1.1、Biostrings最新バージョンは2.32.1。しかし、R ver. 2.15.0など本体のバージョンが古いと、2014年8月14日にBiostringsパッケージを個別にインストールしてもver. 2.26.3と昔のバージョンがインストールされる点に注意が必要です。

定期的なバージョンアップをお勧め

■ R本体もBioconductorも定期的にバージョンアップがなされている

- R (<http://www.r-project.org/>)
 - 2014-07-10にver. 3.1.1をリリース
 - 2014-04-10にver. 3.1.0をリリース
 - 2014-03-06にver. 3.0.3をリリース
 - ...
 - 2012-03-30にver. 2.15.0をリリース
 - ...

基本的にはバグの修正や新たな機能が
どんどん追加されているので最新版の利
用をお勧め。毎年5月初めと11月初めごろ
にBioconductorを覗きにいくとよいだろう。

- Bioconductor (<http://bioconductor.org/>)は半年ごとにリリース

- 2014-04にver. 2.14をリリース (R ver. 3.1.0で動作確認)、提供パッケージ数 : 824
 - 2013-10にver. 2.13をリリース (R ver. 3.0で動作確認)、提供パッケージ数 : 750
 - 2013-04にver. 2.12をリリース (R ver. 3.0で動作確認)、提供パッケージ数 : 672
 - 2012-10にver. 2.11をリリース (R ver. 2.15.1で動作確認)、提供パッケージ数 : 608
 - 2012-04にver. 2.10をリリース (R ver. 2.15.0で動作確認)、提供パッケージ数 : 553
 - 2011-11にver. 2.9をリリース (R ver. 2.14.0で動作確認)、提供パッケージ数 : 517
 - 2011-04にver. 2.8をリリース (R ver. 2.13.0で動作確認)、提供パッケージ数 : 466



バージョンアップの意義：具体例

BMC Bioinformatics. 2013 Jul 9;14:219. doi: 10.1186/1471-2105-14-219.

TCC: an R package for comparing tag count data with robust normalization strategies.

Sun J¹, Nishiyama T, Shimizu K, Kadota K.

Author information

Abstract

BACKGROUND: Differential expression analysis based on "next-generation" sequencing technologies is a fundamental means of studying RNA expression. We recently developed a multi-step normalization method (called TbT) for two-group RNA-seq data with replicates and demonstrated that the statistical methods available in four R packages (edgeR, DESeq, baySeq, and NBPSeq) together with TbT can produce a well-ranked gene list in which true differentially expressed genes (DEGs) are top-ranked and non-DEGs are bottom ranked. However, the advantages of the current TbT method come at the cost of a huge computation time. Moreover, the R packages did not have normalization methods based on such a multi-step strategy.

RESULTS: TCC (an acronym for Tag Count Comparison) is an R package that provides a series of functions for differential expression analysis of tag count data. The package incorporates multi-step normalization methods, whose strategy is to remove potential DEGs before performing the data normalization. The normalization function based on this DEG elimination strategy (DEGES) includes (i) the original TbT method based on DEGES for two-group data with or without replicates, (ii) much faster methods for two-group data with or without replicates, and (iii) methods for multi-group comparison. TCC provides a simple unified interface to perform such analyses with combinations of functions provided by edgeR, DESeq, and baySeq. Additionally, a function for generating simulation data under various conditions and alternative DEGES procedures consisting of functions in the existing packages are provided. Bioinformatics scientists can use TCC to evaluate their methods, and biologists familiar with other R packages can easily learn what is done in TCC.

CONCLUSION: DEGES in TCC is essential for accurate normalization of tag count data, especially when up- and down-regulated DEGs in one of the samples are extremely biased in their number. TCC is useful for analyzing tag count data in various scenarios ranging from unbiased to extremely biased differential expression. TCC is available at <http://www.iu.a.u-tokyo.ac.jp/~kadota/TCC/> and will appear in Bioconductor (<http://bioconductor.org/>) from ver. 2.13.

2013年7月の論文publish以降も継続的にアップデートしています：

- 多群間比較やpaired dataへの対応など、解析可能な実験デザインを拡張
- DESeq2対応もほぼ完了
- サンプル間クラスタリング用関数やマイクロアレイデータ用組織特異的発現パターン検出法ROKUの実装
- ドキュメントが充実(TCC ver. 1.4.0で74ページに!)

- [解析 | 発現変動 | について \(last modified 2014/07/10\) NEW](#)
- [解析 | 発現変動 | 2群間 | 対応なし | について \(last modified 2014/07/10\) NEW](#)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC \(Sun 2013\) \(last modified 2014/07/10\)](#)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | DESeq \(Anders 2010\) \(last modified 2014/07/10\)](#)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | edgeR \(Robinson 2010\) \(last modified 2014/07/10\)](#)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | GPSeq \(Srivastava 2010\) \(last modified 2014/07/10\)](#)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | baySeq \(Hardcastle and Kelly, BMC Bioinformatics 2010\) \(last modified 2014/07/10\)](#)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | DESeq \(Anders and Huber, Genome Biology 2010\) \(last modified 2014/07/10\)](#)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | DESeq2 \(Anders and Huber, Genome Biology 2012\) \(last modified 2014/07/10\)](#)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | NBPSeq \(Di et al., SAGMB, 2011\) \(last modified 2014/07/10\)](#)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | BBSeq \(Zhou et al., Bioinformatics, 2012\) \(last modified 2014/07/10\)](#)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | NOISeq \(Tarazona et al., Genome Research, 2012\) \(last modified 2014/07/10\)](#)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | PoissonSeq \(Li et al., Biostatistics, 2012\) \(last modified 2014/07/10\)](#)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | SAMseq \(Li and Tibshirani, Stat Methods in Med Res, 2012\) \(last modified 2014/07/10\)](#)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | easyRNASeq \(Delhomme et al., Bioinformatics, 2012\) \(last modified 2014/07/10\)](#)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | DSGseq \(Wang et al., Gene, 2013\) \(last modified 2014/07/10\)](#)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | sSeq \(Yu et al., Bioinformatics, 2013\) \(last modified 2014/07/10\)](#)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | TCC \(Sun et al., BMC Bioinformatics, 2013\) \(last modified 2014/07/10\)](#)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | tweeDEseq \(Esnaola et al., BMC Bioinformatics, 2013\) \(last modified 2014/07/10\)](#)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | NPEBseq \(Bi et al., BMC Bioinformatics, 2013\) \(last modified 2014/07/10\)](#)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | DER Finder \(Frazee et al., Biostatistic, 2013\) \(last modified 2014/07/10\)](#)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | Characteristic Direction\(CD\) \(Clark et al., Bioinformatics, 2013\) \(last modified 2014/07/10\)](#)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | edgeR-robust \(Zhou et al., Nucleic Acids Research, 2013\) \(last modified 2014/07/10\)](#)
- [解析 | 発現変動 | 2群間 | 対応なし | 複製あり | ShrinkBayes \(Van De Wiel et al., BMC Bioinformatics, 2013\) \(last modified 2014/07/10\)](#)

解析 | 発現変動 | 2群間 | 対応なし | について NEW

実験データ用:

Aさん
Bさん
Cさん
Dさん
Eさん
Fさん
Gさん

2014年

R用:

- [DEGSeq: Wang et al., Bioinformatics, 2010](#)
- [edgeR: Robinson et al., Bioinformatics, 2010](#)
- [GPSeq: Srivastava et al., Nucleic Acid Research, 2010](#)
- [baySeq: Hardcastle and Kelly, BMC Bioinformatics, 2010](#)
- [DESeq: Anders and Huber, Genome Biology, 2010](#)
- [DESeq2: Anders and Huber, Genome Biology, 2012](#)
- [NBPSeq: Di et al., SAGMB, 2011](#)
- [BBSeq: Zhou et al., Bioinformatics, 2012](#)
- [NOISeq: Tarazona et al., Genome Research, 2012](#)
- [PoissonSeq: Li et al., Biostatistics, 2012](#)
- [SAMseq: Li and Tibshirani, Stat Methods in Med Res, 2012](#)
- [easyRNASeq: Delhomme et al., Bioinformatics, 2012](#)
- [DSGseq: Wang et al., Gene, 2013](#)
- [sSeq: Yu et al., Bioinformatics, 2013](#)
- [TCC: Sun et al., BMC Bioinformatics, 2013](#)
- [tweeDEseq: Esnaola et al., BMC Bioinformatics, 2013](#)
- [NPEBseq: Bi et al., BMC Bioinformatics, 2013](#)
- [DER Finder: Frazee et al., Biostatistic, 2013](#)
- [Characteristic Direction\(CD\): Clark et al., Bioinformatics, 2013](#)
- [edgeR-robust: Zhou et al., Nucleic Acids Research, 2013](#)
- [ShrinkBayes: Van De Wiel et al., BMC Bioinformatics, 2013](#)



Home

Install

Hel

[Home](#) » [Bioconductor 2.14](#) » [Software Packages](#) » TCC

TCC

TCC: Differential expression analysis for tag count data with robust normalization strategies

Bioconductor version: Release (2.14)

This package provides a series of functions for performing differential expression analysis from RNA-seq count data using robust normalization strategy (called DEGES). The basic idea of DEGES is that potential differentially expressed genes or transcripts (DEGs) among compared samples should be removed before data normalization to obtain a well-ranked gene list where true DEGs are top-ranked and non-DEGs are bottom ranked. This can be done by performing a multi-step normalization strategy (called DEGES for DEG elimination strategy). A major characteristic of TCC is to provide the robust normalization methods for several kinds of count data (two-group with or without replicates, multi-group/multi-factor, and so on) by virtue of the use of combinations of functions in depended packages.

Author: Jianqiang Sun, Tomoaki Nishiyama, Kentaro Shimizu, and Koji Kadota

Maintainer: Jianqiang Sun <wukong@bi.a.u-tokyo.ac.jp>, Tomoaki Nishiyama <tomoakin@staff.kanazawa-u.ac.jp>

Citation (from within R, enter `citation("TCC")`):

TCC

TCC: Differential expression normalization strategies

Bioconductor version: Release (2.14)

This package provides a set of count data using robust normalization strategies to obtain a well ranked list of differentially expressed genes. It also provides a DEG elimination strategy. A major advantage of this package is that it can handle several kinds of count data (e.g., RNA-seq, microarray) by virtue of the use of combinatorial designs.

Author: Jianqiang Sun, Tomoaki Kondo

Maintainer: Jianqiang Sun <wukun@staff.kanazawa-u.ac.jp>

Documentation

To view documentation for the version of this package, please choose a version below:

[browseVignettes\("TCC"\)](#)

[PDF](#)

[R Script](#)

TCC

[PDF](#)

[Reference Manual](#)

[Text](#)

Details

biocViews

[DifferentialExpression](#), [RNASEq](#)

Version

1.4.0

In Bioconductor since BioC 2.13 (R-3.0)

License

GPL-2

Depends

R (>= 2.15), methods, [DESeq2](#)

Imports

[samr](#)

Suggests

[RUnit](#), [BiocGenerics](#)

System Requirements

URL

Depends On Me

Imports Me

Suggests Me [compcodeR](#)

CHANGES IN VERSION 1.3.2

- DESeq2 was implemented in TCC for identifying DEGs.
- EBSeq was removed from TCC.
- arguments of 'WAD' function were changed.
- fixed bug in '.testByDeseq' that missing to treat size factors.
- the strategies for DE analysis of paired two-group dataset were implemented.
- add the section for describing DE analysis of paired two-group dataset into vignette.

CHANGES IN VERSION 1.2.0

- this package was released as a Bioconductor package (previously CRAN).
- WAD method for identifying DEGs was added.
- ROKU method for identifying tissue-specific genes was added.
- 'increment' argument of 'calcNormFactor' function was added.
- 'replicates' field of TCC class was deleted.

CHANGES IN VERSION 1.1.3

- 'generateSimulationData' function was renamed to 'simulateReadCount'.
- 'names' field of TCC class was changed to 'gene_id'.
- 'hypoData' was reduced to a smaller data set.
- 'hypoData_mg' was created. This is the simulation dataset which consists of 1,000 genes and 9 samples.

CHANGES IN VERSION 1.0.0

- 'TCC' class was implemented as a R5 reference class. Wrapper functions with functional programming semantics were provided.

NEWSのところは、バージョンアップで主に何が変わったかが簡潔に記述されている。例えばTCC ver. 1.1.3でシミュレーションデータ作成用のgenerateSimulationData関数をsimulateReadCountという名前に変更していることがわかる。

TCC ver. 1.4.0で利用可能な関数名を一覧する場合はReference ManualのPDFをクリック。

[Home](#) » [Bioconductor 2.14](#) » [Software Packages](#) » TCC

TCC

TCC: Differential expression normalization strategies

Bioconductor version: Release

This package provides a series of count data using robust normalization of differentially expressed genes or data normalization to obtain a weighted bottom ranked. This can be done by DEG elimination strategy). A major for several kinds of count data (on) by virtue of the use of combinatorial

Author: Jianqiang Sun, Tomoaki

Maintainer: Jianqiang Sun <wukangstaff.kanazawa-u.ac.jp>

Documentation

To view documentation for the version of this package installed in your system, start R and enter:

```
browseVignettes("TCC")
```

[PDF](#)[R Script](#)

TCC

[Reference Manual](#)[Text](#)[NEWS](#)

Details

biocViews[DifferentialExpression](#), [RNASEq](#), [Sequencing](#), [Software](#)**Version**

1.4.0

In Bioconductor since BioC 2.13 (R-3.0)**License**

GPL-2

DependsR (>= 2.15), methods, [DESeq](#), [DESeq2](#), [edgeR](#), [baySeq](#), [ROC](#)**Imports**[samr](#)**Suggests**[RUnit](#), [BiocGenerics](#)**System Requirements****URL****Depends On Me****Imports Me****Suggests Me** [compcodeR](#)

To view documentation for the version of this package installed in your system, start R and enter:

これが関数一覧

Package 'TCC'

August 13, 2014

Type Package

Title TCC: Differential expression analysis for tag count data with robust normalization strategies

Version 1.4.0

Author Jianqiang Sun, Tomoaki Nishiyama, Kentaro Shimizu, and Koji Kadota

Maintainer Jianqiang Sun <wukong@bi.a.u-tokyo.ac.jp>, Tomoaki Nishiyama <tomoakin@staff.kanazawa-u.ac.jp>

Description This package provides a series of functions for performing differential expression analysis from RNA-seq count data using robust normalization strategy (called DEGES). The basic idea of DEGES is that potential differentially expressed genes or transcripts (DEGs) among compared samples should be removed before data normalization to obtain a well-ranked gene list where true DEGs are top-ranked and non-DEGs are bottom ranked. This can be done by performing a multi-step normalization strategy (called DEGES for DEG elimination strategy). A major characteristic of TCC is to provide the robust normalization methods for several kinds of count data (two-group with or without replicates, multi-group/multi-factor, and so on) by virtue of the use of combinations of functions in depended packages.

Depends R (>= 2.15), methods, DESeq, DESeq2, edgeR, baySeq, ROC

Imports samr

Suggests RUnit, BiocGenerics

Enhances snow

License GPL-2

Copyright Authors listed above

biocViews Sequencing, DifferentialExpression, RNASeq

1

2

arab

PDF R Script TC Ref NE Details biocViews Version In Bioconductor since Bi License Depends Imports Suggests System Requirements URL Depends On Me Imports Me Suggests Me

PDF Text

browseVignettes("TCC")

しおり

arab calcAUCValue calcNormFactors clusterSample do_TbT estimateDE exactTestafterTbT filterLowCountGenes getNormalizedData getResult hypoData hypoData_mg hypoData_ts MApplot nakai NBsample plot plotFCPseudocolor ROKU simulateReadCounts TCC TCC-class WAD Index

To view documentation for the version of this package installed in your system, start R and enter:

The screenshot shows the Bioconductor package documentation interface. On the left, there's a sidebar with tabs for PDF, R Script, Text, and Details. Under Details, various package metadata is listed. A red arrow points from the sidebar towards the main content area.

The main content area displays the documentation for the `NBsample` function. The title is `NBsample` (*Sampling from negative binomial distribution*). The `Description` section states: "This methods allow sampling from Negative Binomial distribution for differentially expressed genes having specified level of differential expression in terms of fold change. The proportion of upregulated are also specified. The distribution of original expression levels are generated by resampling real data of *Arabidopsis* RNA-seq data from `arab`. This function will be obsoleted. Use `simulateReadCounts` instead." The `Usage` section shows the function call: `NBsample(DEG_foldchange = 4, repA = 3, repB = 3, Ngene = 3000, PDEG = 0.15, PA = 0.2)`. The `Arguments` section lists parameters: `DEG_foldchange` (Fold change value of differentially expressed genes), `repA` (Replicate number for sample A), `repB` (Replicate number for sample B), `Ngene` (Number of genes to produce), `PDEG` (Proportion of differentially expressed genes), and `PA` (Proportion of upregulated genes in sample A among differentially expressed genes). The `Examples` section shows R code examples.

TCCに限らず、(開発者側の論理で)現在は非推奨の関数が残っている場合がある。大抵、赤の下線部分のように消去予定だという通告か、こっちを使え的なアドバイスが書かれています。いつのリリースで消すかは開発者次第。

```
> library(TCC)
> ?NBsample
> |
```

TCC/パッケージをロード
NBsample関数のマニュ

Reference Manual中の記述と「?関数名」で得られる記述は同じです。

R Documentation

NBsample {TCC}

Sampling from negative binomial distribution

Description

This methods allow sampling from Negative Binomial distribution with specified proportion of differentially expressed genes having specified level of differential expression in terms of fold change. The proportion of upregulated are also specified. The distribution of original expression levels are generated by resampling real data of *Arabidopsis* RNA-seq data from [arab](#). This function will be obsolete. Use [simulateReadCounts](#) instead.

Usage

```
NBsample(DEG_foldchange = 4, repA = 3, repB = 3,
          Ngene = 3000, PDEG = 0.15, PA = 0.2)
```

Arguments

`DEG_foldchange` Fold change value of differentially expressed genes

`repA` Replicate number for sample A

`repB` Replicate number for sample B

`Ngene` Number of genes to produce

`PDEG` Proportion of differentially expressed genes

`PA` Proportion of upregulated genes in sample A among differentially expressed genes (DEGs)

Examples

```
## Not run:
sample <- NBsample()

## End(Not run)
```

Contents

- 3-4. R Bioconductor I、2014/09/09 10:30–14:45、中級、実習
 - Tips : setwd関数を利用した効率的な作業ディレクトリの変更
 - データの型1 : translate関数の入力情報(AAStringSet)
 - Tips : オブジェクトの消去
 - データの型2 : 翻訳配列取得のコードの中身を解説
 - バージョン情報把握とバージョンアップ
 - バージョン違いの影響
 - 過去から現在 : BiostringsパッケージのreadDNAStringSet関数
 - 現在から未来 : BSgenome.Hsapiens.UCSC.hg19パッケージでプロモーター配列取得
 - Bioconductor概観



バージョン違いの影響

■ 過去から現在 : Biostrings パッケージ

イントロ | 一般 | 翻訳配列(translate)を取得 NEW

(Rで)塩基配列解析中のコードは比較的新しいR環境を想定しています。つまり定期的なアップデートを行っているという前提。

塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。

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

1. FASTA形式ファイル([sample1.fasta](#))の場合:

```
in_f <- "sample1.fasta"          #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
fastas <- readDNAStringSet(in_f, format="fasta")#in_fで指定したファイルの読み込み

#本番
hoge <- translate(fastas)        #fastasをアミノ酸配列に翻訳した結果をhogeに格納
names(hoge) <- names(fasta)       #現状では翻訳した結果のオブジェクトhogeのdescriptor
fastas <- hoge                   #hogeの中身をfastasに格納
fastas                               #確認しているだけです

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



バージョン違いの影響

■ 過去から現在 : Biostrings パッケージ

イントロ | 一般 | 翻訳配列(translate)を取得 NEW

塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。

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

1. FASTA形式ファイル([sample1.fasta](#))の場合:

```
in_f <- "sample1.fasta"
out_f <- "hoge1.fasta"

#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, format="f

#本番
hoge <- translate(fasta)
names(hoge) <- names(fasta)
fasta <- hoge
fasta

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

R Console

R version 2.12.0 (2010-10-15)
Copyright (C) 2010 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-pc-mingw32/x64 (64-bit)

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

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

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

バージョン違いの影響

■ 過去から現在 : Biostrings パッケージ

イントロ | 一般 | 翻訳配列(translate)を取得 NEW

塩基配列を読み込んでアミノ酸配列に翻訳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。

理由は、R ver. 2.12.0の頃は
readDNAStringSetという関数名ではなくread.DNAStringSet
だったからです。

1. FASTA形式ファイル(sample1.fasta)

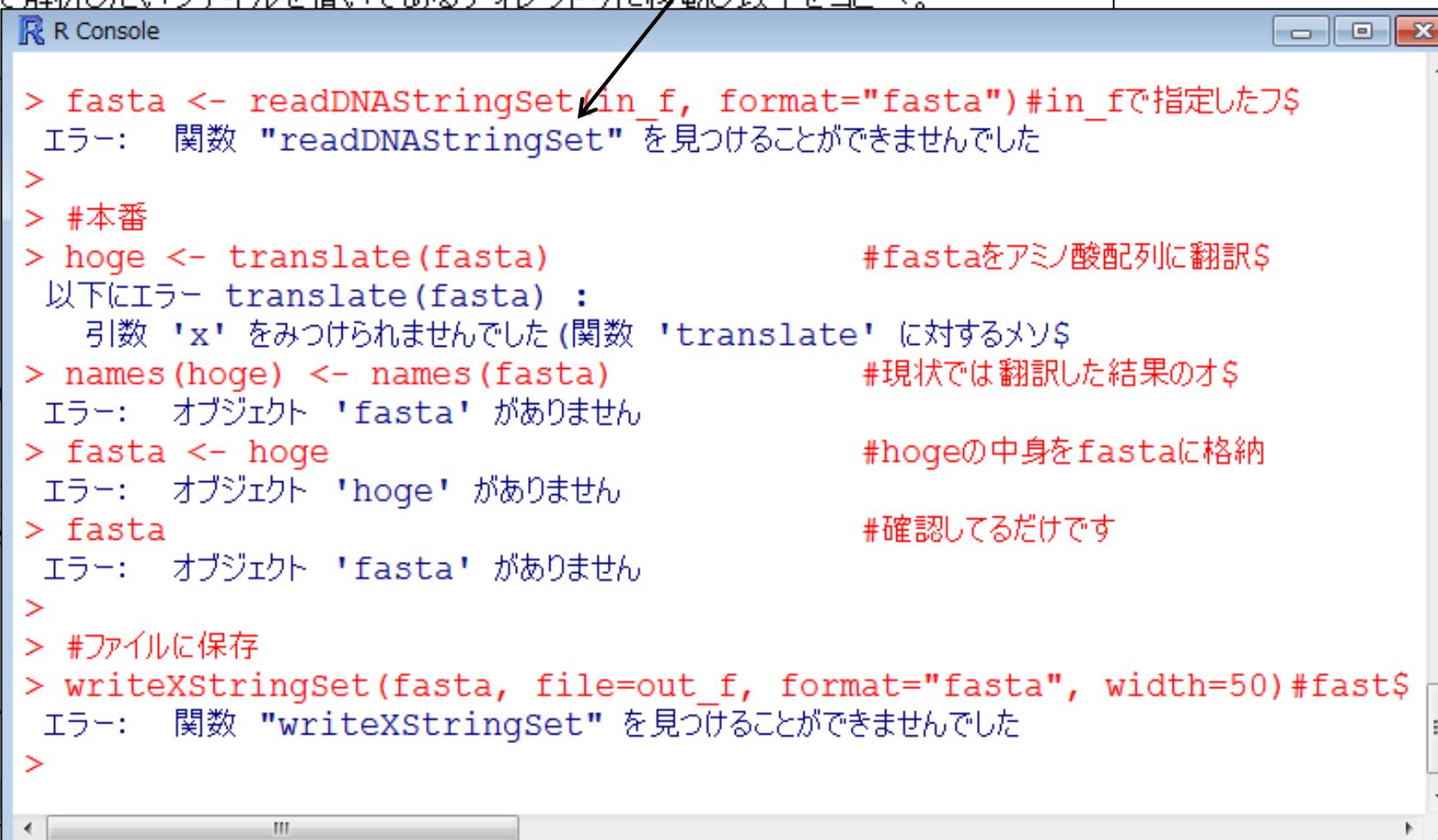
```
in_f <- "sample1.fasta"
out_f <- "hoge1.fasta"

#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 <- readDNAStringS

#本番
hoge <- translate(fasta)
names(hoge) <- names(fasta)
fasta <- hoge
hoge

#ファイルに保存
writeXStringSet(fasta,
```



The screenshot shows an R console window titled 'R Console'. It displays R code attempting to read a FASTA file and translate its DNA sequence into amino acids. The code uses functions like 'readDNAStringSet' and 'writeXStringSet', which are highlighted in red, indicating they are not found. A red arrow points from the text 'readDNAStringSetという関数名ではなくread.DNAStringSetだったからです。' (The reason is that the function name was not 'readDNAStringSet' but 'read.DNAStringSet' in R version 2.12.0.) to the word 'readDNAStringSet' in the error message.

```
> fasta <- readDNAStringSet(in_f, format="fasta") #in_fで指定したフ$  
エラー: 関数 "readDNAStringSet" を見つけることができませんでした  
>  
> #本番  
> hoge <- translate(fasta) #fastaをアミノ酸配列に翻訳$  
以下にエラー translate(fasta) :  
引数 'x' をみつけられませんでした (関数 'translate' に対するメソ$  
> names(hoge) <- names(fasta) #現状では翻訳した結果のオ$  
エラー: オブジェクト 'fasta' がありません  
> fasta <- hoge #hogeの中身をfastaに格納  
エラー: オブジェクト 'hoge' がありません  
> fasta #確認してるだけです  
エラー: オブジェクト 'fasta' がありません  
>  
> #ファイルに保存  
> writeXStringSet(fasta, file=out_f, format="fasta", width=50) #fast$  
エラー: 関数 "writeXStringSet" を見つけることができませんでした  
>
```

バージョン違いの影響

■ 過去から現在 : Biostrings パッケージ

R Console

```
> search()
[1] ".GlobalEnv"
[4] "package:stats"
[7] "package:utils"
[10] "Autoloads"
> ?readDNAStringSet
No documentation for 'readDNAStringSet' in specified packages and 1$  
you could try '??readDNAStringSet'
> ?read.DNAStringSet
> |
```

search()で表示されるパッケージリストには、確かにBiostringsが存在する。`?readDNAStringSet`でNo documentationとなる一方で、`?read.DNAStringSet`ではマニュアルがちゃんと開く。

XStringSet-io {Biostrings} R Documentation

Read/write an XStringSet object from/to a file

Description

Functions to read/write an [XStringSet](#) object from/to a file.

Usage

```
## Read FASTA (or FASTQ) files in an XStringSet object:
read.BStringSet(filepath, format="fasta",
                nrec=-1L, skip=0L, use.names=TRUE)
read.DNAStringSet(filepath, format="fasta",
                  nrec=-1L, skip=0L, use.names=TRUE)
read.RNAStringSet(filepath, format="fasta",
                  nrec=-1L, skip=0L, use.names=TRUE)
read.AAStringSet(filepath, format="fasta",
                 nrec=-1L, skip=0L, use.names=TRUE)
```

バージョン違いの影響

■ 過去から現在 : Biostrings パッケージ

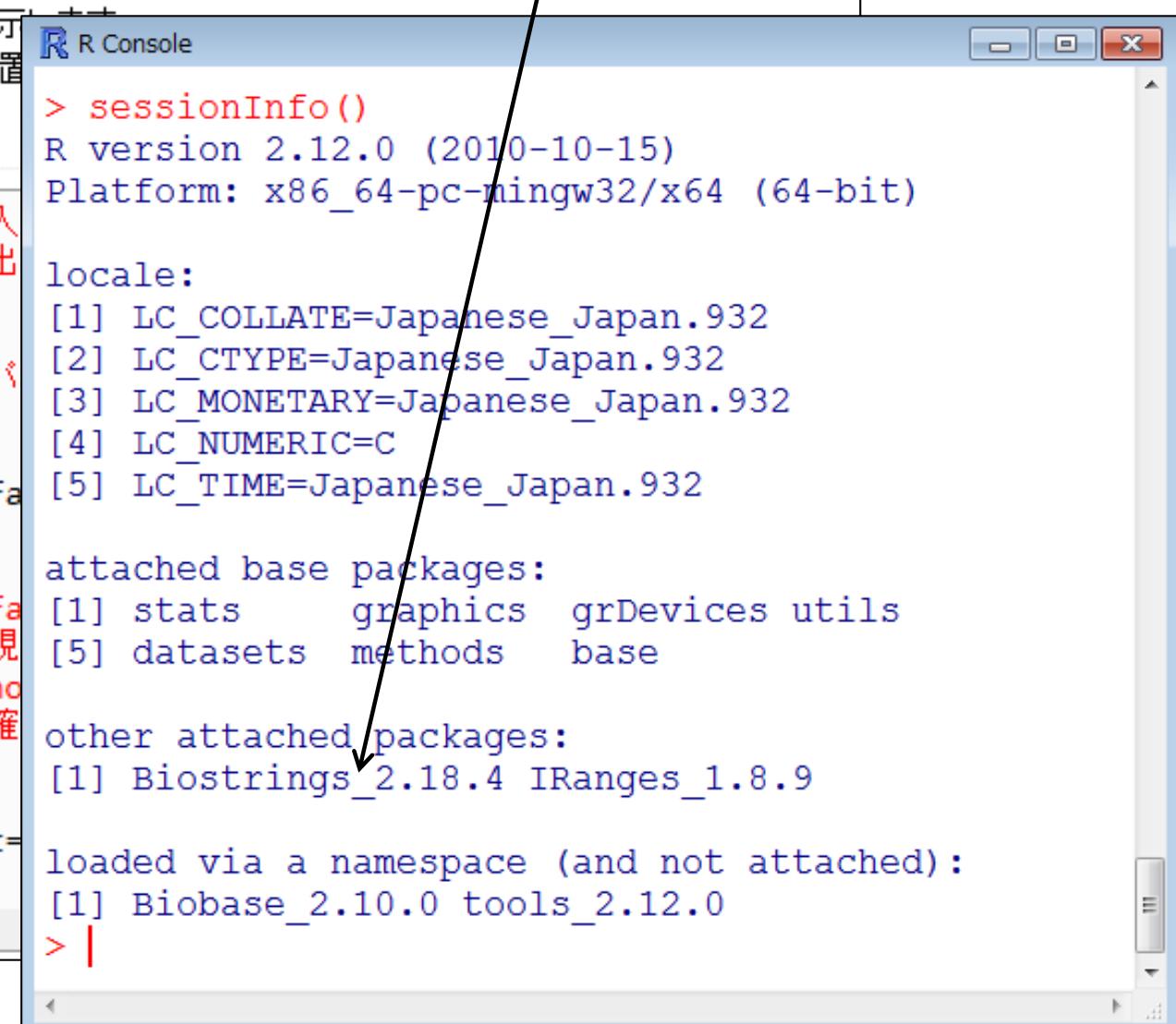
2010年10月リリースのR ver. 2.12.0の頃は、Biostrings ver. 2.18.4です。

イントロ | 一般 | 翻訳配列(translate)を取得 NEW

塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示す
「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く

1. FASTA形式ファイル([sample1.fasta](#))の場合:

```
in_f <- "sample1.fasta"          #入力
out_f <- "hoge1.fasta"           #出力
#必要なパッケージをロード
library(Biostrings)              #パッケージ
#入力ファイルの読み込み
fasta <- readDNAStringSet(in_f, format="fa")
#本番
hoge <- translate(fasta)         #fa
names(hoge) <- names(fasta)       #現状
fasta <- hoge                     #確認
#ファイルに保存
writeXStringSet(fasta, file=out_f, format="fa")
```



R Console window showing session information:

```
> sessionInfo()
R version 2.12.0 (2010-10-15)
Platform: x86_64-pc-mingw32/x64 (64-bit)

locale:
[1] LC_COLLATE=Japanese_Japan.932
[2] LC_CTYPE=Japanese_Japan.932
[3] LC_MONETARY=Japanese_Japan.932
[4] LC_NUMERIC=C
[5] LC_TIME=Japanese_Japan.932

attached base packages:
[1] stats      graphics   grDevices utils
[5] datasets   methods    base

other attached packages:
[1] Biostrings_2.18.4 IRanges_1.8.9

loaded via a namespace (and not attached):
[1] Biobase_2.10.0 tools_2.12.0
> |
```

バージョン違いの影響

R R Console

```
R version 2.15.0 (2012-03-30)
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-pc-mingw32/x64 (64-bit)
```

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

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

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

2012年3月リリースのR ver. 2.15.0
で実行するとエラーは出ません。
この当時はreadDNAStringSetで
もread.DNAStringSetでもどちらで
も受け入れてくれていた。

R R Console

```
> library(Biostrings)
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fasta") #i$ #パッケージ
>
> #本番
> hoge <- translate(fasta) #fastaをア$ #現状では$
> names(hoge) <- names(fasta) #hogeの中$ #確認して$
> fasta <- hoge
> fasta
A AAStringSet instance of length 1
  width seq
[1]      4 SDGL
  names
[1] kadota
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta") $
```

バージョン違いの影響

R R Console

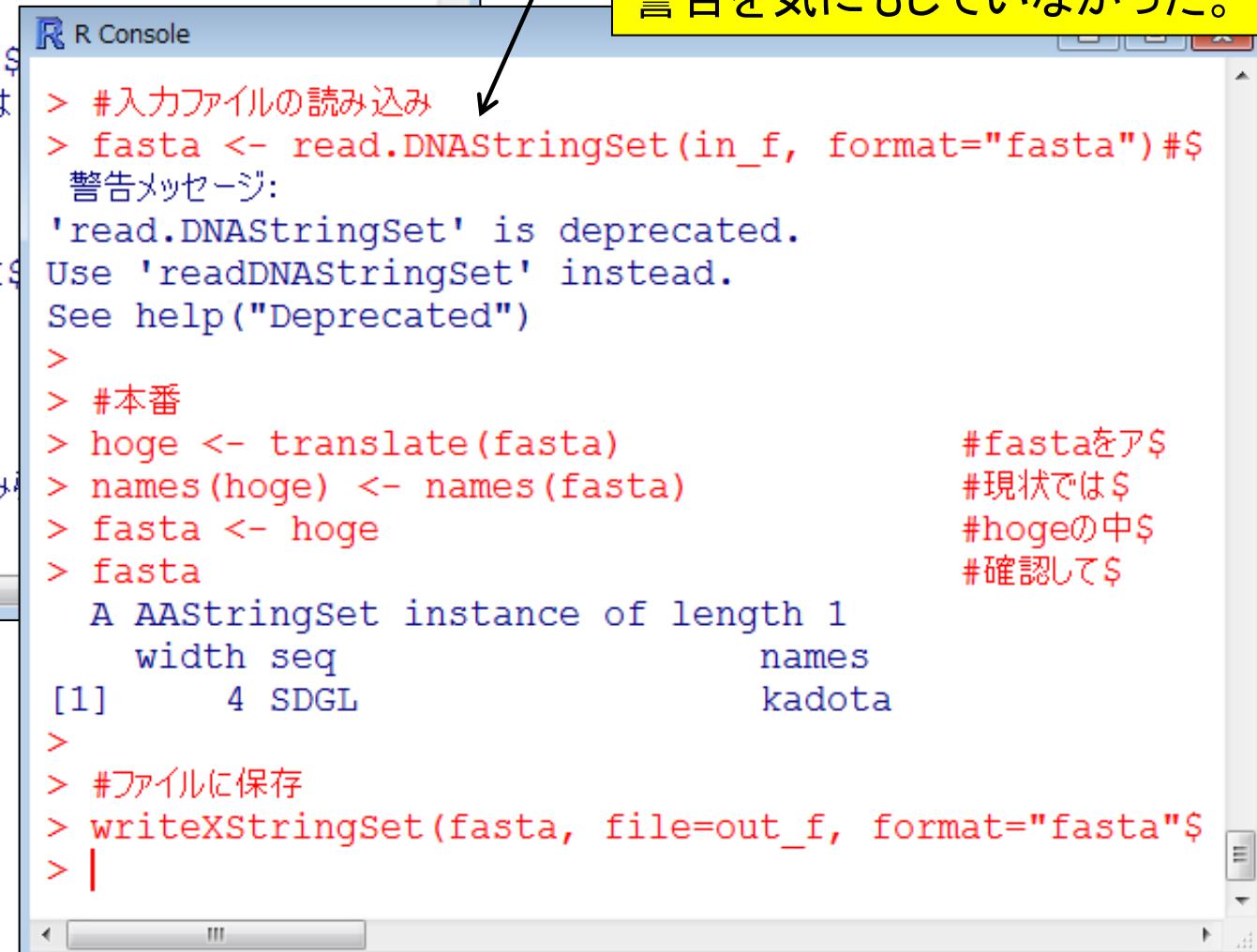
```
R version 2.15.0 (2012-03-30)
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-pc-mingw32/x64 (64-bit)
```

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

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

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

2012年3月リリースのR ver. 2.15.0
 で実行するとエラーは出ません。
 この当時はreadDNAStringSetでもread.DNAStringSetでもどちらでも受け入れてくれていた。そのため、下記の使えなくなるぞという警告を気にもしていなかった。



```
> #入力ファイルの読み込み
> fasta <- read.DNAStringSet(in_f, format="fasta") #$
警告メッセージ:
'read.DNAStringSet' is deprecated.
Use 'readDNAStringSet' instead.
See help("Deprecated")
>
> #本番
> hoge <- translate(fasta)                                #fastaをアシメテ
> names(hoge) <- names(fasta)                            #現状では$名前
> fasta <- hoge                                         #hogeの中身
> fasta
A AAStringSet instance of length 1
  width seq
[1]      4 SDGL
                names
                kadota
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta") $#
> |
```

バージョン違いの影響

R R Console

```
> ?readDNAStringSet ←  
> ?read.DNAStringSet ←  
> sessionInfo()  
R version 2.15.0 (2012-03-30)  
Platform: x86_64-pc-mingw32/x64 (64-bit)
```

```
locale:  
[1] LC_COLLATE=Japanese_Japan.932  
[2] LC_CTYPE=Japanese_Japan.932  
[3] LC_MONETARY=Japanese_Japan.932  
[4] LC_NUMERIC=C  
[5] LC_TIME=Japanese_Japan.932
```

```
attached base packages:  
[1] stats      graphics   grDevices utils  
[5] datasets   methods    base
```

```
other attached packages:  
[1] Biostings_2.26.3   IRanges_1.16.6  
[3] BiocGenerics_0.4.0 BiocInstaller_1.8.3
```

```
loaded via a namespace (and not attached):  
[1] parallel_2.15.0 stats4_2.15.0   tools_2.15.0  
> |
```

2012年3月リリースのR ver. 2.15.0で実行するとエラーは出ません。この当時はreadDNAStringSetでもread.DNAStringSetでもどちらでも受け入れてくれていた。そのため、下記の使えなくなるぞという警告を気にもしていなかった。この頃は、?readDNAStringSetでも?read.DNAStringSetでもマニュアルがちゃんと開いていた。

ファイル名 : rcode_20140909.txt

ファイル名 : rcode_translate3.txt

```
#####↓  
### 作業ディレクトリの変更↓  
#####↓  
setwd("C:/Users/kadota/Desktop/hoge") # デスクトップ上の#  
#setwd("C:/Users/kadota/Desktop") # 「デスクトップ」  
#setwd("C:/Users/kadota/Documents") # 「マイドキュメント」  
↓  
#####↓  
### オブジェクトの消去↓  
#####↓  
rm(list = ls())↓  
↓  
#####↓  
### 翻訳配列取得↓  
#####↓  
in_f <- "sample1.fasta" # 入力ファイル名を指定してin_fに格納↓  
out_f <- "hogel.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="fasta")#in_fで指定したファイルの読み込み↓  
↓  
#本番↓  
hoge <- translate(fasta) #fastaをアミノ酸配列に翻訳した結果をhogeに格納↓  
names(hoge) <- names(fasta) #現状では翻訳した結果のオブジェクトhogeのdescriptionを記述↓  
fasta <- hoge #hogeの中身を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イルに保存↓
```

これらのテンプレートコードのコピーでスライド作成を行っています。

バージョン違いの影響

R R Console

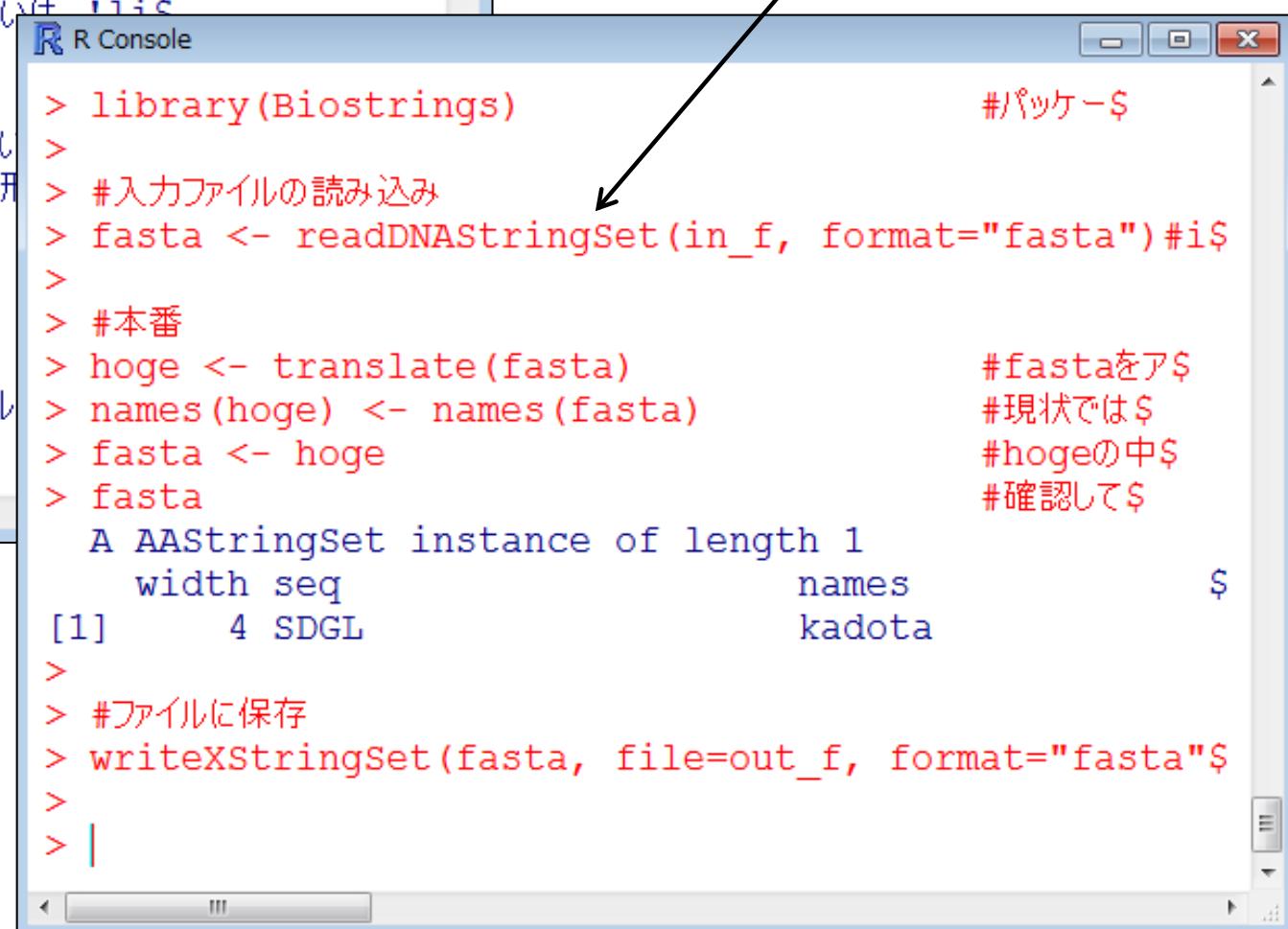
```
R version 3.1.0 (2014-04-10) <- "Spring Dance"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
```

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

R は多くの貢献者による共同プロジェクトです。
 詳しくは 'contributors()' と入力してください。
 また、R や R のパッケージを出版物で引用する際の用語
 '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
 'q()' と入力すれば R を終了します。

2014年4月リリースのR ver. 3.1.0
 では、完全にreadDNAStringSetしか受け入れなくなっている。



The diagram illustrates the relationship between the R console output and the R script. A black arrow points from the line 'R version 3.1.0 (2014-04-10) <- "Spring Dance"' in the R console to the corresponding line in the R script. Another black arrow points from the text '2014年4月リリースのR ver. 3.1.0では、完全にreadDNAStringSetしか受け入れなくなっている。' in the top right corner to the 'readDNAStringSet' call in the R script.

```
R R Console
> library(Biostrings)
#パッケージ
>
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fasta") #i$#
>
> #本番
> hoge <- translate(fasta) #fastaをアソビ
#現状では$#
#hogeの中$#
#確認して$#
> names(hoge) <- names(fasta)
> fasta <- hoge
> fasta
A AAStringSet instance of length 1
  width seq
[1]      4 SDGL
          names
          kadota
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta$")
>
> |
```

バージョン違いの影響

R Console

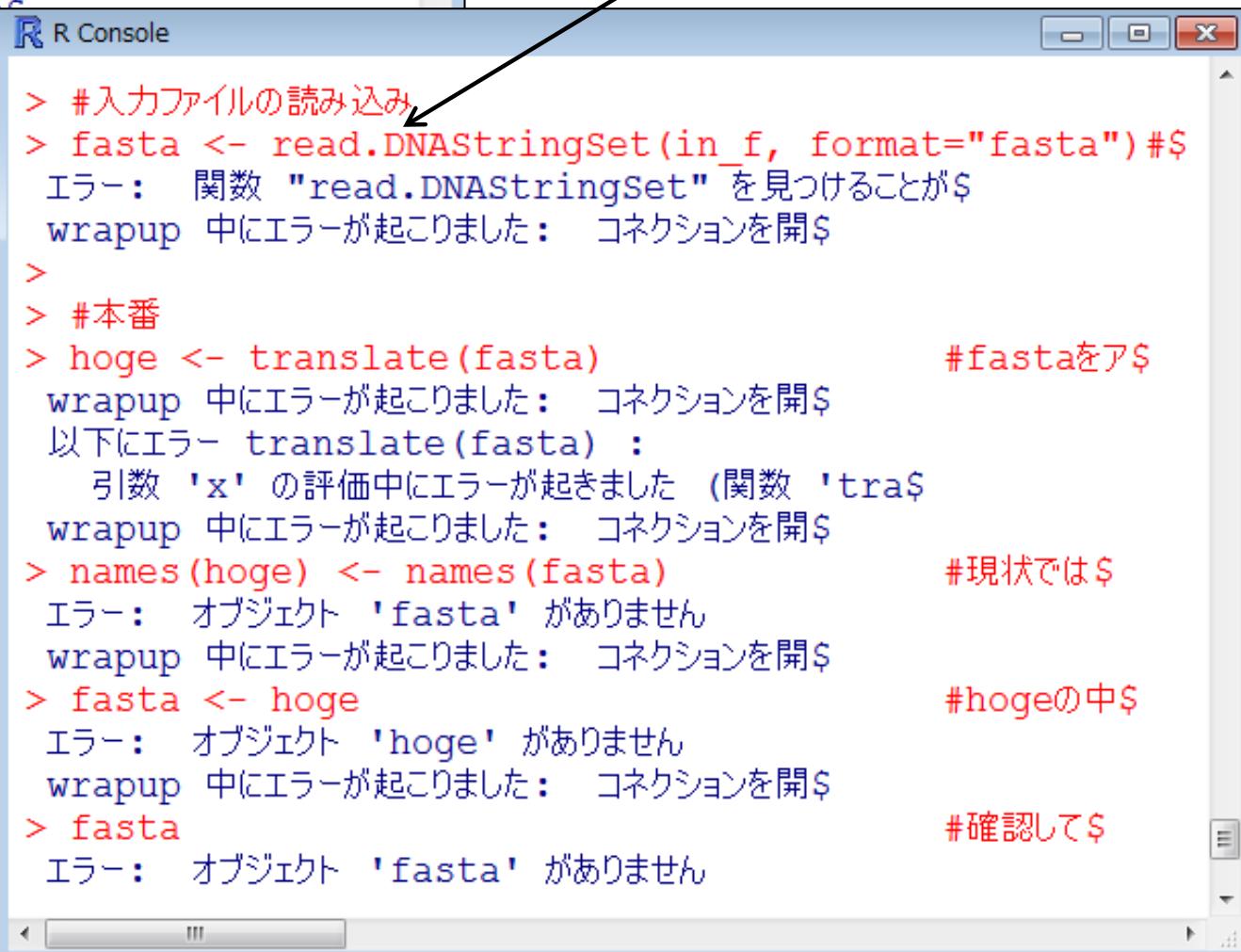
```
R version 3.1.0 (2014-04-10) <- "Spring Dance"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
```

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

R は多くの貢献者による共同プロジェクトです。
 詳しくは 'contributors()' と入力してください。
 また、R や R のパッケージを出版物で引用する際の用語
 '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
 'q()' と入力すれば R を終了します。

2014年4月リリースのR ver. 3.1.0
 では、完全にreadDNAStringSetしか受け入れなくなっている。



```
> #入力ファイルの読み込み
> fasta <- read.DNAStringSet(in_f, format="fasta") #\$エラー: 関数 "read.DNAStringSet"を見つけることが\$wrapup中にエラーが起こりました: コネクションを開\$>
> #本番
> hoge <- translate(fasta) #fastaをア\$wrapup中にエラーが起こりました: コネクションを開\$以下にエラー translate(fasta) :引数 'x' の評価中にエラーがきました (関数 'tra\$wrapup中にエラーが起こりました: コネクションを開\$> names(hoge) <- names(fasta) #現状では\$エラー: オブジェクト 'fasta' がありませんwrapup中にエラーが起こりました: コネクションを開\$> fasta <- hoge #hogeの中\$エラー: オブジェクト 'hoge' がありませんwrapup中にエラーが起こりました: コネクションを開\$> fasta #確認して\$エラー: オブジェクト 'fasta' がありません
```

バージョン違いの影響

2014年4月リリースのR ver. 3.1.0では、完全にreadDNAStringSetしか受け入れなくなっている。

R R Console

```
> ?readDNAStringSet
> ?read.DNAStringSet ← # マニュアルを表示
No documentation for 'read.DNAStringSet' in specified packages and librari$  
you could try '??read.DNAStringSet'
> sessionInfo() ← # 自分のR環境を確認
R version 3.1.0 (2014-04-10)
Platform: x86_64-w64-mingw32/x64 (64-bit)

locale:
[1] LC_COLLATE=Japanese_Japan.932  LC_CTYPE=Japanese_Japan.932
[3] LC_MONETARY=Japanese_Japan.932 LC_NUMERIC=C
[5] LC_TIME=Japanese_Japan.932

attached base packages:
[1] parallel  stats      graphics   grDevices  utils      datasets  methods
[8] base

other attached packages:
[1] TCC_1.4.0          ROC_1.40.0        baySeq_1.18.0
[4] edgeR_3.6.0         limma_3.20.1       DESeq2_1.4.0
[7] RcppArmadillo_0.4.200.0 Rcpp_0.11.1       GenomicRanges_1.16.1
[10] GenomeInfoDb_1.0.2  DESeq_1.16.0       lattice_0.20-29
[13] locfit_1.5-9.1     Biobase_2.24.0    Biostrings_2.32.0
[16] XVector_0.4.0      IRanges_1.22.3    BiocGenerics_0.10.0

loaded via a namespace (and not attached):
[1] annotate_1.42.0      AnnotationDbi_1.26.0 DBI_0.2-7
[4] genefilter_1.46.0    geneplotter_1.42.0  grid_3.1.0
```

バージョン違いの影響

■ 過去から現在:Biostringsパッケージ

- R ver. 2.12.0 (2010年10月リリース)
 - Bioconductor ver. 2.7; Biostrings ver. 2.18.4):read.DNAStringSet関数のみ
- R ver. 2.15.0 (2012年3月リリース)
 - Bioconductor ver. 2.11; Biostrings ver. 2.26.3):移行期
- R ver. 3.1.0 (2014年4月リリース)
 - Bioconductor ver. 2.14; Biostrings ver. 2.32.0):readDNAStringSet関数のみ

これは過去の出来事ですが、
現在進行形の事例もあります。

バージョン違いの影響

- 現在から未来: BSgenome.Hsapiens.UCSC.hg19パッケージ
 - R ver. 3.0.3 (2014年3月リリース)
 - Bioconductor ver. 2.13: パッケージ内に上流配列情報が格納?!されている
 - R ver. 3.1.0 (2014年4月リリース)
 - Bioconductor ver. 2.14: 移行期(Transcript DB形式のオブジェクト利用を推奨)
 - R ver. 3.X.Y (2014年10月リリース?!)
 - Bioconductor ver. 2.15: ...

R ver. 3.1.0で移行期に入っている事例を紹介

Contents

- 3-4. R Bioconductor I、2014/09/09 10:30–14:45、中級、実習
 - Tips : setwd関数を利用した効率的な作業ディレクトリの変更
 - データの型1 : translate関数の入力情報(AAStringSet)
 - Tips : オブジェクトの消去
 - データの型2 : 翻訳配列取得のコードの中身を解説
 - バージョン情報把握とバージョンアップ
 - バージョン違いの影響
 - 過去から現在 : BiostingsパッケージのreadDNAStringSet関数
 - 現在から未来 : BSgenome.Hsapiens.UCSC.hg19パッケージでプロモーター配列取得
 - Bioconductor概観



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скри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 | プロモーター配列 | BSgenome

[BSgenome](#)パッケージを用いて様々な生物種のプロモーター配列(転写開始点近傍配列; 上流配列)を取得するやり方を示します。シロイヌナズナ(*A.thaliana*)、ウシ(*B.taurus*)、線虫(*C.elegans*)、犬(*C.familiaris*)、キイロショウジョウバエ(*D.melanogaster*)、ゼブラフィッシュ(*D.rerio*)、大腸菌(*E.coli*)、イトヨ(*G.aculeatus*)、セキショクヤケイ(*G.gallus*)、ヒト(*H.sapiens*)、アカゲザル(*M.mulatta*)、マウス(*M.musculus*)、チンパンジー(*P.troglodytes*)、ラット(*R.norvegicus*)、出芽酵母(*S.cerevisiae*)、トキソプラズマ(*T.gondii*)と実際に様々な生物種が利用可能であることがわかりますが、生物種によって、上流配列情報がないものもあります(例:シロイヌナズナ)。

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

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

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

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

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

Sep 8-9 2014 NGS速習コーザ

[BSgenome](#)パッケージを用いて様々な生物種のプロモーター配列(転写開始点近傍配列; 上流配列)を取得するやり方を示します。シロイヌナズナ(*A.thaliana*)、ウシ(*B.taurus*)、線虫(*C.elegans*)、犬(*C.familiaris*)、キイロショウジョウバエ(*D.melanogaster*)、ゼブラフィッシュ(*D.rerio*)、大腸菌(*E.coli*)、イトヨ(*G.aculeatus*)、セキショクヤケイ(*G.gallus*)、ヒト(*H.sapiens*)、アカゲザル(*M.mulatta*)、マウス(*M.musculus*)、チンパンジー(*P.troglodytes*)、ラット(*R.norvegicus*)、出芽酵母(*S.cerevisiae*)、トキソプラズマ(*T.gondii*)と実際に様々な生物種が利用可能であることがわかりますが、生物種によって、上流配列情報がないものもあります(例:シロイヌナズナ)。「ファイル」→「ディレクトリの変更」でファイルを保存したいディレクトリに移動し以下をコピペ。

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

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

```
#本番（利用可能なリストアップ；インストール済みとは限らない）
available.genomes()
```

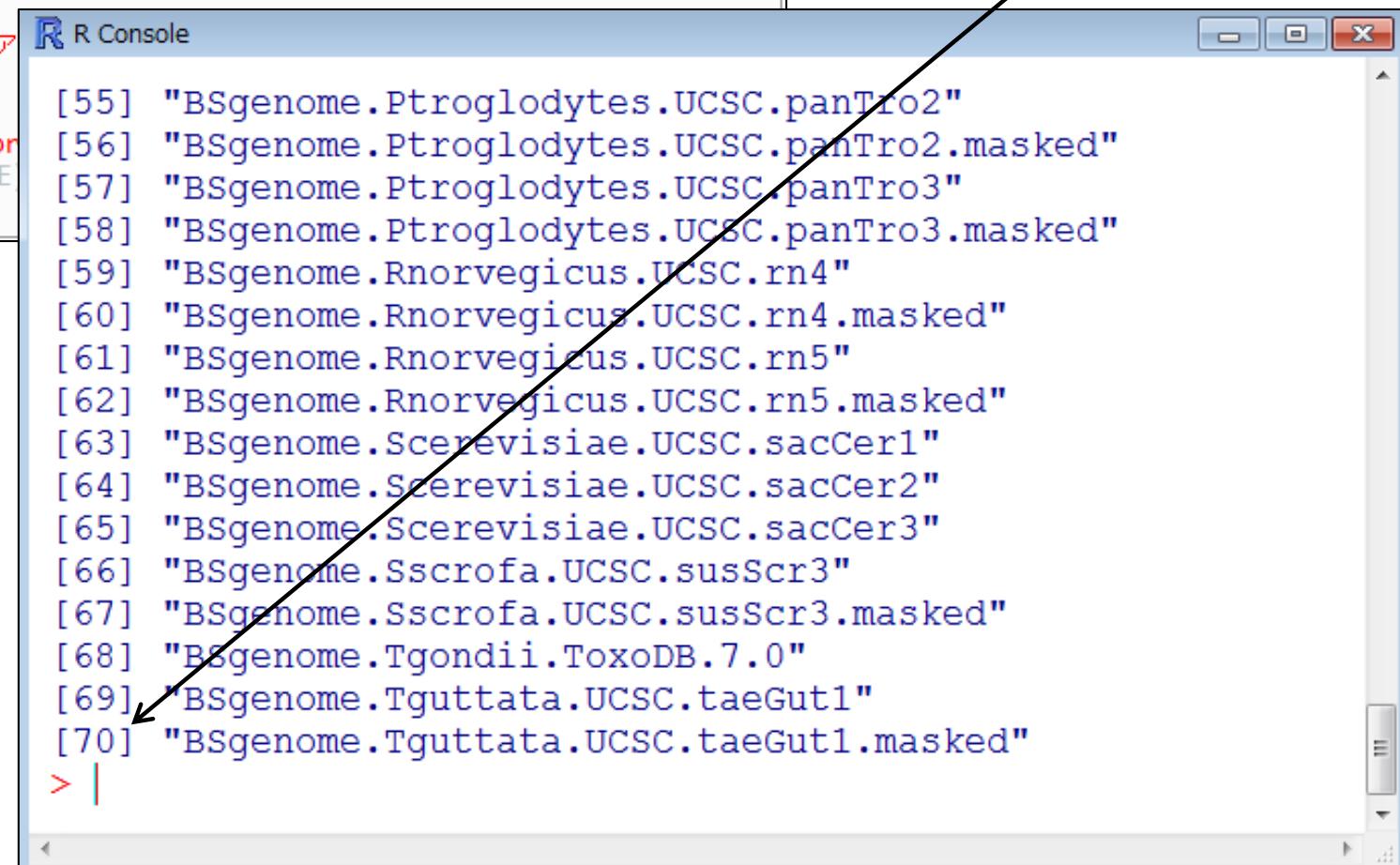
```
#本番（インストール済みの生物種をリストアップ）
installed.genomes()
```

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

#パッケージの読み込み

#このパッケージ中で利用可能なゲノムをリストアップ

R ver. 3.1.0 (Bioconductor ver. 2.14)で利用可能な生物種のパッケージ名をリストアップ。70個あることが分かる。Rのバージョンが古いとおそらくパッケージ数は少なくなる。



```
R R Console
[55] "BSgenome.Ptroglodytes.UCSC.panTro2"
[56] "BSgenome.Ptroglodytes.UCSC.panTro2.masked"
[57] "BSgenome.Ptroglodytes.UCSC.panTro3"
[58] "BSgenome.Ptroglodytes.UCSC.panTro3.masked"
[59] "BSgenome.Rnorvegicus.UCSC.rn4"
[60] "BSgenome.Rnorvegicus.UCSC.rn4.masked"
[61] "BSgenome.Rnorvegicus.UCSC.rn5"
[62] "BSgenome.Rnorvegicus.UCSC.rn5.masked"
[63] "BSgenome.Scerevisiae.UCSC.sacCer1"
[64] "BSgenome.Scerevisiae.UCSC.sacCer2"
[65] "BSgenome.Scerevisiae.UCSC.sacCer3"
[66] "BSgenome.Sscrofa.UCSC.sussScr3"
[67] "BSgenome.Sscrofa.UCSC.sussScr3.masked"
[68] "BSgenome.Tgondii.ToxoDB.7.0"
[69] "BSgenome.Tguttata.UCSC.taeGut1"
[70] "BSgenome.Tguttata.UCSC.taeGut1.masked"
> |
```

```
> available.genomes()
[1] "BSgenome.Alyrata.JGI.v1"
[2] "BSgenome.Amellifera.BeeBase.assembly"
[3] "BSgenome.Amellifera.UCSC.apiMel2"
[4] "BSgenome.Amellifera.UCSC.apiMel2.masked"
[5] "BSgenome.Athaliana.TAIR.04232008"
[6] "BSgenome.Athaliana.TAIR.TAIR9"
[7] "BSgenome.Btaurus.UCSC.bosTau3"
[8] "BSgenome.Btaurus.UCSC.bosTau3.masked"
[9] "BSgenome.Btaurus.UCSC.bosTau4"
[10] "BSgenome.Btaurus.UCSC.bosTau4.masked"
[11] "BSgenome.Btaurus.UCSC.bosTau6"
[12] "BSgenome.Btaurus.UCSC.bosTau6.masked"
[13] "BSgenome.Celegans.UCSC.ce10"
[14] "BSgenome.Celegans.UCSC.ce2"
[15] "BSgenome.Celegans.UCSC.ce6"
[16] "BSgenome.Cfamiliaris.UCSC.canFam2"
[17] "BSgenome.Cfamiliaris.UCSC.canFam2.masked"
[18] "BSgenome.Cfamiliaris.UCSC.canFam3"
[19] "BSgenome.Cfamiliaris.UCSC.canFam3.masked"
[20] "BSgenome.Dmelanogaster.UCSC.dm2"
[21] "BSgenome.Dmelanogaster.UCSC.dm2.masked"
[22] "BSgenome.Dmelanogaster.UCSC.dm3"
[23] "BSgenome.Dmelanogaster.UCSC.dm3.masked"
[24] "BSgenome.Drerio.UCSC.danRer5"
[25] "BSgenome.Drerio.UCSC.danRer5.masked"
[26] "BSgenome.Drerio.UCSC.danRer6"
[27] "BSgenome.Drerio.UCSC.danRer6.masked"
[28] "BSgenome.Drerio.UCSC.danRer7"
```

#この\$

```
[29] "BSgenome.Drerio.UCSC.danRer7"
[30] "BSgenome.Ecoli.NCBI.20080605"
[31] "BSgenome.Gaculeatus.UCSC.gasAcul"
[32] "BSgenome.Gaculeatus.UCSC.gasAcul.masked"
[33] "BSgenome.Ggallus.UCSC.galGal3"
[34] "BSgenome.Ggallus.UCSC.galGal3.masked"
[35] "BSgenome.Ggallus.UCSC.galGal4"
[36] "BSgenome.Ggallus.UCSC.galGal4.masked"
[37] "BSgenome.Hsapiens.NCBI.GRCh38"
[38] "BSgenome.Hsapiens.UCSC.hg17"
[39] "BSgenome.Hsapiens.UCSC.hg17.masked"
[40] "BSgenome.Hsapiens.UCSC.hg18"
[41] "BSgenome.Hsapiens.UCSC.hg18.masked"
[42] "BSgenome.Hsapiens.UCSC.hg19"
[43] "BSgenome.Hsapiens.UCSC.hg19.masked"
[44] "BSgenome.Mmulatta.UCSC.rheMac2"
[45] "BSgenome.Mmulatta.UCSC.rheMac2.masked"
[46] "BSgenome.Mmulatta.UCSC.rheMac3"
[47] "BSgenome.Mmulatta.UCSC.rheMac3.masked"
[48] "BSgenome.Mmusculus.UCSC.mm10"
[49] "BSgenome.Mmusculus.UCSC.mm10.masked"
[50] "BSgenome.Mmusculus.UCSC.mm8"
[51] "BSgenome.Mmusculus.UCSC.mm8.masked"
[52] "BSgenome.Mmusculus.UCSC.mm9"
[53] "BSgenome.Mmusculus.UCSC.mm9.masked"
[54] "BSgenome.Osativa.MSU.MSU7"
[55] "BSgenome.Ptroglodytes.UCSC.panTro2"
```

2013年12月にリリースされたヒトゲノム最新リリース(GRCh38)のRパッケージも利用可能です。

[BSgenome](#)パッケージを用いて様々な生物種のプロモーター配列(転写開始点近傍配列; 上流配列)を取得するやり方を示します。シロイヌナズナ(*A.thaliana*)、ウシ(*B.taurus*)、線虫(*C.elegans*)、犬(*C.familiaris*)、キイロショウジョウバエ(*D.melanogaster*)、ゼブラフィッシュ(*D.rerio*)、大腸菌(*E.coli*)、イトヨ(*G.aculeatus*)、セキショクヤケイ(*G.gallus*)、ヒト(*H.sapiens*)、アカゲザル(*M.mulatta*)、マウス(*M.musculus*)、チンパンジー(*P.troglodytes*)、ラット(*R.norvegicus*)、出芽酵母(*S.cerevisiae*)、トキソプラズマ(*T.gondii*)と実際に様々な生物種が利用可能であることがわかりますが、生物種によって、上流配列情報がないものもあります(例:シロイヌナズナ)。「ファイル」→「ディレクトリの変更」でファイルを保存したいディレクトリに移動し以下をコピペ。

1. 利用可能な生物種とRにインストール済みの生物種をリストアップしたい

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

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

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

#後処理（パッケージ名でだいたいわかるがproviderやversionを分けて表示）
installed.genomes(splitNameParts=TRUE) #インストール済みの
```

実際にインストール済みのものは(このPC環境では)7パッケージであることがわかる。植物のシロイヌナズナ(*Arabidopsis thaliana*)のパッケージは推奨手順通りにインストール作業をしたヒトは存在するはず。

```
R Console
[65] "BSgenome.SceI"
[66] "BSgenome.SscI"
[67] "BSgenome.SscII"
[68] "BSgenome.Tgondii.ToxoDB.7.0"
[69] "BSgenome.Tguttata.UCSC.taeGut1"
[70] "BSgenome.Tguttata.UCSC.taeGut1.masked"
> installed.genomes() #インストール済みの
[1] "BSgenome.Athaliana.TAIR.TAIR9"
[2] "BSgenome.Celegans.UCSC.ce2"
[3] "BSgenome.Drerio.UCSC.danRer7"
[4] "BSgenome.Ecoli.NCBI.20080805"
[5] "BSgenome.Hsapiens.NCBI.GRCh38"
[6] "BSgenome.Hsapiens.UCSC.hg19"
[7] "BSgenome.Mmusculus.UCSC.mm9"
> |
```

NGSデータ解析とR

Rのインストールと起動 NEW

基本的には[こちら](#)または[こちら](#)をご覧ください。

よく分からぬ人でWindowsユーザーの方は以下を参考にしてください。[2](#) のインストール手順は[こちら](#)。2014年5月14日にアップデートしたMac版のます。注意点は、「Mac OS X のバージョンに問わらず R-3.1.0-snowleopard」

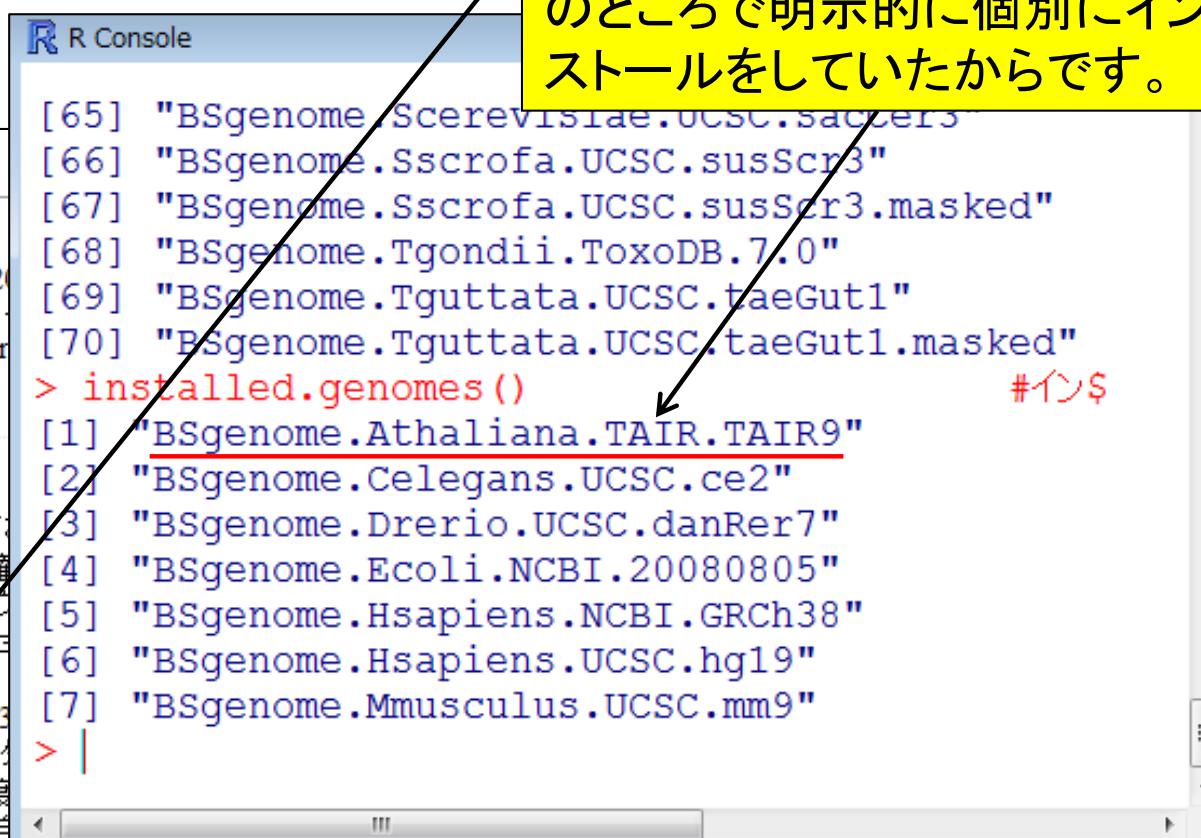
1. Windows release版のインストールの場合:

1. [Rのインストーラ](#)を「実行」
2. 聞かれるがままに「次へ」などを押してとにかくインストールを完了
3. **Windows Vista**の人は(パッケージのインストール中に書き込み権限に)「コントロールパネル」→「ユーザーアカウント」→「ユーザーアカウント制御(UAC)を使ってコンピュータの保護に役立てる」強くお勧めします。
4. インストールが無事完了したら、デスクトップに出現する「R 3.X.Y(3.0.2 より異なります)」または「R x64 3.X.Y(64 bitの場合)」アイコンを
5. 以下を、「Rコンソール画面上」でコピー&ペーストする。10GB程度す。(どこからダウンロードするか?と聞かれるので、その場合は自

```
install.packages(available.packages()[,1], dependencies=TRUE) #CRAN中にある全てのパッケージ
source("http://www.bioconductor.org/biocLite.R") #おまじない
biocLite(all_group()) #Bioconductor中にある全てのパッケージをインストール
biocLite("BSgenome.Athaliana.TAIR.TAIR9", suppressUpdates=TRUE) #Bioconductor中にある全てのパッケージをインストール
```

6. 「コントロールパネル」→「デスクトップのカスタマイズ」→「フォルダオプション」→「表示(タブ)」→「詳細設定」のところで、「登録されている拡張子は表示しない」のチェックを外してください。

理由: このパッケージはデフォルトではインストールされないが、「Rのインストールと起動」のところで明示的に個別にインストールをしていたからです。



The screenshot shows the R Console window with the following text:

```
[65] "BSgenome.Scerevisiae.ucsc.saccer3"
[66] "BSgenome.Sscrofa.UCSC.susScr3"
[67] "BSgenome.Sscrofa.UCSC.susScr3.masked"
[68] "BSgenome.Tgondii.ToxoDB.7.0"
[69] "BSgenome.Tguttata.UCSC.taeGut1"
[70] "BSgenome.Tguttata.UCSC.taeGut1.masked"
> installed.genomes() #インストール済みのGenomeを確認
[1] "BSgenome.Athaliana.TAIR.TAIR9"
[2] "BSgenome.Celegans.UCSC.ce2"
[3] "BSgenome.Drerio.UCSC.danRer7"
[4] "BSgenome.Ecoli.NCBI.20080805"
[5] "BSgenome.Hsapiens.NCBI.GRCh38"
[6] "BSgenome.Hsapiens.UCSC.hg19"
[7] "BSgenome.Mmusculus.UCSC.mm9"
> |
```

Contents

- 3-4. R Bioconductor I、2014/09/09 10:30–14:45、中級、実習
 - Tips : setwd関数を利用した効率的な作業ディレクトリの変更
 - データの型1 : translate関数の入力情報(AAStringSet)
 - Tips : オブジェクトの消去
 - データの型2 : 翻訳配列取得のコードの中身を解説
 - バージョン情報把握とバージョンアップ
 - バージョン違いの影響
 - 過去から現在 : BiostingsパッケージのreadDNAStringSet関数
 - 現在から未来 : BSgenome.Hsapiens.UCSC.hg19パッケージでプロモーター配列取得
 - Bioconductor概観



バージョン違いの影響

[イントロ](#) | [一般](#) | [配列取得](#) | [プロモーター配列](#) | [BSgenome NEW](#)

[BSgenome](#)パッケージを用いて様々な生物種のプロモーター配列(転写開始点、近傍配列; 上流配列)を示します。シロイヌナズナ(*A.thaliana*)、ウシ(*B.taurus*)、線虫(*C.elegans*)、犬(*C.familiaris*)、ウバエ(*D.melanogaster*)、ゼブラフィッシュ(*D.rerio*)、大腸菌(*E.coli*)、イトヨ(*G.aculeatus*)、セキ(*G.gallus*)、ヒト(*H.sapiens*)、アカゲザル(*M.mulatta*)、マウス(*M.musculus*)、チンパンジー(*P.ti*)、
(*R.norvegicus*)、出芽酵母(*S.cerevisiae*)、トキソプラズマ(*T.gondii*)と実際に様々な生物種が利用可能であることがわかりますが、生物種によって上流配列情報がないものもあります(例:シロイヌナズナ)。

「ファイル」→「ディレクトリ」→「上流配列」

5. インストール済みのヒト("BSgenome.Hsapiens.UCSC.hg19")の転写開始点上流配列(1000bp)をmulti-FASTAファイルで保存したい場合:

上流1000bp以外に2000bp, 5000bpもあります...

```
#必要なパッケージ
library(BSgenome)

#本番（利用可能な）
available.genome

#本番（インストール済み）
installed.genome

#後処理（パッケージ）
installed.genome
```

2. ゼブラフィッシュ("BSgenome.Zv9")

400MB程度あります...

```
out_f <- "hoge5.fasta" #出力ファイル名を指定してout_fに格納
param1 <- "BSgenome.Hsapiens.UCSC.hg19" #パッケージ名を指定
param2 <- "upstream1000" #上流1000bpを指定

#必要なパッケージをロード
library(param1, character.only=T) #param1で指定したパッケージの読み込み

#前処理(param1で指定したパッケージ中のオブジェクト名をhogeに統一)
#tmp <- unlist(strsplit(param1, ".", fixed=TRUE))[2] #param1で指定した文字列からオブジェクト名を取得した
tmp <- ls(paste("package", param1, sep=":")) #param1で指定したパッケージで利用可能なオブジェクト名を取得
hoge <- eval(parse(text=tmp)) #文字列tmpをRオブジェクトとしてhogeに格納(パッケージ中にはオブジェクト確認しているだけです(ここで、multiple sequencesのところにparam2)
hoge

#本番
tmp <- paste("hoge$", param2, sep="")
fasta <- eval(parse(text=tmp)) #param2で指定した文字列を含むオブジェクト名を作成した結果をtmpに
#文字列tmpをRオブジェクトとしてfastaに格納

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

ヒトゲノム(BSgenome.Hsapiens.UCSC.hg19)の上流1000bp (upstream1000)の配列を取得し、hoge5.fastaというファイル名で保存するコードをR ver. 3.0.3とR ver. 3.1.0で実行し、挙動の違いを眺めます。

バージョン違いの影響

R ver. 3.0.3での実行例

R version 3.0.3 (2014-03-06) -- "Warm Puppy"
 Copyright (C) 2014 The R Foundation for Statistical Computing
 Platform: x86_64-w64-mingw32/x64 (64-bit)

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

R は多くの貢献者による共同プロジェクトです。
 詳しくは 'contributors()' と入力してください。
 また、R や R のパッケージを出版物で引用する際の形式
 '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
 'q()' と入力すれば R を終了します。

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

R Console

```
| chrUn_g1000237      chrUn_g1000238
| chrUn_g1000239      chrUn_g1000240
| chrUn_g1000241      chrUn_g1000242
| chrUn_g1000243      chrUn_g1000244
| chrUn_g1000245      chrUn_g1000246
| chrUn_g1000247      chrUn_g1000248
| chrUn_g1000249

| multiple sequences (see '?mseqnames'):
| upstream1000 upstream2000 upstream5000
|
| (use the '$' or '[[]' operator to access a given
| sequence)
>
> #本番
> tmp <- paste("hoge$", param2, sep="")
#param2で指す
> fasta <- eval(parse(text=tmp))
#文字列tmpを$に置換
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta",
|
```

バージョン違いの影響

5. インストール済みのヒト("BSgenome.Hsapiens.UCSC.hg19")の転写開始点上流配列(1000bp)を取得したい場合:

上流1000bp以外に2000bp, 5000bpもあります ...

```
out_f <- "hoge5.fasta" #出力ファイル名を指定してout_fに格納
param1 <- "BSgenome.Hsapiens.UCSC.hg19" #パッケージ名を指定
param2 <- "upstream1000" #上流1000bpを指定
```

#必要なパッケージをロード

```
library(param1, character.only=T) #param1で指定したパッケージの読み込み
```

#前処理(param1で指定したパッケージを読み込む)

```
#tmp <- unlist(strsplit(param1$description, "\n"))
tmp <- ls(paste("package", tmp))
hoge <- eval(parse(text=tmp[1]))
hoge
```

#本番

```
tmp <- paste("hoge$", param1$name)
fasta <- eval(parse(text=tmp))
```

#ファイルに保存

```
writeXStringSet(fasta, file="hoge5.fasta")
```

getwd()でみられる作業ディレクトリ上に、out_fで指定した出力ファイル(hoge5.fasta)が作成されているはずです。width列の数値が全て1000になっていることから、上流1,000bpを切り出していることが推察できます。

R Console window showing the execution of R code to extract upstream DNA sequences and save them to a file.

```

R Console
> fasta
A DNAStringSet instance of length 28020
  width seq
[1] 1000 CCACCTGGGGAAAGCG...TGCCGTTCCCTCTCCC names
[2] 1000 TGGACAACGACTTGG...CGTGCTCCTGCCGCC NM_013943_up_1000...
[3] 1000 GAGGCAGAGGTTGCA...ACTATGGGCGGGGCC NM_052998_up_1000...
[4] 1000 ATGTGAGAGAGTTCA...GTCCCTCCCCGCCCTC NM_032785_up_1000...
[5] 1000 ATCAGAACAGTTGGGA...GCTGCCGCTCTAGCC NM_001145277_up_1...
...
[28016] 1000 AGCGACGCGGGGACT...TCCCCCACCAACCCCC NM_001127389_up_1...
[28017] 1000 AAAGACAGAGCGACG...CCACGCCCTCCCCCA NM_033178_up_1000...
[28018] 1000 AAAGACAGAGCGACG...CCACGCCCTCCCCCA NM_033178_up_1000...
[28019] 1000 TTGTATTTTAGTAG...CCTCTAGCTGTGTGT NM_006625_up_1000...
[28020] 1000 TTGTATTTTAGTAG...CCTCTAGCTGTGTGT NM_054016_up_1000...

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

```

バージョン違いの影響

5. インストール済みのヒト("BSgenome.Hsapiens.UCSC.hg19")の転写開始点上流配列(1000bp)を取得したい場合:

上流1000bp以外に2000bp, 5000bpもあります ...

```

out_f <- "hoge5.fasta"          #出力ファイル名を指定してout_fに格納
param1 <- "BSgenome.Hsapiens.UCSC.hg19" #パッケージ名を指定
param2 <- "upstream1000"         #上流1000bpを指定

#必要なパッケージをロード
library(param1, character.only=T) #param1で指定したパッケージをロード

#前処理(param1で指定したパッケージ中のオブジェクト名をhogeに統一)
#tmp <- unlist(strsplit(param1, ".", fixed=TRUE))[2] #param1で指定したパッケージ名をtmpに格納
tmp <- ls(paste("package", param1, sep=":")) #param1で指定したパッケージ名をtmpに格納
hoge <- eval(parse(text=tmp)) #文字列tmpをRオブジェクトとして評価
hoge

#本番
tmp <- paste("hoge$", param2, sep="") #param2で指定した文字列をtmpに格納
fasta <- eval(parse(text=tmp)) #文字列tmpをRオブジェクトとして評価

#ファイルに保存
writeXStringSet(fasta, file=out_f, format="fasta", width=50) #fastaデータをwidth=50でout_fに保存

```

慣れてくると、実物(hoge5.fasta)を眺めて安心するよりも、fastaオブジェクトを眺めるほうが全体像をつかみやすいと思います。出力ファイルの行あたりの塩基数が50になっているのは、widthオプションを50にしているから。

```

>NM_032291 up 1000 chr1 66998825 f chr1:66998825-66999824
CCACCTGGGAAGCGAGGCCGCTCCCGTGTGCGCTGGCGGGAGGCACGC↓
GCTGGCGCGGGACTCGAACCTCGGTCGTCGAAGGGCGCCCCGCAGGGTCC↓
CGTCTGTAGCCCGGACGGCGAGGCCAGGTTGGAGTGGGAGGGAGGCCGGCG↓
TGCACAGCTGTCTTGGTCGGGCTCAGGCTCCGCTCCCTGCCTGCTCCC↓
CTTTCAGCCTCCGCCAGAAAACGATCTCGAGCGTTGCCAGTTGATT↓
CCAGAGCCCCACTCGGGTGGGTTCTTTGTTCTTGTTTAATGACAGT↓
TCCCAGCCCTTCAGCATGTCATGCCAATTAAATTCCCTGGCTCTACGAA↓
GGCAGCTGGGTGAAATTCTTCTGCATCATCCTTGGGATGTTTATG↓
ATGTGACGTCAGCGGATTGATTTCTCTCTGAATGAAGGATGGGAGG↓
GGAGAAAGAGAGACGGAGAGAGAGAGACGCACAGATGTGCACGGAGGC↓
CACAGACACTGACATTGGAATTCCCTCAGGTAAAGGACACCGGAATG↓
GGAGCTTAGAAGTGTGTTGCTAAGATTCCGGCTGCACGGAATTATTAAG↓
TTTTCTTAAAAAAACAAAAAAAGAAAAGAAAAGAAAAGAAAAAGAAC↓
CCCCTCCGCAGCGAGCCACTTAGGTGCTGTTCACGCCAGAGTCCCCTG↓
TTAAGGTGGCAGCCCTTGATAACTAATCTCGGGCACCCAGCCGTTCTGT↓
AAGCTTAAGGAGACGACGAGGGAGGGGGAGTGCGTCACCAGGTGG↓
GGAAGGGGCTGTGTATTTGGTACAAGCGGGAGGCATGGGGTGGAGGG↓
GAATGGGGACGGGAAATAGGTTCTGTGCTCTCCGGGGTATTGTGTCA↓
GGAGATGCAGGCTGGCTACCATGTGACGCGGTCCAAGCTGAAGGGATTG↓
GCCGAGGCAGCGCAGGCCAGCTGGCCAGCTTGCCGTTCCCTCTCCC↓
>NM_013943 up 1000 chr1 25070760 f chr1:25070760-25071759
TGGACAAACGACTTGGAAAGTCCTTAGTGGCCTGCAGGCAGTGAGATTGAG↓
TTCACATCTTACCTTACCTTACCTTACTCTCTCCAAACCCCTACCTCACAC↓

```

バージョン違いの影響

5. インストール済みのヒト("BSgenome.Hsapiens.UCSC.hg19")の転写開始点上流配列(1000bp)を取得したい場合:

上流1000bp以外に2000bp, 5000bpもあります ...

```
out_f <- "hoge5.fasta"
param1 <- "BSgenome.Hsapiens.UCSC.hg19" #パッケージ名
param2 <- "upstream1000" #上流距離

#必要なパッケージをロード
library(param1, character.only=T)      #param1を読み込む

#前処理(param1で指定したパッケージ中のオブジェクトをリスト化)
#tmp <- unlist(strsplit(param1, ".", fixed=T))
#tmp <- ls(paste("package", param1, sep=":"))
#hoge <- eval(parse(text=tmp))          #文字列tmpを評価
#hoge
#確認

#本番
tmp <- paste("hoge$", param2, sep="")
fasta <- eval(parse(text=tmp))          #文字列tmpを評価

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

R Console

```
| chrUn_g1000237           chrUn_g1000238
| chrUn_g1000239           chrUn_g1000240
| chrUn_g1000241           chrUn_g1000242
| chrUn_g1000243           chrUn_g1000244
| chrUn_g1000245           chrUn_g1000246
| chrUn_g1000247           chrUn_g1000248
| chrUn_g1000249

| multiple sequences (see '?mseqnames'):
| upstream1000  upstream2000  upstream5000
|
| (use the '$' or '[[]' operator to access a given
| sequence)
>
> #本番
> tmp <- paste("hoge$", param2, sep="")    #param2で指定
> fasta <- eval(parse(text=tmp))             #文字列tmpを評価
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", $)
> |
```

R ver. 3.0.3では、特に何の問題もなく上流配列を取得できます。理由は、param2で指定したupstream1000情報がこのパッケージのバージョン(ver. 1.3.19)中に存在するからです。

バージョン違いの影響

sessionInfo()でR環境情報を取得

5. インストール済みのヒト("BSgenome.Hsapiens.UCSC.hg19")の転写開始点上流配列(1000bp)をmulti-FASTAファイルで保存したい場合:

上流1000bp以外に2000bp, 5000bp

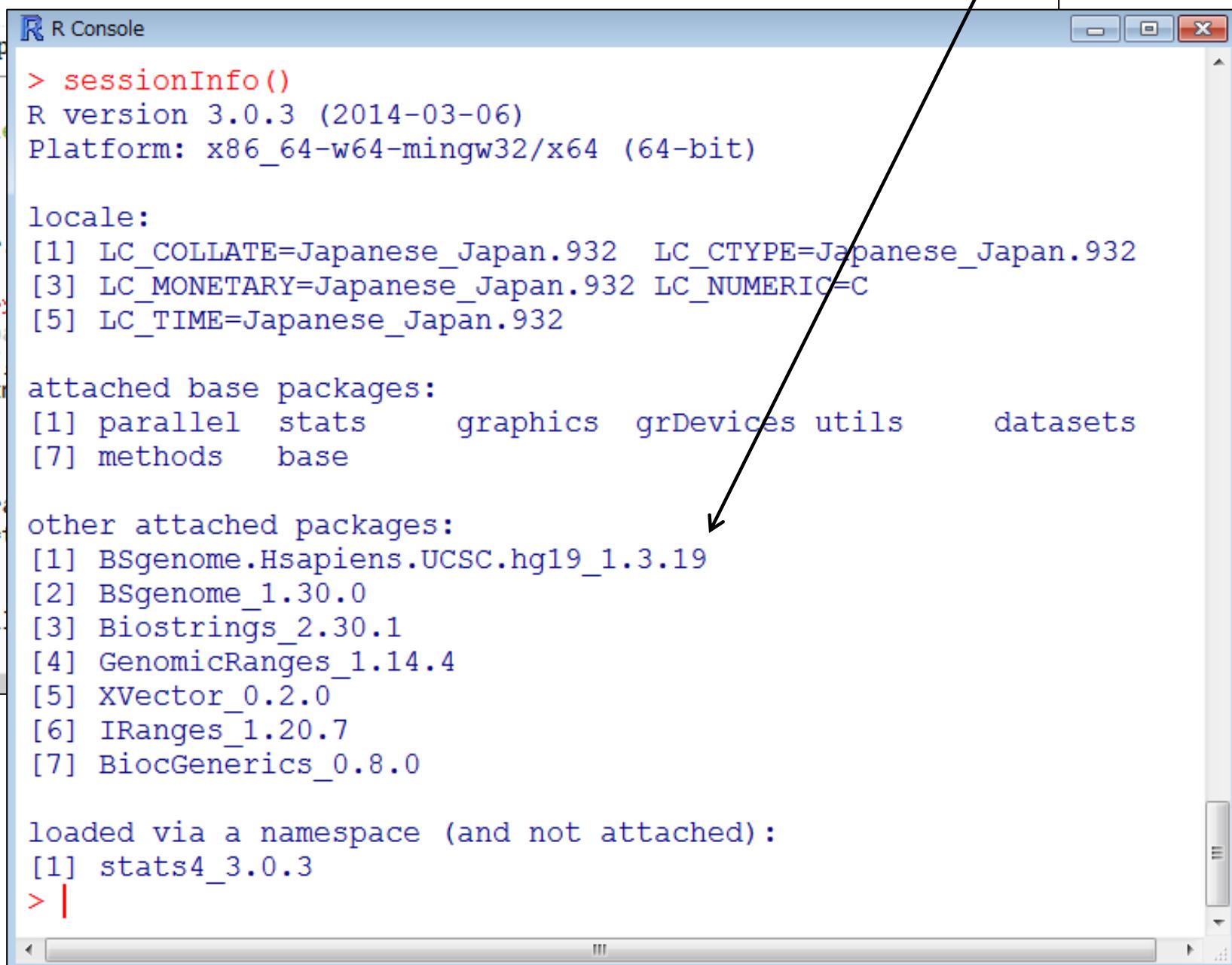
```
out_f <- "hoge5.fasta"
param1 <- "BSgenome.Hsapiens.UCSC.hg19"
param2 <- "upstream1000"
```

```
#必要なパッケージをロード
library(param1, character)

#前処理(param1で指定したパッケージ)
#tmp <- unlist(strsplit(param1, "\n"))
tmp <- ls(paste("package", param1))
hoge <- eval(parse(text=tmp))

#本番
tmp <- paste("hoge$", param2)
fasta <- eval(parse(text=tmp))
```

```
#ファイルに保存
writeXStringSet(fasta, file="hoge5.fasta")
```



R Console window showing the output of sessionInfo(). The window title is "R Console". The output shows the R version (3.0.3), platform (x86_64-w64-mingw32/x64), locale settings, attached base packages (parallel, stats, graphics, grDevices, utils, datasets, methods, base), other attached packages (BSgenome.Hsapiens.UCSC.hg19_1.3.19, BSgenome_1.30.0, Biostrings_2.30.1, GenomicRanges_1.14.4, XVector_0.2.0, IRanges_1.20.7, BiocGenerics_0.8.0), and loaded via a namespace (stats4_3.0.3). A yellow box highlights the "sessionInfo()" command.

```
> sessionInfo()
R version 3.0.3 (2014-03-06)
Platform: x86_64-w64-mingw32/x64 (64-bit)

locale:
[1] LC_COLLATE=Japanese_Japan.932   LC_CTYPE=Japanese_Japan.932
[3] LC_MONETARY=Japanese_Japan.932 LC_NUMERIC=C
[5] LC_TIME=Japanese_Japan.932

attached base packages:
[1] parallel  stats      graphics  grDevices utils      datasets
[7] methods    base

other attached packages:
[1] BSgenome.Hsapiens.UCSC.hg19_1.3.19
[2] BSgenome_1.30.0
[3] Biostrings_2.30.1
[4] GenomicRanges_1.14.4
[5] XVector_0.2.0
[6] IRanges_1.20.7
[7] BiocGenerics_0.8.0

loaded via a namespace (and not attached):
[1] stats4_3.0.3
> |
```

バージョン違いの影響

```
R R Console
R version 3.1.0 <-->
Copyright (C) 2014
Platform: x86_64-w64-mingw32

R は、自由なソフトウェアであります。一定の条件に従えば、自由配布条件の詳細に関しては

R は多くの貢献者による共同開発によって構成されています。詳しくは 'contributors()' を見ることをおすすめします。また、R や R のパッケージの開発者に関する情報は 'citation()' と入力することで確認できます。'demo()' と入力すればデモデータを確認できます。'help()' とすればオンラインヘルプを表示できます。'help.start()' でヘルプページを開くことができます。'q()' と入力すれば R を終了できます。
```

```
> getwd()
[1] "C:/Users/kado"
> |
```

```
R R Console
> #本番
> tmp <- paste("hoge$", param2, sep="")
> fasta <- eval(parse(text=tmp)) #
```

警告メッセージ:

Starting with BioC 2.14, upstream sequences are deprecated.
However they can easily be extracted from the full genome sequences with something like (for example for hg19):

```
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
gn <- sort(genes(txdb))
up1000 <- flank(gn, width=1000)
library(BSgenome.Hsapiens.UCSC.hg19)
genome <- BSgenome.Hsapiens.UCSC.hg19
up1000seqs <- getSeq(genome, up1000)
```

IMPORTANT: Make sure you use a TxDb package (or TranscriptDb object) that contains a gene model based on the exact same reference genome as the BSgenome object you pass to getSeq(). Note that you can make your own custom TranscriptDb object from various annotation resources. See the makeTranscriptDbFromUCSC(), makeTranscriptDbFromBiomart(), and makeTranscriptDbFromGFF() functions in the GenomicFeatures package.

```
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=50) #fastaの$
```

R ver. 3.1.0での実行例。R ver. 3.0.3のときと違って、警告メッセージが出ていることが分かります。BioC 2.14は、2014年4月リリースのBioconductor ver. 2.14のこと。

バージョン違いの影響

5. インストール済みのヒト("BSgenome.Hsapiens.UCSC.hg19")の転写開始点上流配列(1000bp)を取得したい場合:

上流1000bp以外に2000bp, 5000bpもあります ...

警告メッセージは出るもの、上流配列自体は取得できるようです(R ver. 3.1.0)。1年後には使えなくなるかも…。

```
out_f <- "hoge5.fasta"          #出力ファイル名を指定してout_fに格納
param1 <- "BSgenome.Hsapiens.UCSC.hg19" #パッケージ名を指定
param2 <- "upstream1000"         #上流1000bpを指定
```

```
#必要なパッケージを
library(param1, c

#前処理(param1で指
#tmp <- unlist(st
tmp <- ls(paste("
hoge <- eval(pars
hoge
```

```
#本番
tmp <- paste("hog
fasta <- eval(par

#ファイルに保存
writeXStringSet(f
```

R Console

```
See the makeTranscriptDbFromUCSC(), makeTranscriptDbFromBiomart(), and
makeTranscriptDbFromGFF() functions in the GenomicFeatures package.

>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=50) #fastaの$ 
> fasta
A DNAStringSet instance of length 28020
      width seq                                     names
[1] 1000 CCACCTGGGGAAAGCGAG...CTTGCCGTTCCCTCTCCC NM_032291_up_1000...
[2] 1000 TGGACAACGACTTGGAA...CCCGTGCTCCTGCCGCC NM_013943_up_1000...
[3] 1000 GAGGCAGAGGTTGCAGT...GCACTATGGGCGGGGCC NM_052998_up_1000...
[4] 1000 ATGTGAGAGAGTTCAAG...GCGTCCCTCCCGCCCTC NM_032785_up_1000...
[5] 1000 ATCAGAACAGTTGGGATC...GAGCTGCCGCTCTAGCC NM_001145277_up_1...
...
[28016] 1000 AGCGACGCCGGGACTGG...CCTCCCCCACCAACCCC NM_001127389_up_1...
[28017] 1000 AAAGACAGAGCGACGCG...CACCACGCCCTCCCCCA NM_033178_up_1000...
[28018] 1000 AAAGACAGAGCGACGCG...CACCACGCCCTCCCCCA NM_033178_up_1000...
[28019] 1000 TTGTATTTTAGTAGAG...GCCCTCTAGCTGTGTGT NM_006625_up_1000...
[28020] 1000 TTGTATTTTAGTAGAG...GCCCTCTAGCTGTGTGT NM_054016_up_1000...
> |
```

バージョン違いの影響

5. インストール済みのヒト("BSgenome.Hsapiens.UCSC.hg19")の転写開始点上流配列(1000bp)を取得したい場合:

上流1000bp以外に2000bp, 5000bpもあります ...

```

out_f <- "hoge5.fasta"          #出力ファイル名
param1 <- "BSgenome.Hsapiens.UCSC.hg19" #パッケージ名
param2 <- "upstream1000"        #上流1000bp

#必要なパッケージをロード
library(param1, character.only=T)    #param1

#前処理(param1で指定したパッケージ中のオブジェクト)
#tmp <- unlist(strsplit(param1, ".", fixed=TRUE))
#tmp <- ls(paste("package", param1, sep=":")) #package名
#hoge <- eval(parse(text=tmp))                #文字列化
#hoge
#確認
#本番
tmp <- paste("hoge$", param2, sep="")      #param2
fasta <- eval(parse(text=tmp))                #文字列化

#ファイルに保存
writeXStringSet(fasta, file=out_f, format="fas"

```

警告メッセージを要約すると…

- upstreamの情報は使うな
- TranscriptDbオブジェクトを使え

R Console

```

> fasta <- eval(parse(text=tmp))          #文字列化
警告メッセージ:
Starting with BioC 2.14, upstream sequences are no longer included in the
BSgenome package. However they can easily be extracted from the
sequences with something like (for example for hg19):

```

```

library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
gn <- sort(genes(txdb))
up1000 <- flank(gn, width=1000)
library(BSgenome.Hsapiens.UCSC.hg19)
genome <- BSgenome.Hsapiens.UCSC.hg19
up1000seqs <- getSeq(genome, up1000)

```

IMPORTANT: Make sure you use a TxDb package (TxDb.Hsapiens.UCSC.hg19.knownGene) that contains a gene model based on the exact same genomic coordinates as the BSgenome object you pass to getSeq(). You can also use your own custom TranscriptDb object from various sources. See the makeTranscriptDbFromUCSC(), makeTranscriptDbFromGFF(), and makeTranscriptDbFromGTF() functions in the GenomeDB package.

```

>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fas")
>

```

バージョン違いの影響

ファイル名 : rcode_20140909.txt

```
#####
### R ver. 3.1.0の推奨手順で上流配列取得
#####
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
gn <- sort(genes(txdb))
up1000 <- flank(gn, width=1000)
library(BSgenome.Hsapiens.UCSC.hg19)
genome <- BSgenome.Hsapiens.UCSC.hg19
up1000seqs <- getSeq(genome, up1000)

```

こんな感じで上流配列を取得できますという赤枠のテンプレートコードは基本的にコピペでうまくいきます。

R Console

```
> library(TxDb.Hsapiens.UCSC.hg19.knownGene)
> txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
> gn <- sort(genes(txdb))
> up1000 <- flank(gn, width=1000)
> library(BSgenome.Hsapiens.UCSC.hg19)
> genome <- BSgenome.Hsapiens.UCSC.hg19
> up1000seqs <- getSeq(genome, up1000)
> up1000seqs
A DNAStringSet instance of length 23056
      width seq
[1] 1000 ACACATGCTACCGCG...TTTTCTTTCTTAA 100287102
[2] 1000 GCTATTATCACCTAT...GAAACGAATAACTCT 79501
[3] 1000 GAATTAGGCTTCTGC...GAGAAAAAGGCCGGGG 643837
[4] 1000 CGGGGAGCCCCGAGG...CCCGAGCTGGGCCA 148398
[5] 1000 CGGCAGGGCTCCTAT...GGCGGGAGCGGCCGGGG 339451
...
[23052] 1000 GGTGAGCCAATCCTG...GTGGAAATCTCAGCC 283788
[23053] 1000 AGCCCTCCACACAAG...TTCTTCCTCTCCAAC 100507412
[23054] 1000 CGGGGCCAGGGAGT...AGGCCTCCTGGCTGC 728410
[23055] 1000 CGGGGCCAGGGAGT...AGGCCTCCTGGCTGC 100653046
[23056] 1000 CAGGCTGAGCCCTGC...CCGGGGCTACCGCG 100288687
> |
```

```

library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
gn <- sort(genes(txdb))
up1000 <- flank(gn, width=1000)
library(BSgenome.Hsapiens.UCSC.hg19)
genome <- BSgenome.Hsapiens.UCSC.hg19
up1000seqs <- getSeq(genome, up1000)

```

・ イントロ | 一般 | 配列取得 | プロモーター配列 | [BSgenome](#)

(Rで)塩基配列解析の記述法。
上流配列の塩基数を任意に設定できて便利。

8. インストール済みのヒト("BSgenome.Hsapiens.UCSC.hg19")の転写開始点上流配列(1000bp)をmulti-FASTAファイルで保存したい場合:

2014年4月リリースのBioconductor 2.14での推奨手順です。ゲノムのパッケージ(例: [BSgenome.Hsapiens.UCSC.hg19](#))と対応するアノテーションパッケージ(例: [TxDb.Hsapiens.UCSC.hg19.knownGene](#))を読み込んで実行しています。

```

out_f <- "hoge8.fasta"          #出力ファイル名を指定してout_fに格納
param1 <- "BSgenome.Hsapiens.UCSC.hg19" #パッケージ名を指定(BSgenome系のゲノムパッケージ)
param2 <- "TxDb.Hsapiens.UCSC.hg19.knownGene" #アノテーションパッケージ名を指定
param3 <- 1000                   #上流 x

#前処理(指定したパッケージ中のオブジェクト名をgenome)
library(param1, character.only=T)   #指定したパッケージを読み込む
tmp <- ls(paste("package", param1, sep=":"))#対応するアノテーションパッケージ名を取得
genome <- eval(parse(text=tmp))     #文字列を評価してオブジェクトを取得

library(param2, character.only=T)   #指定したアノテーションパッケージを読み込む
tmp <- ls(paste("package", param2, sep=":"))#対応するゲノムパッケージ名を取得
txdb <- eval(parse(text=tmp))     #文字列を評価してオブジェクトを取得

#本番
gn <- sort(genes(txdb))          #遺伝子をソート
hoge <- flank(gn, width=param3)    #指定した幅で上流配列を取得
fasta <- getSeq(genome, hoge)      #指定したゲノムから配列を取得

#ファイルに保存
writeXStringSet(fasta, file=out_f, format="fas"

```

R Console

```

> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fas"
> fasta
A DNAStringSet instance of length 23056
      width seq
      [1] 1000 ACACATGCTAC...TTTCGTTAA 100287102
      [2] 1000 GCTATTATCAC...GAATAACTCT 79501
      [3] 1000 GAATTAGGCTT...AAAGGCGGGG 643837
      [4] 1000 CGGGGAGCCCC...GCTTGGGCCA 148398
      [5] 1000 CGGCGGGGGCTC...GAGCGGGCGGG 339451
      ...
      ...
[23052] 1000 GGTGAGCCAAT...AATCTCAGCC 283788
[23053] 1000 AGCCCTCCACA...CCTCTCCAAC 100507412
[23054] 1000 CGGGGGCCCAGG...TCCTGGCTGC 728410
[23055] 1000 CGGGGGCCCAGG...TCCTGGCTGC 100653046
[23056] 1000 CAGGCTGAGCC...GCTCACCGCG 100288687
>

```

バージョン違いの影響

sessionInfo()でR環境情報を取得

```
> sessionInfo()
R version 3.1.0 (2014-04-10)
Platform: x86_64-w64-mingw32/x64 (64-bit)

locale:
[1] LC_COLLATE=Japanese_Japan.932  LC_CTYPE=Japanese_Japan.932
[3] LC_MONETARY=Japanese_Japan.932 LC_NUMERIC=C
[5] LC_TIME=Japanese_Japan.932

attached base packages:
[1] parallel stats      graphics grDevices utils      datasets methods base

other attached packages:
[1] TxDb.Hsapiens.UCSC.hg19.knownGene_2.14.0 GenomicFeatures_1.16.0
[3] AnnotationDbi_1.26.0                      Biobase_2.24.0
[5] BSgenome.Hsapiens.UCSC.hg19_1.3.99        BSgenome_1.32.0
[7] Biostrings_2.32.0                          XVector_0.4.0
[9] GenomicRanges_1.16.1                      GenomeInfoDb_1.0.2
[11] IRanges_1.22.3                           BiocGenerics_0.10.0

loaded via a namespace (and not attached):
[1] BatchJobs_1.2          BBmisc_1.5           BiocParallel_0.6.0
[4] biomaRt_2.20.0         bitops_1.0-6        brew_1.0-6
[7] codetools_0.2-8         DBI_0.2-7           digest_0.6.4
[10] fail_1.2               foreach_1.4.2       GenomicAlignments_1.0.0
[13] iterators_1.0.7        plyr_1.8.1          Rcpp_0.11.1
[16] RCurl_1.95-4.1         Rsamtools_1.16.0    RSQLite_0.11.4
[19] rtracklayer_1.24.0     sendmailR_1.1-2     stats4_3.1.0
[22] stringr_0.6.2          tools_3.1.0          XML_3.98-1.1
[25] zlibbioc_1.10.0
```

バージョン違いの影響

- 現在から未来: Bsgenome.Hsapiens.UCSC.hg19パッケージ
 - R ver. 3.0.3 (2014年3月リリース)
 - Bioconductor ver. 2.13: パッケージ内に上流配列情報が格納?!されている
 - R ver. 3.1.0 (2014年4月リリース)
 - Bioconductor ver. 2.14: 移行期(Transcript DB形式のオブジェクト利用を推奨)
 - R ver. 3.X.Y (2014年10月リリース?!)
 - Bioconductor ver. 2.15:

R ver. 3.1.0では、2009年2月リリースのヒトゲノム(hg19)パッケージでは警告は出るもののかupstream1000で取得できる。しかし、2013年12月リリースのヒトゲノム(GRCh38)パッケージではすでに取得できなくなっているなど混沌としています。

6. インストール済みのヒト("BSgenome.Hsapiens.NCBI.GRCh38")の転写開始点上流配列(1000bp)をmulti-FASTAファイルで保存したい場合:

2013年12月にリリースされたGenome Reference Consortium GRCh38のRパッケージです。上流配列情報を含まないのでエラーがでます。

```
out_f <- "hoge6.fasta"          #出力ファイル名を指定してout_fに格納
param1 <- "BSgenome.Hsapiens.NCBI.GRCh38" #パッケージ名
param2 <- "upstream1000"        #上流1000bpを指定

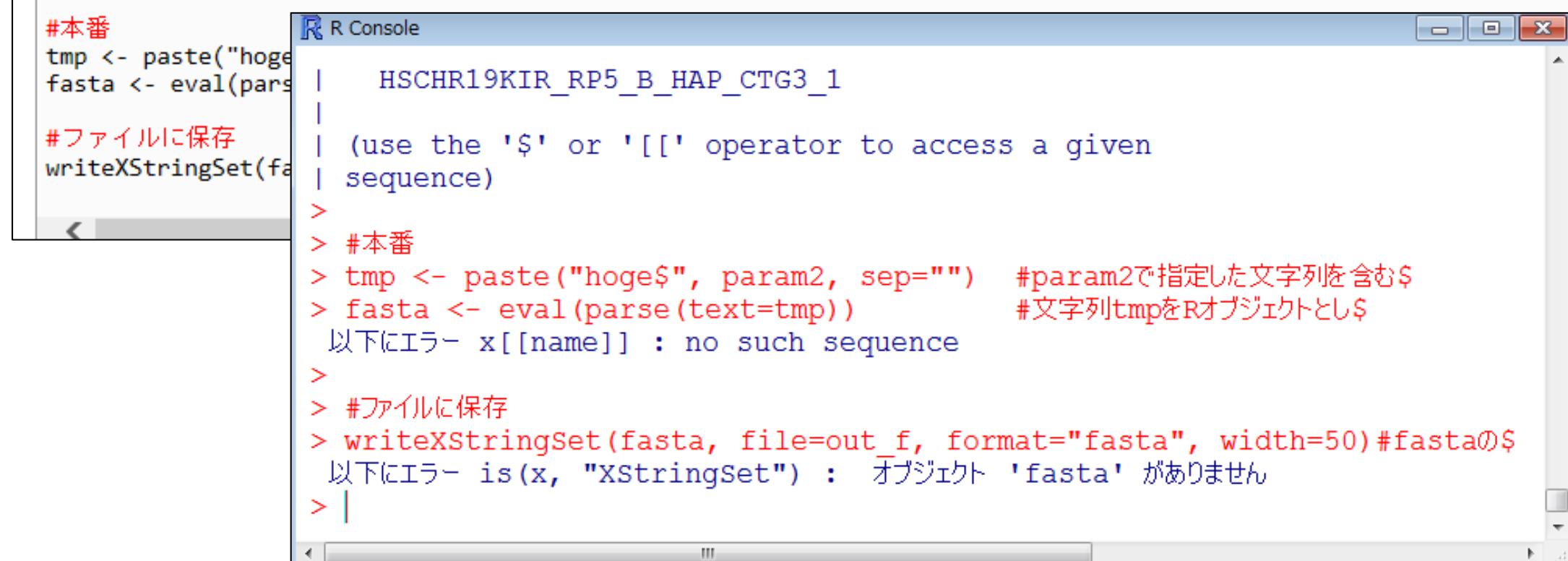
#必要なパッケージをロード
library(param1, character.only=T) #param1で指定したパッケージを読み込む

#前処理(param1で指定したパッケージ中のオブジェクト名を取得)
#tmp <- unlist(strsplit(param1, ".", fixed=TRUE))[2]
tmp <- ls(paste("package", param1, sep=":"))#param1で指定したパッケージの中のオブジェクト名を取得
hoge <- eval(parse(text=tmp))           #文字列tmpをRオブジェクトとして評価
hoge

#本番
tmp <- paste("hoge", param2, sep="")
fasta <- eval(parse(text=tmp))           #param2で指定した文字列を含む$を評価

#ファイルに保存
writeXStringSet(fasta, file=out_f, format="fasta", width=50) #fastaの$を出力
```

R ver. 3.1.0では、2009年2月リリースのヒトゲノム(hg19)パッケージでは警告は出るもの upstream1000で取得できる。しかし、2013年12月リリースのヒトゲノム(GRCh38)パッケージではすでに取得できなくなっているなど混沌としています、
の実例。



The screenshot shows the R Console window with the following text output:

```
R Console
| HSCHR19KIR_RP5_B_HAP_CTG3_1
|
| (use the '$' or '[[]]' operator to access a given
| sequence)
>
> #本番
> tmp <- paste("hoge$", param2, sep="")
> fasta <- eval(parse(text=tmp))           #param2で指定した文字列を含む$を評価
                                         #文字列tmpをRオブジェクトとし$
以下にエラー x[[name]] : no such sequence
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=50) #fastaの$を出力
以下にエラー is(x, "XStringSet") : オブジェクト 'fasta' がありません
> |
```

A red arrow points to the first line of the R code: "out_f <- "hoge6.fasta"".

Contents

- 3-4. R Bioconductor I、2014/09/09 10:30–14:45、中級、実習
 - Tips : setwd関数を利用した効率的な作業ディレクトリの変更
 - データの型1 : translate関数の入力情報(AAStringSet)
 - Tips : オブジェクトの消去
 - データの型2 : 翻訳配列取得のコードの中身を解説
 - バージョン情報把握とバージョンアップ
 - バージョン違いの影響
 - 過去から現在 : BiostingsパッケージのreadDNAStringSet関数
 - 現在から未来 : BSgenome.Hsapiens.UCSC.hg19パッケージでプロモーター配列取得
 - Bioconductor概観



```

library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
gn <- sort(genes(txdb))
up1000 <- flank(gn, width=1000)
library(BSgenome.Hsapiens.UCSC.hg19)
genome <- BSgenome.Hsapiens.UCSC.hg19
up1000seqs <- getSeq(genome, up1000)

```

・ イントロ | 一般 | 配列取得 | プロモーター配列 | [BSgenome](#)

8. インストール済みのヒト("BSgenome.Hsapiens.UCSC.hg19")の転写開始点上流配列

2014年4月リリースの Bioconductor 2.14での推奨手順です。ゲノムのパッケージ(例: [BSgenome.Hsapiens.UCSC.hg19](#))と対応するアノテーションパッケージ(例: [TxDb.Hsapiens.UCSC.hg19.knownGene](#))を読み込んで実行しています。

```

out_f <- "hoge8.fasta"                                #出力ファイル名を指定してout_fに格納
param1 <- "BSgenome.Hsapiens.UCSC.hg19"               #パッケージ名を指定(BSgenome系のゲノムパッケージ)
param2 <- "TxDb.Hsapiens.UCSC.hg19.knownGene"          #転写開始点上流配列を格納するパッケージ名を指定
param3 <- 1000                                         #上流配列の幅

#前処理(指定したパッケージ中のオブジェクト名をgenome)
library(param1, character.only=T)                      #指定したパッケージを読み込む
tmp <- ls(paste("package", param1, sep=":"))#リスト化
genome <- eval(parse(text=tmp))                      #文字列を評価してオブジェクトを生成

library(param2, character.only=T)                      #指定したパッケージを読み込む
tmp <- ls(paste("package", param2, sep=":"))#リスト化
txdb <- eval(parse(text=tmp))                        #文字列を評価してオブジェクトを生成

#本番
gn <- sort(genes(txdb))                            #遺伝子をソート
hoge <- flank(gn, width=param3)                     #指定した幅で転写開始点上流配列を取得
fasta <- getSeq(genome, hoge)                        #指定したゲノムから配列を取得

#ファイルに保存
writeXStringSet(fasta, file=out_f, format="fas"

```

TxDbというパッケージは、R内での統一規格でアノテーション情報を格納したもの。一般にはGTF/GFF形式のアノテーションファイルが用いられる。

R Console

```

> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta")
> fasta
A DNAStringSet instance of length 23056
      width seq
      [1] 1000 ACACATGCTAC...TTTCGTTAA 100287102
      [2] 1000 GCTATTATCAC...GAATAACTCT 79501
      [3] 1000 GAATTAGGCTT...AAAGGCGGGG 643837
      [4] 1000 CGGGGAGCCCC...GCTTGGGCCA 148398
      [5] 1000 CGCGGGGGCTC...GAGCGGCAGG 339451
      ...
      ...
[23052] 1000 GGTGAGCCAAT...AATCTCAGCC 283788
[23053] 1000 AGCCCTCCACA...CCTCTCCAAC 100507412
[23054] 1000 CGGGGCCAGG...TCCTGGCTGC 728410
[23055] 1000 CGGGGCCAGG...TCCTGGCTGC 100653046
[23056] 1000 CAGGCTGAGCC...GCTCACCGCG 100288687
>

```

(Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランскри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)

- What's

 - イントロ | 一般 | Tips | [任意の拡張子でファイルを保存](#) (last modified 2013/09/26)
 - イントロ | 一般 | Tips | [拡張子は同じで任意の文字を追加して保存](#) (last modified 2013/09/26)
 - イントロ | 一般 | 配列取得 | ゲノム配列 | [公共DBから](#) (last modified 2014/05/28)
 - この | すの | 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 2014/04/25)
 - 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 modified 2014/04/25)
 - 門田 | イントロ | 一般 | 配列取得 | トランскриプトーム配列 | [公共DBから](#) (last modified 2014/04/25)
 - イントロ | 一般 | 配列取得 | トランскриптーム配列 | [biomaRt\(Durinck 2009\)](#) (last modified 2014/04/25)
 - マッコニック | 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
 - 東大申込 | イントロ | NGS | 配列取得 | FASTQ or SRALite | [公共DBから](#) (last modified 2014/06/28) NEW
 - イントロ | NGS | 配列取得 | FASTQ or SRALite | [SRAdb\(Zhu 2013\)](#) (last modified 2014/06/26) NEW
 - 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
 - 参考 | 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 modified 2014/06/23)
 - 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\(Lawrence 2013\)](#) (last modified 2013/10/08)
 - 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 modified 2014/04/01)
 - 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)

TxDBというパッケージは、R内での統一規格でアノテーション情報を格納したもの。一般にはGTF/GFF形式のアノテーションファイルが提供されていて、それらを読み込んでTxDBオブジェクトに変換するための関数なども用意されている。

GFF/GTF形式ファイルの例

GFF3形式ファイルの例(シロイヌナズナ; TAIR10_GFF3_genes.gff)

A	B	C	D	E	F	G	H	I
1	Chr1	TAIR10	chromosome	1	30427671	.	.	ID=Chr1;Name=Chr1
2	Chr1	TAIR10	gene	3631	5899	+	.	ID=AT1G01010;Note=protein_coding_gene;Name=AT1G01010
3	Chr1	TAIR10	mRNA	3631	5899	+	.	ID=AT1G01010.1;Parent=AT1G01010;Name=AT1G01010.1;Index=1
4	Chr1	TAIR10	protein	3760	5630	+	.	ID=AT1G01010.1-Protein;Name=AT1G01010.1;Derives_from=AT1G01010.1
5	Chr1	TAIR10	exon	3631	3913	+	.	Parent=AT1G01010.1
6	Chr1	TAIR10	five_prime_UTR	3631	3759	+	.	Parent=AT1G01010.1
7	Chr1	TAIR10	CDS	3760	3913	+	0	Parent=AT1G01010.1,AT1G01010.1-Protein;
8	Chr1	TAIR10	exon	3996	4276	+	.	Parent=AT1G01010.1
9	Chr1	TAIR10	CDS	3996	4276	+	2	Parent=AT1G01010.1,AT1G01010.1-Protein;
10	Chr1	TAIR10	exon	4486	4605	+	.	Parent=AT1G01010.1
11	Chr1	TAIR10	CDS	4486	4605	+	0	Parent=AT1G01010.1,AT1G01010.1-Protein;
12	Chr1	TAIR10	exon	4706	5005	+	.	Parent=AT1G01010.1

遺伝子ごとに、どの染色体
のどの座標上に存在する
のかなどの情報を含むタブ
区切りテキストファイル

GTF形式ファイルの例(ゼブラフィッシュ; Danio rerio.Zv9.75.gtf)

A	B	C	D	E	F	G	H	I
1	#genome-build	Zv9						
2	#genome-version	Zv9						
3	#genome-date	2010-04						
4	#genome-build-accession	NCBI:GCA_000002035.2						
5	#genome-build-last-updated	2014-02						
6	7	protein_coding	gene	100958	101715	+	.	gene_id "ENSDARG00000076051"; gene_name "CABZ01062994.1"; gene
7	7	protein_coding	transcript	100958	101715	+	.	gene_id "ENSDARG00000076051"; transcript_id "ENSDART00000113409
8	7	protein_coding	exon	100958	100975	+	.	gene_id "ENSDARG00000076051"; transcript_id "ENSDART00000113409
9	7	protein_coding	CDS	100958	100975	+	0	gene_id "ENSDARG00000076051"; transcript_id "ENSDART00000113409
10	7	protein_coding	exon	101077	101715	+	.	gene_id "ENSDARG00000076051"; transcript_id "ENSDART00000113409
11	7	protein_coding	CDS	101077	101715	+	0	gene_id "ENSDARG00000076051"; transcript_id "ENSDART00000113409
12	7	protein_coding	gene	116160	117573	+	.	gene_id "ENSDARG00000088691"; gene_name "BX511027.1"; gene_sour
13	7	protein_coding	transcript	116160	117573	+	.	gene_id "ENSDARG00000088691"; transcript_id "ENSDART00000129330

(Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランскриプトーム、正規化、発現変動、統計、モデル、バイオイ
(last modified 2014/07/18, since 2010)

・ イントロ | 一般 | [任意のキーワードを含む行を抽出\(基礎\)](#)

- ・ 書籍 | 日本乳酸菌学会誌 | [第1回イントロダクション](#) (last modified 2014/07/07) NEW
- ・ イントロ | 一般 | [ランダムに行を抽出](#) (last modified 2014/07/17) NEW
- ・ イントロ | 一般 | [任意の文字列を行の最初に挿入](#) (last modified 2014/07/17) NEW
- ・ イントロ | 一般 | [任意のキーワードを含む行を抽出\(基礎\)](#) (last modified 2014/04/11)
- ・ イントロ | 一般 | [ランダムな塩基配列を生成](#) (last modified 2014/06/16)
- ・ イントロ | 一般 | [任意の長さの可能な全ての塩基配列を作成](#) (last modified 2013/06/14)

任意の染色体のみ、遺伝子名をもつもののみ抽出したい場合は、この項目をテンプレートとして利用可能。

イントロ | 一般 | 任意のキーワードを含む行を抽出(基礎)

例えばタブ区切りテキストファイルが手元にあり、この中からリストファイル中の文字列を含む行を抽出するやり方を示します。Linux (UNIX)のgrepコマンドのようなものであり、perlのハッシュのようなものです。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

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

```
in_f1 <- "annotation.txt"          #入力ファイル名を指定してin_f1に格納(アノテーション
in_f2 <- "genelist1.txt"           #入力ファイル名を指定してin_f2に格納(リストファイル)
out_f <- "hogel1.txt"               #出力ファイル名を指定してout_fに格納
param <- 1                          #アノテーションファイル中の検索したい列番号を指定

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

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

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

(Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランскри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)

What's
この
すの
2014
ンフ
2014
門田
マッ
ニッ
したが
2014
東大
申込
jm.
参考

- イントロ | 一般 | Tips | [任意の拡張子でファイルを保存](#) (last modified 2013/09/21)
- イントロ | 一般 | Tips | [拡張子は同じで任意の文字を追加して保存](#) (last modified 2013/09/21)
- 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)
- この
すの
2014
ンフ
2014
門田
マッ
ニッ
したが
2014
東大
申込
jm.
参考
- イントロ | 一般 | 配列取得 | ゲノム配列 | [BSgenome](#) (last modified 2014/06/28)
- 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 2014/06/28)
- イントロ | 一般 | 配列取得 | プロモーター配列 | [BSgenome](#) (last modified 2014/06/28)
- イントロ | 一般 | 配列取得 | プロモーター配列 | [GenomicFeatures\(Lawrence 2013\)](#) (last modified 2014/06/28)
- イントロ | 一般 | 配列取得 | トランскриプトーム配列 | [公共DBから](#) (last modified 2014/06/28)
- イントロ | 一般 | 配列取得 | トランскриプトーム配列 | [biomaRt\(Durinck 2009\)](#) (last modified 2014/06/28)
- 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) NEW
- イントロ | NGS | 配列取得 | FASTQ or SRALite | [SRAdb\(Zhu 2013\)](#) (last modified 2014/06/26) NEW
- 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 modified 2014/06/23) NEW
- 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\(Lawrence 2013\)](#) (last modified 2013/10/08)
- 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 modified 2014/04/01)
- 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\)](#) (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)

1. (Rで)塩基配列解析で提供されている様々な項目を眺め、多くの例題をこなし、できることの全体像を把握。
2. エラー例とその対処法を身につける。基本戦略は「?関数名」や検索エンジンなどを駆使。やりたいことの多くは基本テクニックの組合せで可能。
3. Bioconductorのパッケージリストを眺め、知識やテクニックの幅を広げる。

Bioconductor概観

(Rで)塩基配列解析

~NGS、RNA-seq、ゲノム、トランскриプトーム、正規化、発現変動、など

(last

- 作図について (last modified 2012/09/10)
 - 作図 | M-A plot(基本編) (last modified 2012/10/01)
 - 作図 | ROC曲線
 - 作図 | SplicingG
 - バイオプロセス
 - バイオプロセス
 - バイオプロセス
 - バイオプロセス
 - バイオプロセス
 - バイオプロセス
 - リンク集

リンク集

- R
- Bioconductor: Gentleman et al., Genome Biol., 2004
- CRAN
- ② RjpWiki
- R Tips(竹澤様)
- BioEdit(フリーの配列編集ソフト)
- BioMart: Smedley et al., BMC Bioinformatics, 2009
- DDBJ Read Annotation Pipeline
- EMBOSS explorer (EMBOSS)
- Biostar: Parnell et al., PLoS Computational Biology, 2009
- SEQanswers: Li et al., Bioinformatics, 2009
- NGS WikiBook: Li et al., Briefings in Bioinformatics, 2010
- HT Sequence Analysis with R

NGSに特化した内容ではないが、R Tipsは門田が独学でRを勉強し始めた2005年頃から今でも時々お世話になっているウェブサイト。

PubMed上で「R Bioconductor」でキーワード検索し原著論文があるパッケージのみ探すのも一つの戦略ですが、原著論文公開前のパッケージも見つかります。

Bioconductor概観

まずは、数分待ちます。このようなフリー
ズ局面にときどき遭遇しますが、四隅の
どこかをいじるなどしてウインドウサイズ
を変えると復旧します。「スクリプトの停
止」ボタンを押す必要はありません。こ
の後の作業をやると体感できますが、リ
ストアップするパッケージ数に比例して
表示に時間がかかるようです。最初に
すごく時間がかかるのは824パッケージ
全てをリストアップしているためです。

Contact us: webmaster@bioconductor.org
Hosting provided by Fred Hutchinson Cancer Research Center
Copyright © 2003 - 2014

FRED HUTCHINSON
CANCER RESEARCH CENTER
A LIFE OF SCIENCE

bioconductor.org は、長時間実行中のスクリプトが原因で応答しません。

スクリプトの停止(S) X

http://bioconductor.org/packages/release/BiocViews.html#Software Bioconductor - BioViews

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Home Install Help Developers Search: 「RNA-seq」など、フリーワード検索をやってもいいとは思いますが、経験上あまりうまく引っかかってこないので私はやりません。

Home » BiocViews

All Packages

Bioconductor version 2.14 (Release)

Autocomplete biocViews search:

Packages found under Software:

Show All entries	Search table:	
Package	Maintainer	Title
a4	Tobias Verbeke, Willem Ligtenberg	Automated Affymetrix Array Analysis Umbrella Package
a4Base	Tobias Verbeke, Willem Ligtenberg	Automated Affymetrix Array Analysis Base Package
a4Classif	Tobias Verbeke, Willem Ligtenberg	Automated Affymetrix Array Analysis Classification Package
a4Core	Tobias Verbeke, Willem Ligtenberg	Automated Affymetrix Array Analysis Core Package
a4Preproc	Tobias Verbeke, Willem Ligtenberg	Automated Affymetrix Array Analysis Preprocessing Pack.
a4Reporting	Tobias Verbeke, Willem Ligtenberg	Automated Affymetrix Reporting Package
ABarray	Yongming Andrew Sun	Microarray QA and analysis for Applied Survey Microarray expression data.
ABSSeq	Wentao Yang	ABSSeq: a new RNA method based on a differences and gen model

Software (824)

- AssayDomain (251)
- BiologicalQuestion (204)
- Infrastructure (170)
- ResearchField (151)
- StatisticalMethod (208)
- Technology (511)
- WorkflowStep (406)
- AnnotationData (867)
- ExperimentData (202)

基本的には左側のカテゴリ分けのところを眺めますが、Biostringsなど何を行うパッケージかがある程度分かっているものから逆引きして感覚をつかんでおくとよいでしょう。

http://bioconductor.org/packages/release/BiocViews.html#Software Bioconductor - BiocViews

Bioconductor OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Home Install Help Developers

Search: 「biostrings」と打つとすぐにリストアップされる。

Home » BiocViews

All Packages

Bioconductor version 2.14 (Release)

Autocomplete biocViews search:

Packages found under Software:

Show All entries	Package	Maintainer	Title
Biostatus	Biostatus	H. Pages	String objects representing biological sequences, and matching algorithms
BSgenome	BSgenome	H. Pages	Infrastructure for Biostatus-based genome data packages

Showing 1 to 2 of 2 entries (filtered from 826 total entries) Previous Next

Software (824)
AssayDomain (251)
BiologicalQuestion (204)
Infrastructure (170)
ResearchField (151)
StatisticalMethod (208)
Technology (511)
WorkflowStep (406)
AnnotationData (867)
ExperimentData (202)



http://bioconductor.org/packages/release/bioc/html/Biostrings.html

Bioconductor - Biostrings

Search:

BioconductorのBiostringsパッケージのページに飛びます。

Bioconductor

OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Home Install Help Develop

Home » Bioconductor 2.14 » Software Packages » Biostrings

Biostrings

String objects representing biological sequences, and matching algorithms

Bioconductor version: Release (2.14)

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

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

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

Citation (from within R, enter `citation("Biostrings")`):

Pages H, Aboyoun P, Gentleman R and DebRoy S. *Biostrings: String objects representing biological sequences, and matching algorithms*. R package version 2.32.1.

Installation

To install this package, start R and enter:

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

Documentation

To view documentation for the version of this package installed in your system, start R and enter:

Workflows

Common Bioconductor workflows include:

- [Oligonucleotide Arrays](#)
- [High-throughput Sequencing](#)
- [Counting Reads for Differential Expression](#) (parathyroideSE vignette)
- [Annotation](#)
- [Annotating Variants](#)
- [Annotating Ranges](#)
- [Flow Cytometry and other assays](#)
- [Candidate Binding Sites for Known Transcription Factors](#)
- [Cloud-enabled cis-eQTL search and annotation](#)

Mailing Lists

Post questions about Bioconductor packages to our mailing lists. Read the [posting guide](#) before posting!

- [bioconductor](#)
- [bioc-devel](#)

http://bioconductor.org/packages/release/bioc/html/Biostrings.html

Bioconductor - Biostrings

Documentation

To view documentation for the version of this package installed in your system, start R and enter:

```
browseVignettes("Biostrings")
```

[PDF](#) [R Script](#) A short presentation of the basic classes defined in Biostrings 2
[PDF](#) [R Script](#) Biostrings Quick Overview
[PDF](#) [R Script](#) Handling probe sequence information
[PDF](#) [R Script](#) Multiple Alignments
[PDF](#) [R Script](#) Pairwise Sequence Alignments
[PDF](#) [R Script](#) Reference Manual
[Text](#) NEWS

Details

biocViews [DataImport](#), [DataRepresentation](#), [Genetics](#), [Infrastructure](#), [SequenceMatching](#), [Sequencing](#), [Software](#)

Version 2.32.1

In
Bioconductor BioC 1.6 (R-2.1) or earlier
since

License Artistic-2.0

Depends R (>= 2.8.0), methods, [BiocGenerics](#)(>= 0.5.4), [IRanges](#)(>= 1.21.35), [XVector](#)(>= 0.3.6)

Imports graphics, methods, stats, utils, [BiocGenerics](#), [IRanges](#), [XVector](#), [zlibbioc](#)
[BSgenome](#)(>= 1.13.14), [BSgenome.Celegans.UCSC.ce2](#)(>= 1.3.11),
[BSgenome.Dmelanogaster.UCSC.dm3](#)(>= 1.3.11), [BSgenome.Hsapiens.UCSC.hg18](#),
[drosophila2probe](#), [hqu95av2probe](#), [hqu133aprobe](#), [GenomicFeatures](#)(>= 1.3.14),
[hqu95av2cdf](#), [affy](#)(>= 1.41.3), [affydata](#)(>= 1.11.5), [RUnit](#)

Suggests

System Requirements

URL

Depends On Me [altcdfenvs](#), [Basic4Cseq](#), [BRAIN](#), [BSgenome](#), [ChIPpeakAnno](#), [ChIPsim](#), [cleaver](#),
[CRISPRseek](#), [DASiR](#), [DECIPHER](#), [deepSNV](#), [FDb.FANTOM4.promoters.hg19](#),
[FDb.InfiniumMethylation.hg18](#), [FDb.InfiniumMethylation.hg19](#), [GeneRegionScan](#),
[genomes](#), [GenomicAlignments](#), [GOTHIC](#), [harbChIP](#), [iPAC](#), [JASPAR2014](#), [methVisual](#), [minfi](#),
[MotifDb](#), [motifRG](#), [oligo](#), [oneChannelGUI](#), [pd.huGene.2.0.st](#), [pd.huGene.2.1.st](#),
[pd.mogene.2.0.st](#), [pd.mogene.2.1.st](#), [pd.nugo.hs1a520180](#), [pd.nugo.mm1a520177](#),



BioconductorのBiostringsパッケージのページで、ちょっと下のほうに移動。biocViewsのところで見えるキーワードっぽいのがさきほどのカテゴリ分けに相当。例えば、DataRepresentationをクリックすると…。

<http://bioconductor.org/packages/release/BiocViews.html#DataRepresentation>

Bioconductor - BioViews

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Home Install Help Develop Search:

Home » BiocViews

All Packages

Bioconductor version 2.14 (Release)

Autocomplete biocViews search:

Packages found under DataRepresentation:

Show All entries

Package	Maintainer	Description
AtlasRDF	James Malone	Gene Expression Atlas query and gene set enrichment package.
BaseSpaceR	Adrian Alexa	R SDK for BaseSpace RESTful API
bigmemoryExtras	Peter M. Haverty	An extension of the bigmemory package with added safety, convenience, and a factor class.
Biostrings	H. Pages	String objects representing biological sequences, and matching algorithms
BSgenome	H. Pages	Infrastructure for Biostrings-based genome data packages
cummeRbund	Loyal A. Goff	Analysis, exploration, manipulation, and visualization of Cufflinks high-throughput sequencing data.
flowPlots	N. Hawkins	flowPlots: analysis plots and data class for gated flow cytometry data
flowWorkspace	Greg Finak, Mike Jiang	Import flowJo Workspaces into BioConductor and replicate flowJo gating with flowCore
FunciSNP	Simon G. Coetzee	Integrating Functional Non-coding Datasets with Genetic Association Studies to Identify Candidate Regulatory SNPs
Gang Feng, Pan Du and	Gang Feng, Pan Du and	Integrating Functional Non-coding Datasets with Genetic Association Studies to Identify Candidate Regulatory SNPs

Software (824)
AssayDomain (251)
BiologicalQuestion (204)
Infrastructure (170)
DataImport (74)
DataRepresentation (26)
GUI (14)
ThirdPartyClient (9)
ResearchField (151)
StatisticalMethod (208)
Technology (511)
WorkflowStep (406)
AnnotationData (867)
ExperimentData (202)

カテゴリ分けの階層関係がわかる。Gene Ontologyの階層分類と似たもの。DataRepresentationのカテゴリに含まれるのは26パッケージであることが分かる。赤枠部分がそのリスト。Biostringsは大分類はSoftware、中分類はInfrastructureとなっており、その下の階層のDataRepresentationに含まれていることがわかる。

http://bioconductor.org/packages/release/bioc/html/Biostrings.html

Bioconductor - Biostrings

Documentation

To view documentation for the version of this package installed in your system, start R and enter:

```
browseVignettes("Biostrings")
```

[PDF](#) [R Script](#) A short presentation of the basic classes defined in Biostrings 2

[PDF](#) [R Script](#) Biostrings Quick Overview

[PDF](#) [R Script](#) Handling probe sequence information

[PDF](#) [R Script](#) Multiple Alignments

[PDF](#) [R Script](#) Pairwise Sequence Alignments

[PDF](#) [Text](#) Reference Manual

[Text](#) NEWS

Details

biocViews [DataImport](#), [DataRepresentation](#), [Genetics](#), [Infrastructure](#), [SequenceMatching](#), [Sequencing](#), [Software](#)

Version 2.32.1

In Bioconductor BioC 1.6 (R-2.1) or earlier since

License Artistic-2.0

Depends R (>= 2.8.0), methods, [BiocGenerics](#)(>= 0.5.4), [IRanges](#)(>= 1.21.35), [XVector](#)(>= 0.3.6)

Imports graphics, methods, stats, utils, [BiocGenerics](#), [IRanges](#), [XVector](#), [zlibbioc](#), [BSgenome](#)(>= 1.13.14), [BSgenome.Celegans.UCSC.ce2](#)(>= 1.3.11), [BSgenome.Dmelanoqaster.UCSC.dm3](#)(>= 1.3.11), [BSgenome.Hsapiens.UCSC.hg18](#), [drosophila2probe](#), [hqu95av2probe](#), [hqu133aprobe](#), [GenomicFeatures](#)(>= 1.3.14), [hqu95av2cdf](#), [affy](#)(>= 1.41.3), [affydata](#)(>= 1.11.5), [RUnit](#)

Suggests

System Requirements

URL

Depends On Me [altcdfenvs](#), [Basic4Cseq](#), [BRAIN](#), [BSgenome](#), [ChIPpeakAnno](#), [ChIPsim](#), [cleaver](#), [CRISPRseek](#), [DASiR](#), [DECIPHER](#), [deepSNV](#), [FDb.FANTOM4.promoters.hg19](#), [FDb.InfiniumMethylation.hg18](#), [FDb.InfiniumMethylation.hg19](#), [GeneRegionScan](#), [genomes](#), [GenomicAlignments](#), [GOTHIC](#), [harbChIP](#), [iPAC](#), [JASPAR2014](#), [methVisual](#), [minfi](#), [MotifDb](#), [motifRG](#), [oligo](#), [oneChannelGUI](#), [pd.huGene.2.0.st](#), [pd.huGene.2.1.st](#), [pd.mogene.2.0.st](#), [pd.mogene.2.1.st](#), [pd.nugo.hs1a520180](#), [pd.nugo.mm1a520177](#)

大分類のSoftware、中分類のInfrastructureも存在する。

http://bioconductor.org/packages/release/BiocViews.html#_Infrastructure

Bioconductor - BioViews

Search:

Home Install Help Develop

Home » BiocViews

All Packages

Bioconductor version 2.14 (Release)

Autocomplete biocViews search:

Packages found under Infrastructure:

Show All entries Search table:

Package	Maintainer	Title
aCGH	Peter Dimitrov	Classes and functions for Array Comparative Genomic Hybridization data.
affxparser	Kasper Daniel Hansen	Affymetrix File Parsing SDK
AffyCompatible	Martin Morgan	Affymetrix GeneChip software compatibility
affyContam	V. Carey	structured corruption of affymetrix cel file data
affyio	Benjamin Milo Bolstad	Tools for parsing Affymetrix data files
affylmGUI	Keith Satterley	GUI for affy analysis using limma package
AllelicImbalance	Jesper R Gadin	Investigates allele specific expression
AnnotationDbi	Bioconductor Package Maintainer	Annotation Database Interface
AnnotationForge	Bioconductor Package Maintainer	Code for Building Annotation Database Packages
AnnotationHub	Marc Carlson	A client for retrieving Bioconductor objects from AnnotationHub
aroma.light	Henrik Bengtsson	Light-weight methods for normalization and visualization of microarray data using only basic R data types
		Access the ArrayExpress Microarray

Software (824)
AssayDomain (251)
BiologicalQuestion (204)
Infrastructure (170)

DataImport (74)
DataRepresentation (26)
GUI (14)
ThirdPartyClient (9)
ResearchField (151)
StatisticalMethod (208)
Technology (511)
WorkflowStep (406)
AnnotationData (867)
ExperimentData (202)

DataRepresentation (26パッケージ)のときに比べ、Infrastructure (170パッケージ)の階層表示には時間がかかることがわかる。

http://bioconductor.org/packages/release/bioc/html/Biostrings.html

Bioconductor - Biostrings

Documentation

To view documentation for the version of this package installed in your system, start R and enter:

```
browseVignettes("Biostrings")
```

[PDF](#) [R Script](#) A short presentation of the basic classes defined in Biostrings 2

[PDF](#) [R Script](#) Biostrings Quick Overview

[PDF](#) [R Script](#) Handling probe sequence information

[PDF](#) [R Script](#) Multiple Alignments

[PDF](#) [R Script](#) Pairwise Sequence Alignments

[PDF](#) [R Script](#) Reference Manual

[Text](#) NEWS

Details

biocViews [DataImport](#), [DataRepresentation](#), [Genetics](#), [Infrastructure](#), [SequenceMatching](#), [Sequencing](#), [Software](#)

Version 2.32.1

In
Bioconductor BioC 1.6 (R-2.1) or earlier
since

License Artistic-2.0

Depends R (>= 2.8.0), methods, [BiocGenerics](#)(>= 0.5.4), [IRanges](#)(>= 1.21.35), [XVector](#)(>= 0.3.6)

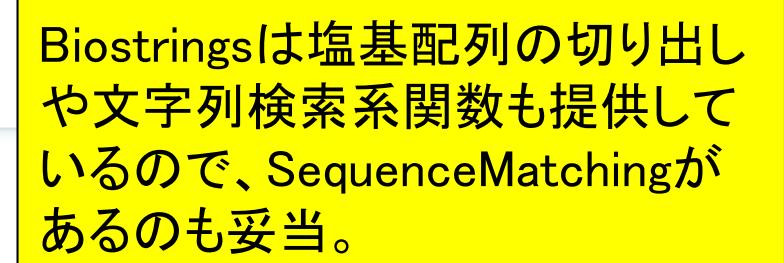
Imports graphics, methods, stats, utils, [BiocGenerics](#), [IRanges](#), [XVector](#), [zlibbioc](#), [BSgenome](#)(>= 1.13.14), [BSgenome.Celegans.UCSC.ce2](#)(>= 1.3.11), [BSgenome.Dmelanogaster.UCSC.dm3](#)(>= 1.3.11), [BSgenome.Hsapiens.UCSC.hg18](#), [drosophila2probe](#), [hqu95av2probe](#), [hqu133aprobe](#), [GenomicFeatures](#)(>= 1.3.14), [hqu95av2cdf](#), [affy](#)(>= 1.41.3), [affydata](#)(>= 1.11.5), [RUnit](#)

Suggests

System Requirements

URL

Depends On Me [altcdfenvs](#), [Basic4Cseq](#), [BRAIN](#), [BSgenome](#), [ChIPpeakAnno](#), [ChIPsim](#), [cleaver](#), [CRISPRseek](#), [DASiR](#), [DECIPHER](#), [deepSNV](#), [FDb.FANTOM4.promoters.hg19](#), [FDb.InfiniumMethylation.hg18](#), [FDb.InfiniumMethylation.hg19](#), [GeneRegionScan](#), [genomes](#), [GenomicAlignments](#), [GOTHIC](#), [harbChIP](#), [iPAC](#), [JASPAR2014](#), [methVisual](#), [minfi](#), [MotifDb](#), [motifRG](#), [oligo](#), [oneChannelGUI](#), [pd.huGene.2.0.st](#), [pd.huGene.2.1.st](#), [pd.mogene.2.0.st](#), [pd.mogene.2.1.st](#), [pd.nugo.hs1a520180](#), [pd.nugo.mm1a520177](#)



Biostringsは塩基配列の切り出しや文字列検索系関数も提供しているので、SequenceMatchingがあるのも妥当。

http://bioconductor.org/packages/release/BiocViews.html#_SequenceMatching

Bioconductor - BioViews

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Home Install Help Develop Search:

Home » BiocViews

All Packages

Bioconductor version 2.14 (Release)

Packages found under SequenceMatching:

Package	Maintainer	Title
Biostrings	H. Pages	String objects representing biological sequences, and matching algorithms
BSgenome	H. Pages	Infrastructure for Biostrings-based genome data packages
cleanUpdTSeq	Sarah Sheppard; Jianhong Ou; Lihua Julie Zhu	This package classifies putative polyadenylation sites as true or false/internally oligodT primed.
cobindR	Manuela Benary	Finding Co-occurring motifs of transcription factor binding sites
CRISPRseek	Lihua Julie Zhu	Design of target-specific guide RNAs in CRISPR-Cas9, genome-editing systems
daqLogo	Jianhong Ou	dagLogo
FunciSNP	Simon G. Coetzee	Integrating Functional Non-coding Datasets with Genetic Association Studies to Identify Candidate Regulatory SNPs
hapFabia	Sepp Hochreiter	hapFabia: Identification of very short segments of identity by descent (IBD) characterized by rare variants in large sequencing data
microRNA	"James F. Reid"	Data and functions for dealing with microRNAs
MmPalateMiRNA	Guy Brock	Murine Palate miRNA Expression Analysis

Autocomplete biocViews search:

Show All entries

Search table:

SequenceMatchingに含まれる17
パッケージの一部しか表示されて
いないが、(Rで)塩基配列解析中
にはないCRISPR関連のパッケージなども存在することに気づく。また、GenomeAnnotationや
GenomicVariationなど様々なパッ
ケージがあることに気づく。

http://bioconductor.org/packages/release/BiocViews.html#_SequenceMatching Bioconductor - BiocViews

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Home Install Help Develop Search:

Home » BiocViews

All Packages

Bioconductor version 2.14 (Release)

Autocomplete biocViews search:

Packages found under SequenceMatching:

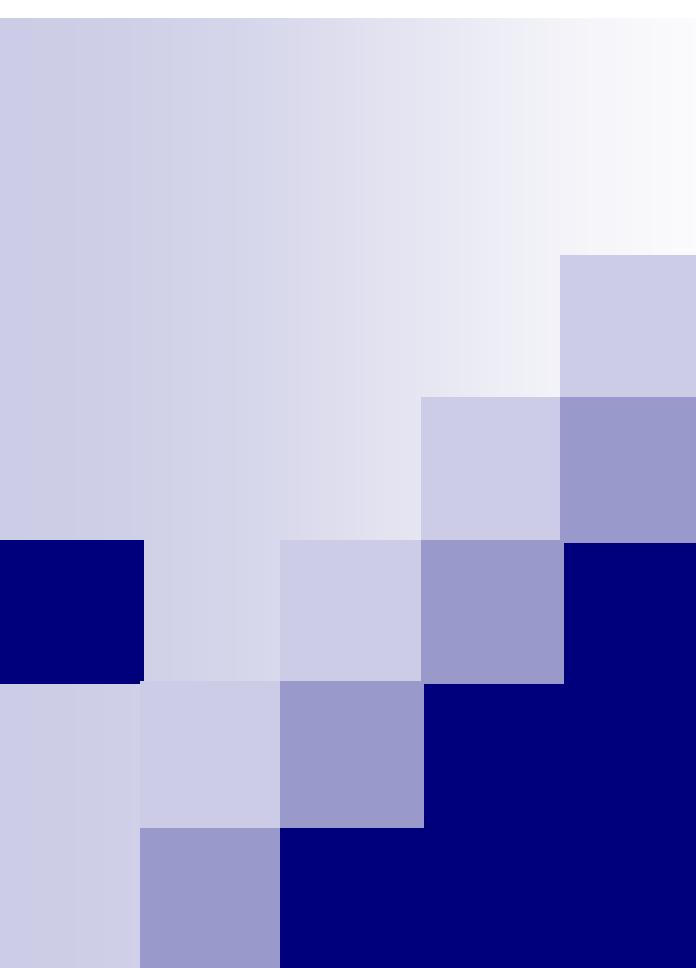
Show All entries

Package	Maintainer	Title
Biostrings	H. Pages	String objects representing biological sequences, and matching algorithms
BSgenome	H. Pages	Infrastructure for Biostrings-based genome data packages
cleanUpdTSeq	Sarah Sheppard; Jianhong Ou; Lihua Julie Zhu	This package classifies putative polyadenylation sites as true or false/internally oligodT primed.
cobindR	Manuela Benary	Finding Co-occurring motifs of transcription factor binding sites
CRISPRseek	Lihua Julie Zhu	Design of target-specific guide RNAs in CRISPR-Cas9, genome-editing systems
daqLogo	Jianhong Ou	dagLogo
FunciSNP	Simon G. Coetzee	Integrating Functional Non-coding Datasets with Genetic Association Studies to Identify Candidate Regulatory SNPs
hapFabia	Sepp Hochreiter	hapFabia: Identification of very short segments of identity by descent (IBD) characterized by rare variants in large sequencing data
microRNA	"James F. Reid"	Data and functions for dealing with microRNAs
MmPalateMiRNA	Guy Brock	Murine Palate miRNA Expression Analysis

Search table:

biocViewsには、中分類に相当するBiologicalQuestionという記述はなかった。自分のパッケージがどこに分類分けされるかを指定するbiocViewsは、キーワード指定のようなものであり、パッケージ開発者次第なのだろうと妄想する。

Software (824)
AssayDomain (251)
BiologicalQuestion (204)
AlternativeSplicing (3)
Coverage (1)
DifferentialExpression (138)
DifferentialMethylation (3)
DifferentialSplicing (3)
FunctionalPrediction (1)
GeneRegulation (15)
GeneSetEnrichment (25)
GenomeAnnotation (3)
GenomicVariation (3)
MotifAnnotation (3)
MotifDiscovery (4)
NetworkEnrichment (7)
NetworkInference (15)



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

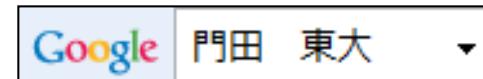
3. データ解析基礎 | 3-5. R Bioconductor II

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

(Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランскри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\(Lawrence 2013\)](#) (last modified 2014/05/29)
- 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 modified 2014/05/29)
- イントロ | NGS | 読み込み | FASTA形式 | [基本情報を取得](#) (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 2014/04/14)

翻訳配列取得以外にも
様々な解析が可能です

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

[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"
out_f <- "hoge1.txt"
```

#入力ファイル名を指定して
#出力ファイル名を指定して

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

```
#必要なパッケージを読み込む
library(Biostrings)

#入力ファイルを読み込む
fasta <- readDNA("hoge4.fa")

#本番(基本)
Total_len
Number_of
Average_len
Median_len
Max_len <-
Min_len <-

#本番(N50)
sorted <-
obj <- (c
N50 <- sort

>contig_1
CGGACAGCTCCTCGGCATCCGGAT

>contig_2
GTCTGCCTCAAGCGCCCCAAGTGGGTTGGAGGCCTAACATCGCAAGTCG
ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCT
GTC

>contig_3
TGTAGGAGAAGGGCGGTATCAGCGTCCACTTACACGATCCGTTACTAATT
GTATGAGGTGGGCA

>contig_4
CGTGCTGATTCCACACAGCAGTAAACCGCGGACCTCTACCTATGAACATG
```

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"
out_f <- "hoge1.txt"
```

#入力ファイル名を指定して
#出力ファイル名を指定して

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

```
#必要なパッケージをインポート
library(Biostrings)

#入力ファイルを読み込む
fasta <- readDNA("hoge4.fa")

#本番(基本)
Total_len
Number_of
Average_len
Median_len
Max_len <-
Min_len <-

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

#入力ファイル名を指定して
#出力ファイル名を指定して

```
>contig_1
CGGACAGCTCCTCGGCATCCGGAT
>contig_2
GTCTGCCTCAAGCGCCCCAAGTGGGTTGGAGGCCTAACATCGCAAGTCG
ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCT
GTC
>contig_3
TGTAGGAGAAGGGCGGTATCAGCGTCCACTTACACGATCCGTTACTAATT
GTATGAGGTGGGCA
>contig_4
CGTGCTGATTCCACACAGCAGTAAACCGCGGACCTCTACCTATGAACATG
```

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

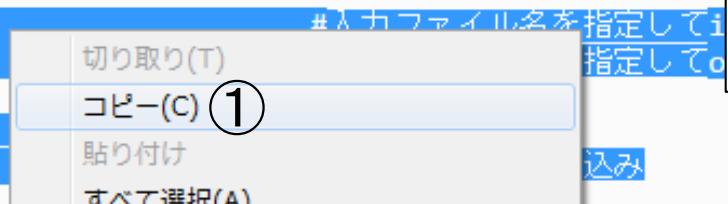
```
in_f <- "hoge4.fa"
out_f <- "hoge1.txt"

#必要なパッケージをロー
library(Biostrings)

#入力ファイルの読み込み
fasta <- readDNAString

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

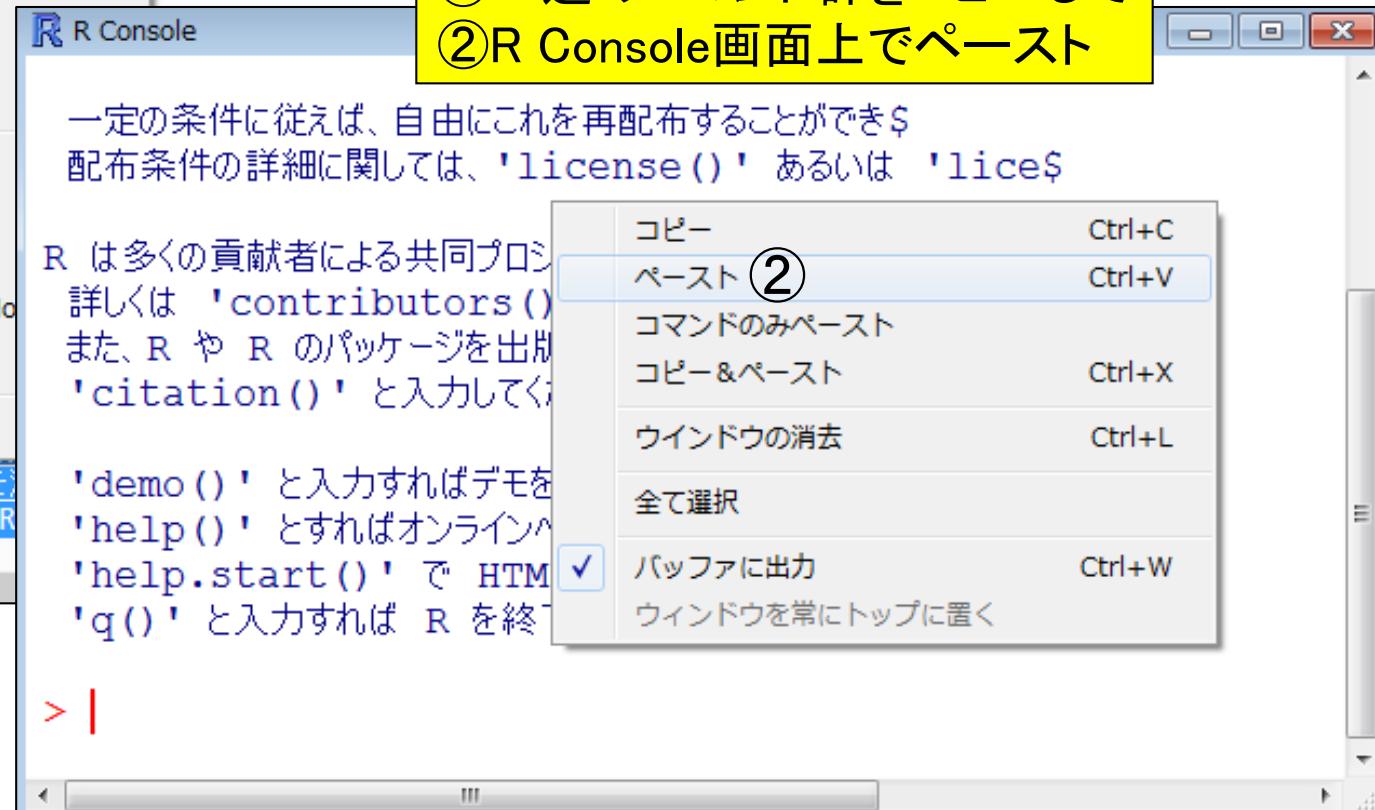
#本番(N50情報取得)
sorted <- rev(sort(width(fasta)))
obj <- (cumsum(sorted) >= Total_len*0.5) #条件を
N50 <- sorted[obj][1] #objがTR
```



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



解析結果

入力: hoge4.fa

```
hoge4.fa - モード
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
>contig_1
CGGACAGCTCCTCGGCATCCGGAT
>contig_2
GTCTGCCTCAAGCGCCCCAAGTGGGTTGGAGGCCTAACATCGCAAGTCG
ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCT
GTC
>contig_3
TGTAGGAGAAGGGCGGTATCAGCGTCCACTTACACGATCCGTTACTAATT
GTATGAGGTCGGGCA
>contig_4
CGTGCTGATTCCACACAGCAGTAAACCGCGGACCTCTACCTATGAACATG
```

出力: hoge1.txt

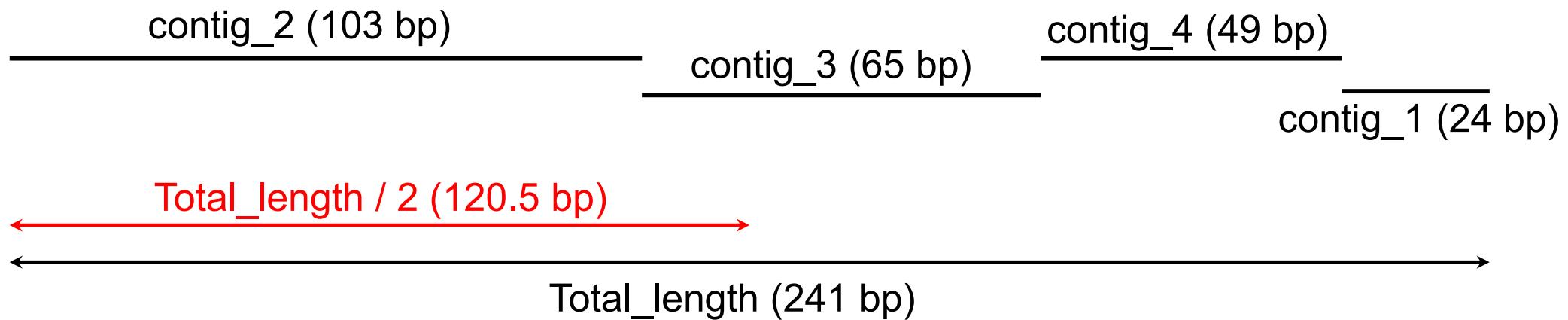
F14		
A	B	
1	Total length (bp)	241
2	Number of contigs	4
3	Average length	60.25
4	Median length	57
5	Max length	103
6	Min length	24
7	N50	65
8	GC content	0.577

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くって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

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



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
```

#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
```

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mu

```

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)

```
huge <- alphabetFrequency(fasta)
```

#A, C, G, T, の数をT

```
CG <- rowSums(hoge[-2:3])
```

#C-Gの総数を計算

```
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)
```

並トータルのGC含量の情報を取得

#ファイルに保存

```
tmp <- NULL
```

```
tmp <- rbind(
```

R Console

```
tmp <- rbind() > fasta
```

```
tmp <- rbind( A DNAstrings
```

```
tmp <- rbind(           # combining the same of long and short
```

```
tmp <- rbind( [1] 24 SGGAGGCTGCTGGGGATGGGGAT
```

```
tmp <- rbind( [1]      24 CGGACAGCTCCCTGGCATCCGGAT
```

```
tmp <- rbind( [2] 103 GTCTGCCTCAAGCGC...CCAGCAC
```

```
write.table(t [3]      65 TGTAGGAGAAGGGCG...GTATGAG
```

[4] 49 CGTGCTGATTCCACA...CTACCTA

```
> alphabetFrequency(fasta)
```

A C G T M B W S Y K V H D B N = +

[1] [1] A B C D E F G H I K R W B I K V H D D N . . .

[2,] 20 34 31 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

[3]

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。

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)
CG <- rowSums(hoge[, 2:3])
ACGT <- rowSums(hoge[, 1:4])
GC_content <- sum(CG)/sum(ACGT)

```

#A,C,G,T..の数を
#C,Gの総数を計算し
#A,C,G,Tの総数を計
#トータルのGC含量の

#ファイルに保存

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。

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)
CG <- rowSums(hoge[, 2:3])
ACGT <- rowSums(hoge[, 1:4])
GC_content <- sum(CG)/sum(ACGT)

```

#A,C,G,T,..の数を
#C,Gの総数を計算し
#A,C,G,Tの総数を計算
#トータルのGC含量の

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

#ファイルに保存

```
tmp <- NULL
tmp <- rbind(
tmp <- rbind(
tmp <- rbind( > hoge
tmp <- rbind(      A   C   G   T   M   R   W   S   Y   K   V   H
tmp <- rbind( [1,]  4   9   7   4   0   0   0   0   0   0   0   0
tmp <- rbind( [2,] 20  34  31  18  0   0   0   0   0   0   0   0
tmp <- rbind( [3,] 16  13  20  16  0   0   0   0   0   0   0   0
tmp <- rbind( [4,] 14  15  10  10  0   0   0   0   0   0   0   0
write.table(t
```

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。

```
> hogel, C(Z, 1)
      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mu

```
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)
CG <- rowSums(hoge[, 2:3])
ACGT <- rowSums(hoge[, 1:4])
GC_content <- sum(CG)/sum(ACGT)

```

#A,C,G,T,..の数を
#C,Gの総数を計算し
#A,C,G,Tの総数を計算
#トータルのGC含量の

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

#ファイルに保存

R R Console

2

```
> 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 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 0 0 0
```

`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。

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mu

```

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

```

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
[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
> 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\(Schneider et al., 1990\)](#) (last modified 2014/07/23) NEW
 - 解析 | 一般 | [上流配列解析 | LDSS\(Yamamoto, 2007\)](#) (last modified 2012/07/17)

解析 | 一般 | GC含量 (GC contents)

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。

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)
CG <- rowSums(hoge[,2:3])      #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
ACGT <- rowSums(hoge[,1:4])    #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

```

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 2014/04/29)
 - イントロ | NGS | 読み込み | FASTQ形式 | 基本情報を取得 (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 (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)

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。以下をペースト。

1. サンプルデータの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みます。

```
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  
#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"
#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
fastas <- readDNAStringSet(in_f, format="fastq")
```

#入力ファイル名を指定してin_fに格納

```
@SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35
NNNNNNNNNNNNNNNNCTACCCCCCCCAGCCCGCCGCA
+SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35
!!!!!!!!!!!
@SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35
NNNNNNNNNNNNNNNAGACAGTTGATTAGCATA
+SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35
!!!!!!!!!!!
@SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35
NNNNNNNNNNNNNNNGGGTGGGGCGTTGTTCTTG
+SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35
!!!!!!!!!!!
@SRR037439.4 HWI-E4_6_30ACL:2:1:0:1240 length=35
```

1. サンプルデータの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)。

・ イントロ | NGS | 読み込み | [FASTQ形式](#)

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          #パッケージの読み込み
library(Biostrings)                #確認してるだけです
#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を読み込みます
fasta
```

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$ #確認してるだけです
> fasta
  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...TTTGTCTTG SRR037439.3 HWI-E...
      [4] 35 NNNNNNNNNN...CCCCTCCCTC SRR037439.4 HWI-E...
      [5] 35 NNNNNNNNNN...GACGCCACCA SRR037439.5 HWI-E...
      ...
      ...
      [496] 35 CTGGACGGCC...CACCCCCCCC SRR037439.496 HWI...
      [497] 35 TGTCACTTGT...CACGGGGCTT SRR037439.497 HWI...
      [498] 35 CCGCCCTTT...CACAAAAAAA SRR037439.498 HWI...
      [499] 35 CGTTCTTGT...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)。

・ イントロ | NGS | 読み込み | [FASTQ形式](#)

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"  
  
#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)  
fastq
```

#入力ファイル名を指定して

#パッケージの読み込み

#in_fで指定したファイルの読み込み
#確認しているだけです

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。

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

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 #パッケージの読み込$  
> library(ShortRead)  
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파$  
> 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:

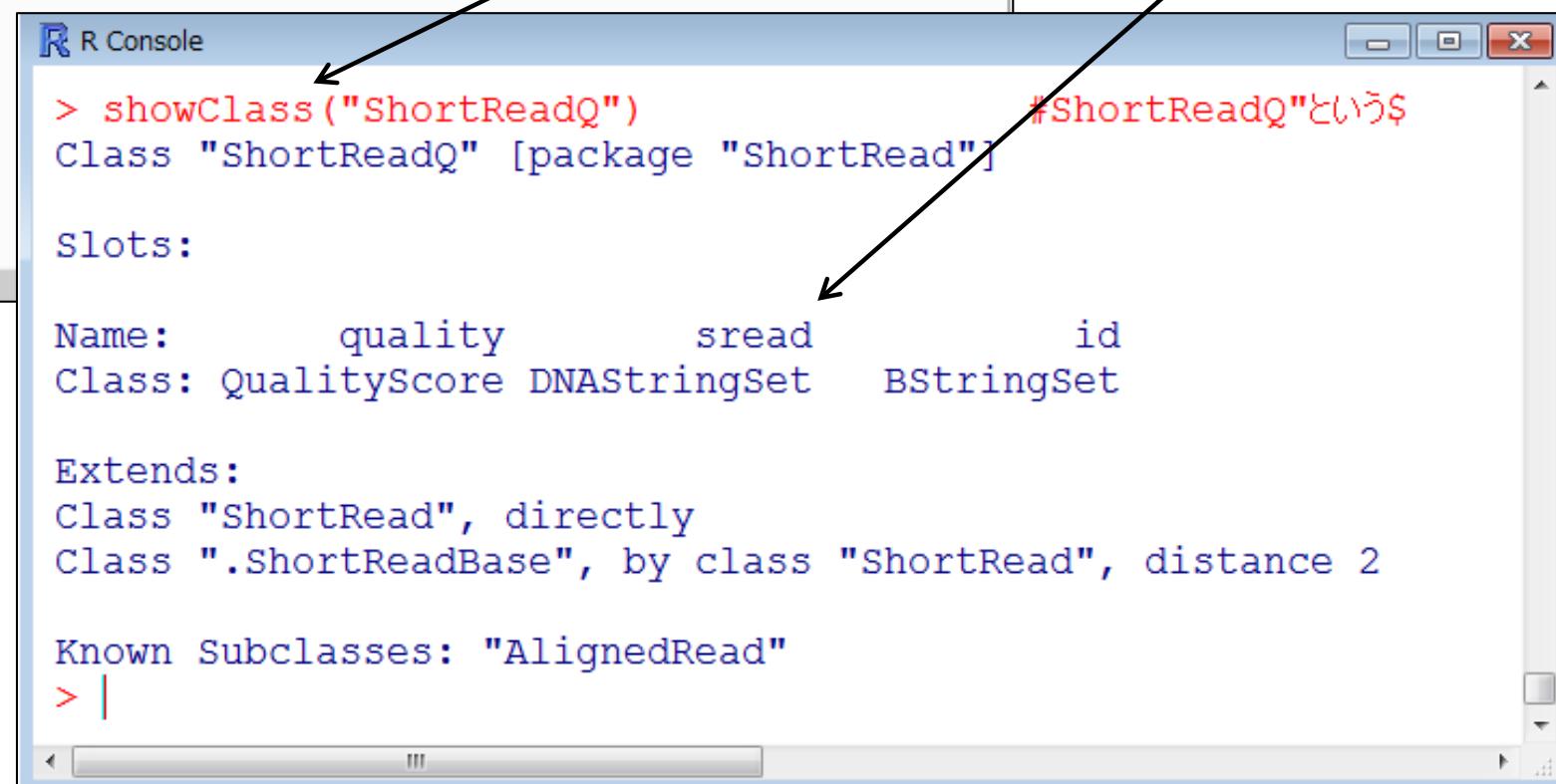
SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです
(Bullard et al., BMC Bioinformatics, 2010)。

・ イントロ | NGS | 読み込み | [FASTQ形式](#)

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  
  
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)。

・ イントロ | NGS | 読み込み | [FASTQ形式](#)

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読み込む

#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)  
fastq
```

#in_fで指定したファイルの読み込み

#確認しているだけです

```
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 NNNNNNNNNNNNNNNCTACCCCCCCCAGCCGCCGCA
[2]      35 NNNNNNNNNNNNNNNNAGACAGTTGATTAGCATAG
[3]      35 NNNNNNNNNNNNNNNNGGTGGGCCTTGTCTTG
[4]      35 NNNNNNNNNNNNNNNNCCCCGCCCCGCCCTCCCTC
[5]      35 NNNNNNNNNNNNNNNNGTCGCCCCGACGCCACA
...
[496]    35 CTGGACGGCCCCCCCCCACACACCCACCCCCCCC
[497]    35 TGTCACTTGTGCTTGCTCTGTCCCACGGGGCTT
[498]    35 CCGCCCTTTTCCAGAAATTTCCGCACAAAAAAA
[499]    35 CGTTCTTGTGCCCCCGGGGGGGGGGGAAAAACC
[500]    35 GGAGCCTCCCCCCCCCCCCAAGGGGGGGGGGGGC
> |
```

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」で解析しな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

4. サンプルデータの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 <-  
out_f <-  
  
#必要なパ  
library()  
  
#入力ファ  
fasta <-  
fasta  
  
#ファイル  
writeXSt
```

```

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"
out_f <- "hoge4.fasta"
```

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

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

```
#入力ファイル名を指定してin_fに
#出力ファイル名を指定してout_fに
#パッケージの読み込み
```

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 window showing the output of the sread function:

```
> sread(fastq)
A DNAStringSet instance of length 500
      width seq
[1]      35 NNNNNNNNNNNNNNNCTACCCCCCCCAGCCGCCGCA
[2]      35 NNNNNNNNNNNNNNNNAGACAGTTGATTAGCATAG
[3]      35 NNNNNNNNNNNNNNNNGGTGGGCCTTGTCTTG
[4]      35 NNNNNNNNNNNNNNNNCCCCGCCCCGCCCTCCCTC
[5]      35 NNNNNNNNNNNNNNNNGTCGCCCCCGACGCCACA
...
[496]     ... ...
[497]      35 CTGGACGGCCCCCCCCCACACACCCACCCCCCCC
[498]      35 TGTCACTTGTGCTTGCTCTGTCCCACGGGGCTT
[499]      35 CCGCCCTTTCCAGAAATTTCCGCACAAAAAAA
[500]      35 CGTTCTTGTGCCCCCGGGGGGGGGGGAAAAACC
> |
```

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"
out_f <- "hoge4.fasta"
```

#入力ファイル名を指定してin_fに
#出力ファイル名を指定してout_fに

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

```
#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。

The screenshot shows an R console window titled 'R Console'. The code entered is:

```
> fasta <- sread(fastq)
> fasta
A DNAStringSet instance of length 500
  width seq
[1] 35 NNNNNNNNNNNNNNNNNCTACCCCCCCCAGCCGCCGCA
[2] 35 NNNNNNNNNNNNNNNNAGACAGTTGATTAGCATAG
[3] 35 NNNNNNNNNNNNNNNNGGTGGGCCTTGTTCCTTG
[4] 35 NNNNNNNNNNNNNNNNCCCCGCCCCGCCCTCCCTC
[5] 35 NNNNNNNNNNNNNNNNGTCGCCCCGACGCCACA
...
[496] 35 CTGGACGGCCCCCCCCCACACACCCACCCCCCCC
[497] 35 TGTCACTTGTGCTTGCTCTGTCCCACGGGGCTT
[498] 35 CCGCCCTTTTCCAGAAATTTCGCACAAAAAAA
[499] 35 CGTTCTTGTGCCCCGGGGCGGGGGGGAAAAACC
[500] 35 GGAGCCTCCCCCCCCCAAGGGGGGGGGGGGC
> |
```

The output shows 500 DNA sequences, each starting with a length indicator (e.g., [1] 35) followed by the sequence itself. The last line shows the prompt '> |'.

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"
out_f <- "hoge4.fasta"
```

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

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

#入力ファイル名を指定してin_fに
#出力ファイル名を指定してout_fに

#パッケージの読み込み

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。

Name:	quality	sread	id
Class:	QualityScore	DNAStringSet	BStringSet

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"
out_f <- "hoge4.fasta"
```

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

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

#入力ファイル名を指定してin_fに
#出力ファイル名を指定してout_fに

#パッケージの読み込み

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 window showing the output of the R command `> id(fastq)`. The output is a BStringSet instance of length 500, containing 500 entries. Each entry consists of a width column (either 46 or 47) and a seq column. The seq column contains identifiers like SRR037439.1 through SRR037439.500, followed by sequencing information (HWI-E4_6...) and a length of 35. Arrows point from the red annotations in the code block above to the R command and its output.

```

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)

#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_
```

```
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

```
@SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35
NNNNNNNNNNNNNNNCTACCCCCCCCAGCCCGCCGCA
+SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35
!!!!!!!!!!!!!".....#"
@SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35
NNNNNNNNNNNNNNNAGACAGTTGATTAGCATAG
+SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35
!!!!!!!!!!!!!".....&
@SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35
NNNNNNNNNNNNNNNGGGTGGGCCGTTGTTCTTG
+SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35
!!!!!!!!!!!!!"5$%%"....."
```

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"
out_f <- "hoge4.fasta"
```

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

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

#入力ファイル名を指定してin_fに
#出力ファイル名を指定してout_fに

#パッケージの読み込み

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)
> fasta
A DNAStringSet instance of length 500
  width seq
[1] 35 NNNNNNNNNNNNN...CCAGCCGCCGCA SRR037439.1 HWI-E...
[2] 35 NNNNNNNNNNNNN...GATTAGCATAG SRR037439.2 HWI-E...
[3] 35 NNNNNNNNNNNNN...CGTTTGTCTTG SRR037439.3 HWI-E...
[4] 35 NNNNNNNNNNNNN...CGCCCCTCCCTC SRR037439.4 HWI-E...
[5] 35 NNNNNNNNNNNNN...CCGACGCCACA SRR037439.5 HWI-E...
...
[496] 35 CTGGACGGCCCC...CCCACCCCCCCC SRR037439.496 HWI...
[497] 35 TGTCACTTGTGCT...CCCACGGGGCTT SRR037439.497 HWI...
[498] 35 CCGCCCTTTTCC...CGCACAAAAAAA SRR037439.498 HWI...
[499] 35 CGTTCTTGTGCC...GGGGGAAAAACC SRR037439.499 HWI...
[500] 35 GGAGCCTCCCCC...GGGGGGGGGGC 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"
out_f <- "hoge4.fasta"
```

#入力ファイル名を指定してin_fに格納
#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↓
NNNNNNNNNNNNNAGACAGTTGATTAGCATAG↓
+SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35↓
!!!!!!!"+"&↓
@SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35↓
NNNNNNNNNNNNNGGTGGGGCGTTGTTCTTG↓
+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↓
NNNNNNNNNNNNNCCCCGCCCGCCCCCTCCCTC↓
+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↓
NNNNNNNNNNNNNGTCGCCCCCGACGCCACA↓
+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↓
NNNNNNNNNNNNNAGACAGTTGATTAGCATAG↓
>SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35↓
NNNNNNNNNNNNNGGTGGGGCGTTGTTCTTG↓
>SRR037439.4 HWI-E4_6_30ACL:2:1:0:1349 length=35↓
NNNNNNNNNNNNNCCCCGCCCGCCCCCTCCCTC↓
>SRR037439.5 HWI-E4_6_30ACL:2:1:0:1669 length=35↓
NNNNNNNNNNNNNGTCGCCCCCGACGCCACA↓
>SRR037439.6 HWI-E4_6_30ACL:2:1:0:1719 length=35↓
NNNNNNNNNNNNNACACGTCCCACCCCCCGCC↓
>SRR037439.7 HWI-E4_6_30ACL:2:1:0:1942 length=35↓
NNNNNNNNNNNNNACATATCTGACCCCTGTGCCC↓
```

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み
                                                     #確認してるだけです
```

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

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

R Console

```
> fasta <- readDNAStringSet(in_f, format="fastq") #in_fで指定する
> fasta                                         #確認してるだけです
A DNAStringSet instance of length 500
  width seq
[1]   35 NNNNNNNNNNNNN...CCAGCCGCCGCA SRR037439.1 HWI-E...
[2]   35 NNNNNNNNNNNNN...GATTTAGCATAG SRR037439.2 HWI-E...
[3]   35 NNNNNNNNNNNNN...CGTTTGTCTTG SRR037439.3 HWI-E...
[4]   35 NNNNNNNNNNNNN...CGCCCCTCCCTC SRR037439.4 HWI-E...
[5]   35 NNNNNNNNNNNNN...CCGACGCCACA SRR037439.5 HWI-E...
...
[496]   35 CTGGACGGCCCC...CCCACCCCCCCC SRR037439.496 HWI...
[497]   35 TGTCACTTGTGCT...CCCACGGGGCTT SRR037439.497 HWI...
[498]   35 CCGCCCTTTTCC...CGCACAAAAAAA SRR037439.498 HWI...
[499]   35 CGTTCTTGTGCC...GGGGGAAAAACC SRR037439.499 HWI...
[500]   35 GGAGCCTCCCCCCC...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)

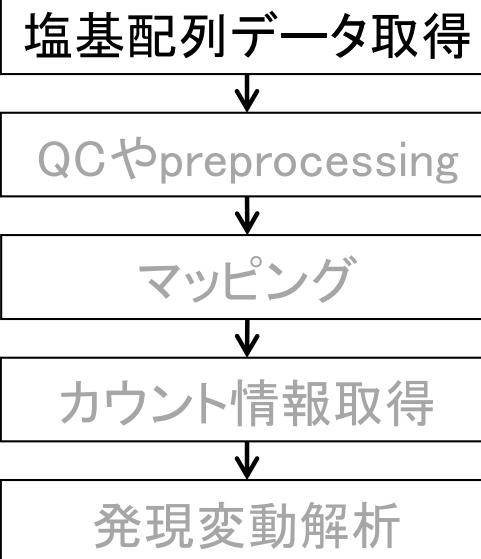


NGSデータ解析とR

(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)
- What • イントロ | 一般 | 配列取得 | ゲノム配列 | [BSgenome](#) (last modified 2014/06/28) NEW
- このすのすの | 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)
- 201201 | イントロ | 一般 | 配列取得 | プロモーター配列 | [BSgenome](#) (last modified 2014/04/25)
- 201201 | イントロ | 一般 | 配列取得 | プロモーター配列 | [GenomicFeatures\(Lawrence 2013\)](#) (last modified 2014/04/25)
- 201201 | 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)
- 門田 | 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 2014/06/11)
- イントロ | NGS | [様々なプラットフォーム](#) (last modified 2014/06/10)
- マツニッした | 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
- 201201 東方申請 | イントロ | NGS | 配列取得 | FASTQ or SRALite | [公共DBから](#) (last modified 2014/06/28) NEW
- イントロ | NGS | 配列取得 | FASTQ or SRALite | [SRAdb\(Zhu 2013\)](#) (last modified 2014/06/11)
- 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 modified 2014/06/25)
- 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\(Lawrence 2013\)](#) (last modified 2014/03/28)
- 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 modified 2014/03/28)



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

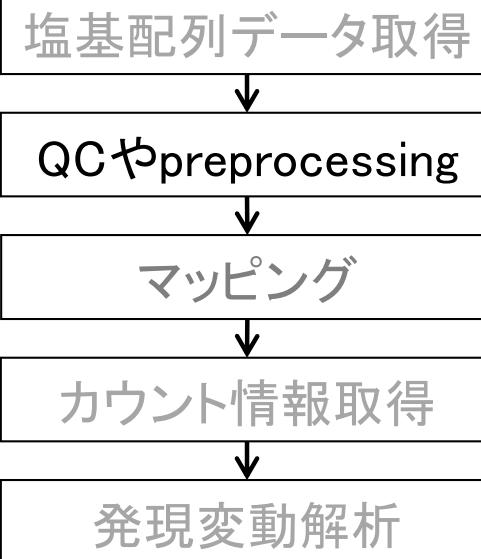


NGSデータ解析とR

(Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランскри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)

- What
- 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)
 - このすのすの | イントロ | NGS | 読み込み | FASTQ形式 (last modified 2014/06/15)
 - 201 | 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)
 - 201 | イントロ | NGS | 読み込み | Illuminaの * seq.txt (last modified 2013/06/13)
 - 201 | イントロ | NGS | 読み込み | Illuminaの * qseq.txt (last modified 2013/06/17)
 - 門田 | 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
 - マッニッシャー | 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)
 - 201 | イントロ | ファイル形式の変換 | qseq --> FASTA (last modified 2013/06/17)
 - 東方 | イントロ | ファイル形式の変換 | qseq --> Illumina FASTQ (last modified 2013/06/17)
 - 申込 | イントロ | ファイル形式の変換 | qseq --> Sanger FASTQ (last modified 2013/08/19)
 - 参考 | 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 | grqc (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скриプトーム、正規化、発現変動、統計、モデル、バーコード
(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 2014/06/15)
 - イントロ | NGS | 読み込み | FASTA形式 | description行の記述を整
 - このすのこの * seq.txt (last modified 2014/06/15)
 - イントロ | NGS | 読み込み | FASTQ形式 | description行の記述を整
 - 201 イントロ | NGS | 読み込み | Illuminaの * seq.txt (last modified 2013/06/15)
 - 201 イントロ | NGS | 読み込み | Illuminaの * qseq.txt (last modified 2013/06/15)
 - 門脇 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 2014/06/09)
 - マツニツ 二ヶ所で イントロ | ファイル形式の変換 | FASTQ --> FASTA (last modified 2014/06/09)
 - した イントロ | ファイル形式の変換 | Genbank --> FASTA (last modified 2014/06/09)
 - 201 東京申込用紙 イントロ | ファイル形式の変換 | qseq --> FASTA (last modified 2013/06/15)
 - イントロ | ファイル形式の変換 | qseq --> Illumina FASTQ (last modified 2014/06/09)
 - イントロ | ファイル形式の変換 | qseq --> Sanger FASTQ (last modified 2014/06/09)
 - 参考 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 2014/06/11)
 - 前処理 | クオリティチェック | PHREDスコアに変換 (last modified 2014/06/11)
 - 前処理 | クオリティチェック | 配列長分布を調べる (last modified 2014/06/11)

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

- [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)

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
```

FastQCと似たようなQCLレポートを得ることができます

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。

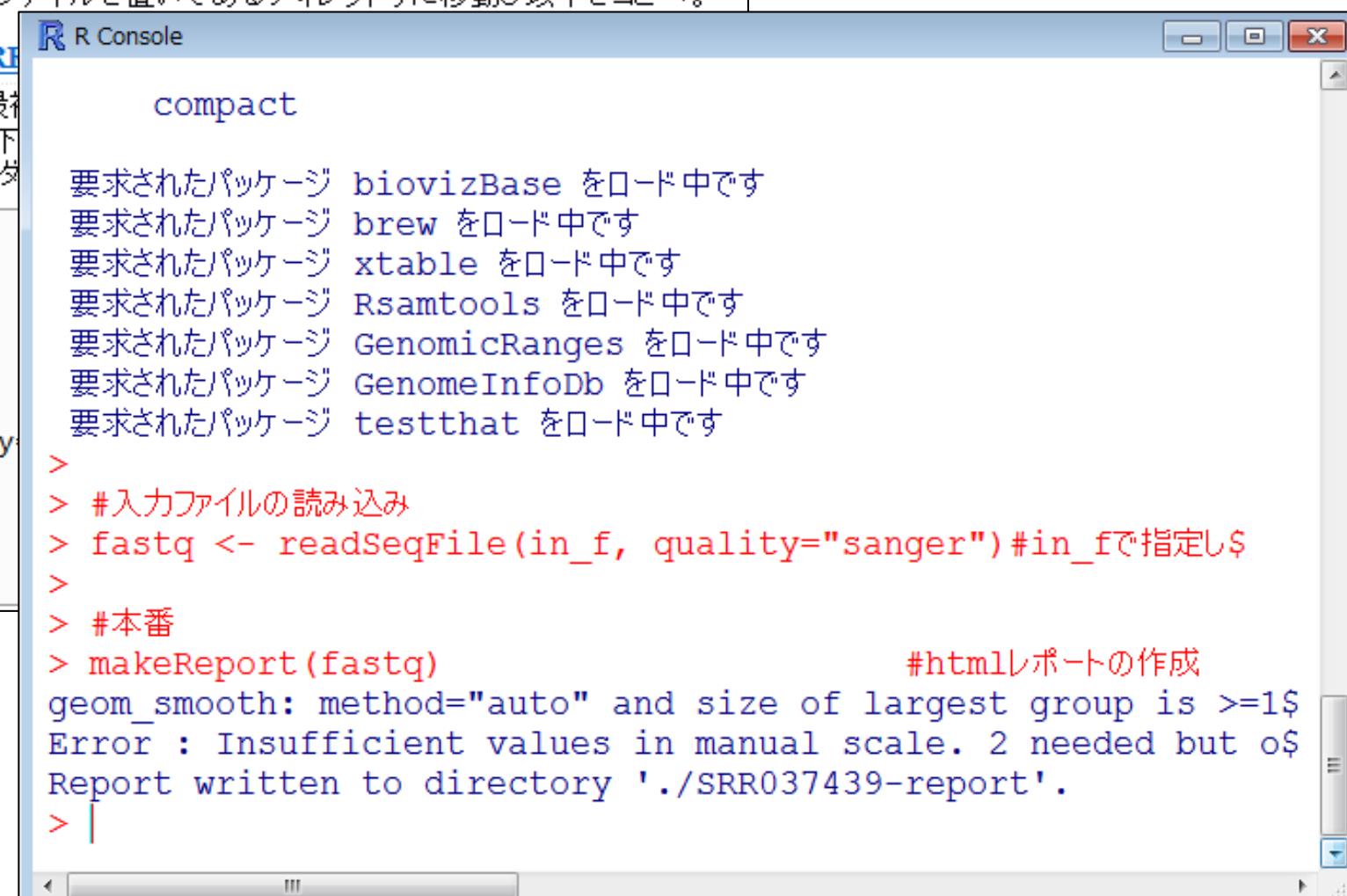
「ファイル」→「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

1. サンプルデータ7のFASTQ形式ファイル(SRR037439)

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をダ

```
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="sanger")
#本番
makeReport(fastq)
```

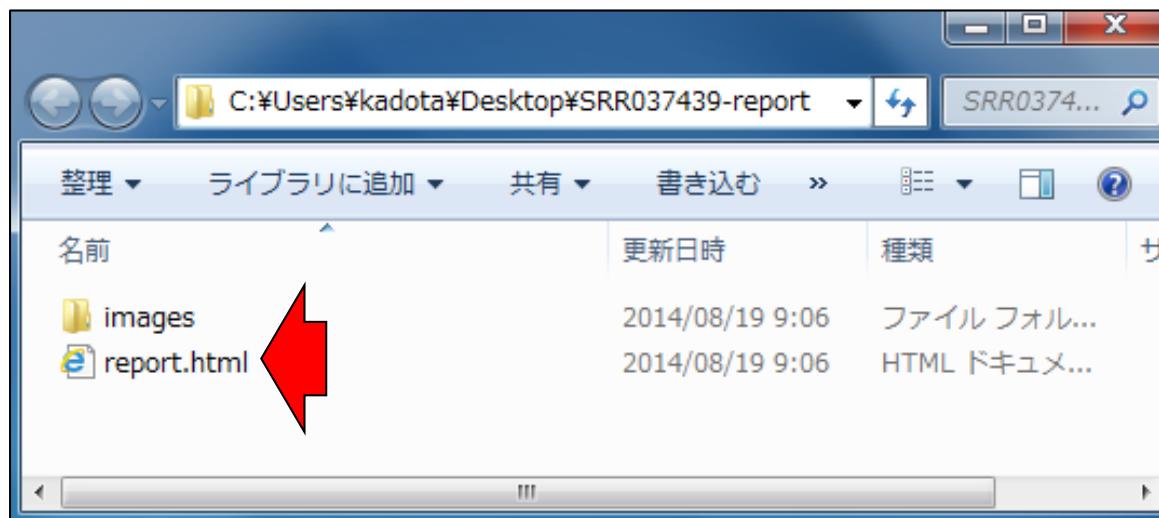
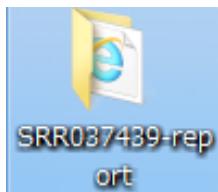
エラーメッセージは出ている
が、一応最後までできている



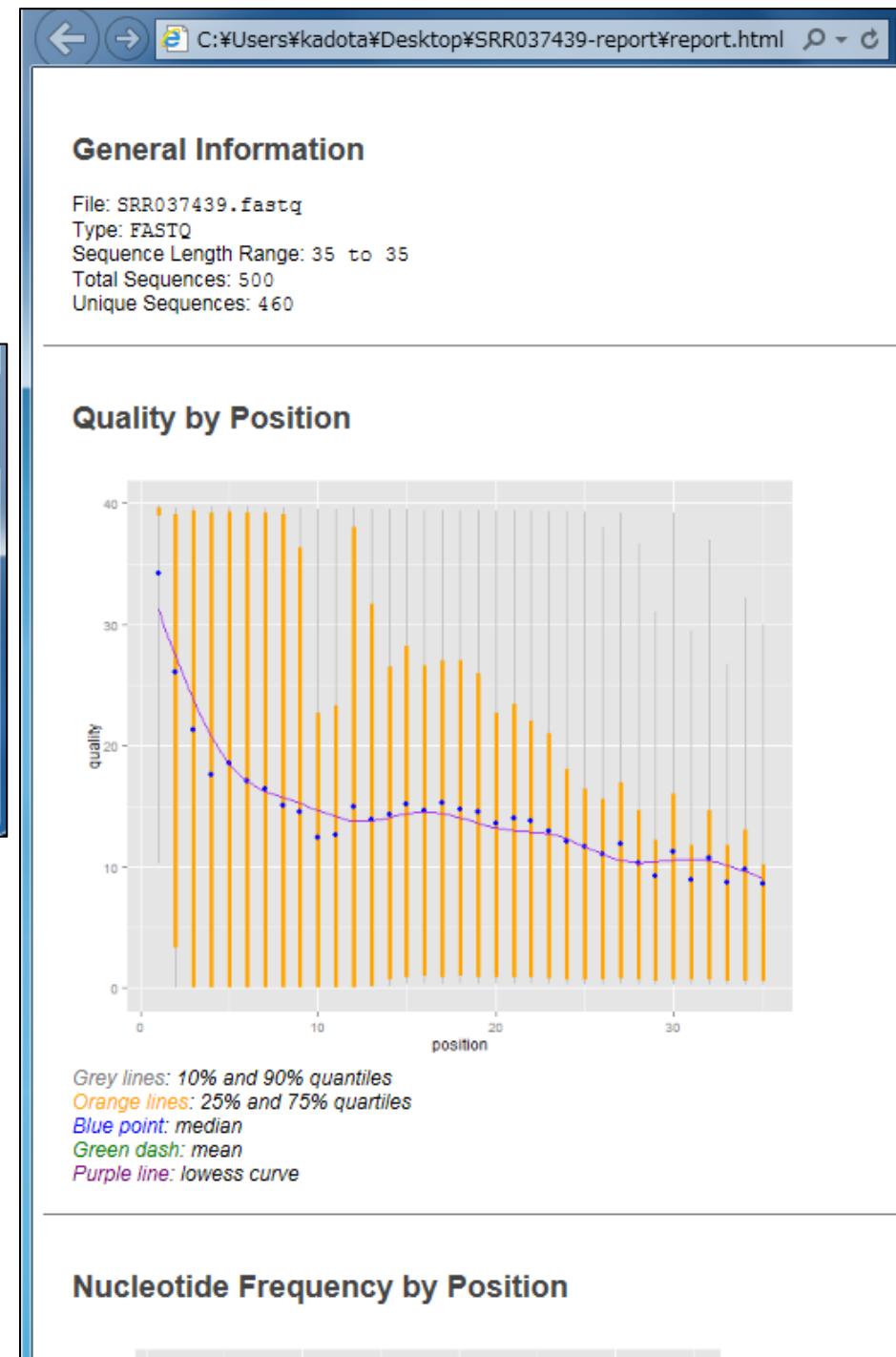
```
R Console
compact

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で指定します
>
> #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、filtrating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 | [qrc](#) (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以外のみの配列](#) (last modified 2012/06/10)
- ・ 前処理 | フィルタリング | [ACGT](#)
- ・ 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 | [指定した](#)
- ・ 前処理 | フィルタリング | [任意](#)
- ・ 前処理 | フィルタリング | [指定した](#)
- ・ 前処理 | フィルタリング | [アセンブル](#) | [について](#) (last n)
- ・ アセンブル | [ゲノム用](#) (last n)
- ・ アセンブル | [トランスクри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、データ

```

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。

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!

```

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)

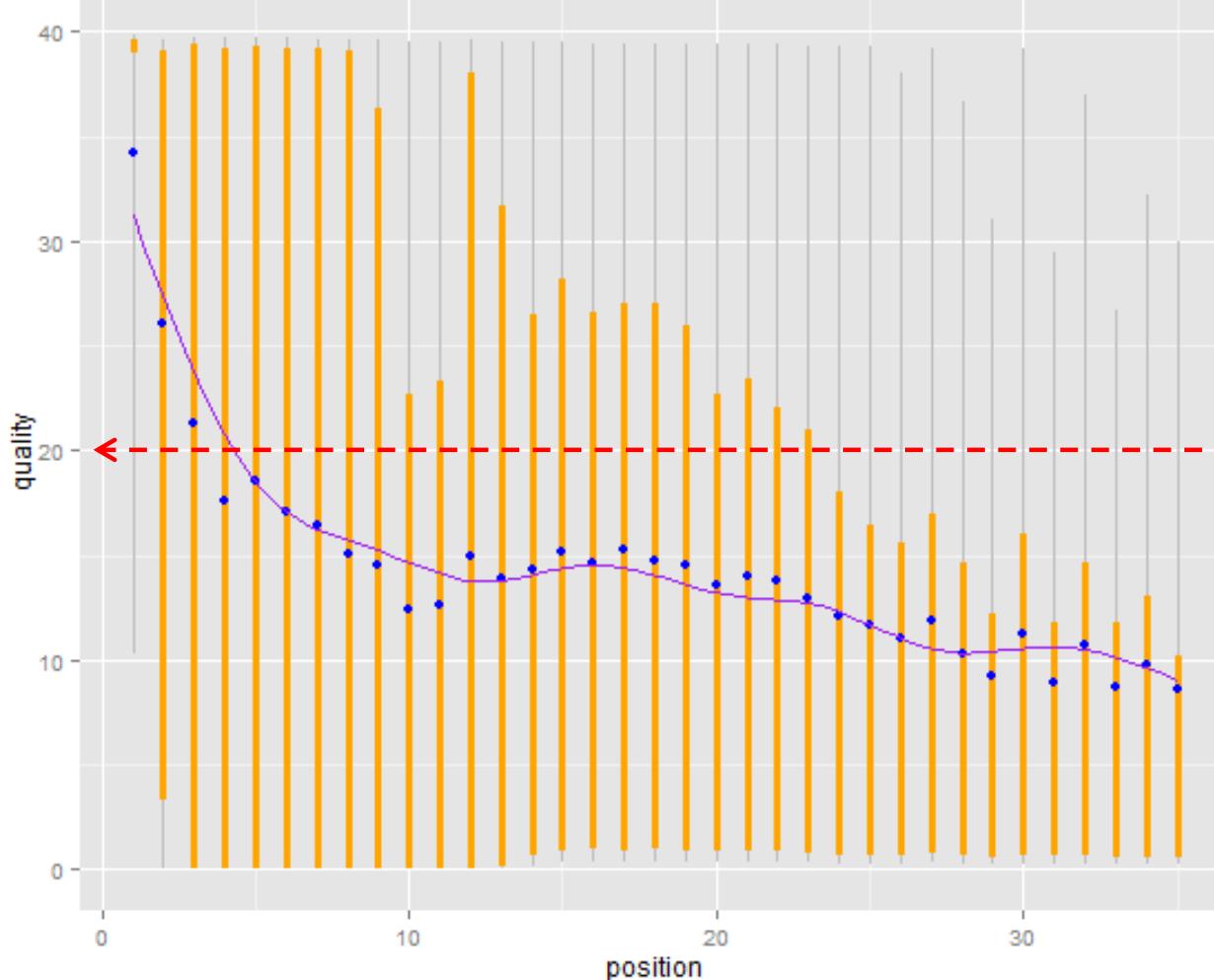
#入力ファイルの読み込み
fastq <- readFastq(in_f)
sread(fastq)

#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)

```

```
R R Console [499] 35 CGTTCTTGTGCCCCCGGGGCGGGGGGAAAAAACC  
[500] 35 GGAGCCTCCCCCCCCCAAGGGGGGGGGGGGC  
>  
> #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 TTTTACGTATGCATATATATGATTATTCTTTGG  
[3] 35 GAACCATGTGCTAATTTTCACAATTATTGACCTG  
[4] 35 GATAGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG  
[5] 35 CGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG  
>  
> #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
CCCCCCCCCAAGGGGGGGGGGGGGGC

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 GGGACGGGGGGGGGGGGGGGGGCCGGGGGG
[2] 35 TTTTACGTATGCATATATATGATTATTCTTTGG
[3] 35 GAACCATGTGCTAATTTCACAATTATTGACCTG
[4] 35 GATAGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
[5] 35 CGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
>
> #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 | [arqc](#) (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ск립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 2014/03/17)
- 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 2014/03/17)
- 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 2013/06/18)
- 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 <- "hoge"
param <- 3
#必要なパッケージをロード
library(Biostrings)
#入力ファイルを読み込む
fasta <- readFastq(in_f)
#本番
obj <- 1:param
fasta <- fastq[1:param]
#ファイルに保存
writeXStringS(fasta, out_f)
```

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"
out_f <- "hoge10.fastq"
param_range <- 5:20

#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
obj <- param_range
fastq <- fastq[obj]
sread(fastq)

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

#入力ファイル名を指定してin_fに格納
#出力ファイル名を指定してout_fに格納
#抽出したいリードの範囲を指定

#パッケージ
#in_fで指定
#確認している
#抽出したい
#objがTRUE
#確認している

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
> fastq <- fastq[obj]
> sread(fastq)
A DNAStringSet instance of length 16
  width seq
[1] 35 NNNNNNNNNNNNNNNNGTCGCCCGACGCCACA
[2] 35 NNNNNNNNNNNNNNNNACACGTCCCACCCCCCGCCC
[3] 35 NNNNNNNNNNNNNNNNACATATCTGACCCTGTGCCG
[4] 35 NNNNNNNNNNNNNNNNGTGGGCCCTAACAGGACCTG
[5] 35 NNNNNNNNNNNNNNNNACTCAACCCTCCCCGCCCA
...
[12] 35 NNNNNNNNNNNNNNNNACTGGCGCGACTGTACAGC
[13] 35 NNNNNNNNNNNNNNTTTCTTTACACAGTGTGTTT
[14] 35 NNNNNNNNNNNNNNGGGGGGGGGGGGGGGGGGGGGG
[15] 35 GNNNNNNNNNNNNNGGGGGGGCGAGGGGGGGCGGG
[16] 35 CNNNNNNNNNNNNNCCCCGTCTACGCACCCCGAC

>
> #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↓
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↓
NNNNNNNNNNNNNAGACAGTTGATTAGCATAG↓
+SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35↓
!!!!!!"+&"↓
@SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35↓
NNNNNNNNNNNNNGGTGGGCGTTGTTCTTG↓
+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↓
NNNNNNNNNNNNNCCCCGCCCGCCCCCTCCCTC↓
+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↓
NNNNNNNNNNNNNGTCGCCCCCGACGCCACA↓
+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↓
NNNNNNNNNNNNNGTCGCCCCCGACGCCACA↓
+↓
!!!!!!"#"&"↓
@SRR037439.6 HWI-E4_6_30ACL:2:1:0:1719 length=35↓
NNNNNNNNNNNNACACGTCCCACCCCCCGCCC↓
+↓
!!!!!!"2%"&%*-*%"↓
@SRR037439.7 HWI-E4_6_30ACL:2:1:0:1942 length=35↓
NNNNNNNNNNNNACATATCTGACCCTGTGCCG↓
+↓
!!!!!!&.""/"+")'("%"$)"↓
@SRR037439.8 HWI-E4_6_30ACL:2:1:0:2001 length=35↓
NNNNNNNNNNNNNGGTGGGCCCTAAGGACCTG↓
+↓
!!!!!!IH8#"&"↓
@SRR037439.9 HWI-E4_6_30ACL:2:1:0:2032 length=35↓
NNNNNNNNNNNNACTCAACCTTCCCCCCCCA↓
```

フィルタリング: サブセット作成

11. サンプルデータの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

paramで指定したリード数をランダム抽出するやり方です

```
in_f <- "SRR037439.fastq"
out_f <- "hoge11.fastq"
param <- 30 #ランダム抽出

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

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

R Console

```
> set.seed(1010) #おまじない
> obj <- sample(1:length(fastq), param, replace=F) #objで指定
> fastq <- fastq[sort(obj)] #確認してます
> sread(fastq)
A DNAStringSet instance of length 30
  width seq
[1] 35 NNNNNNNNNNNNNNNNAGACAGTTGATTAGCATAG
[2] 35 NNNNNNNNNNNNNNNNGTCGGCCCCGACGCCACA
[3] 35 TNNNNNNNNNNNNNTTTTATTTTTTTTTTTTT
[4] 35 CNNNNNNNNNNNNNTGGCGTTGCTGTGGCGGGCGCG
[5] 35 CNNNNNNNNNNNNCCCAAAAAAAACAAAAAAAC
...
[26] 35 TGGGACGGACCCCCCCCACAAAAAAACAGAGAGA
[27] 35 CATTCAAGGGCAAGAACTTTTTGGGGGGGGGGGG
[28] 35 CAAAGGTTGGCGTGCGCTGAGACAATATTTTTGGT
[29] 35 CGGATCTTTTTTTGTTTCTCCCTTTCCCCC
[30] 35 GTCAAAGATGAGGGGGGGTTGGGGGGGGGAC
>
> #ファイルに保存
> writeFastq(fastq, out_f, compress=F) #fastqの中身
> |
```

フィルタリング: サブセット作成

11. サンプルデータの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る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"
out_f <- "hoge11.fastq"
param <- 30
#入力ファイル名を指定してin_fに格納
#出力ファイル名を指定してout_fに格納
#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↓
NNNNNNNNNNNNNCTACCCCCCCCAGCCGCCGCA↓
+SRR037439.1 HWI-E4_6_30ACL:2:1:0:176 length=35↓
!!!!!!"#"↓
@SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35↓
NNNNNNNNNNNNNAGACAGTTGATTAGCATAG↓
+SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35↓
!!!!!!"+&"↓
@SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35↓
NNNNNNNNNNNNNGGTGGGCGTTGTTCTTG↓
+SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35↓
!!!!!!"/$%&%↓
@SRR037439.4 HWI-E4_6_30ACL:2:1:0:1349 length=35↓
NNNNNNNNNNNNNCCCCGCCCGCCCCCTCCCTC↓
+SRR037439.4 HWI-E4_6_30ACL:2:1:0:1349 length=35↓
!!!!!!"0"("&$"↓
@SRR037439.5 HWI-E4_6_30ACL:2:1:0:1669 length=35↓
NNNNNNNNNNNNNGTCGCCCCCGACGCCACA↓
+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↓
NNNNNNNNNNNNNAGACAGTTGATTAGCATAG↓
+↓
!!!!!!"!"+&"↓
@SRR037439.5 HWI-E4_6_30ACL:2:1:0:1669 length=35↓
NNNNNNNNNNNNNGTCGCCCCCGACGCCACA↓
+↓
!!!!!!"!","#"&"&"↓
@SRR037439.28 HWI-E4_6_30ACL:2:1:1:799 length=35↓
TNNNNNNNNNNNNNTTTATTTTTTTTTTT↓
+↓
!!!!!!"#$"/"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↓
CNNNNNNNNNNNNCCCCAAGAACAAAAAAC↓
```

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

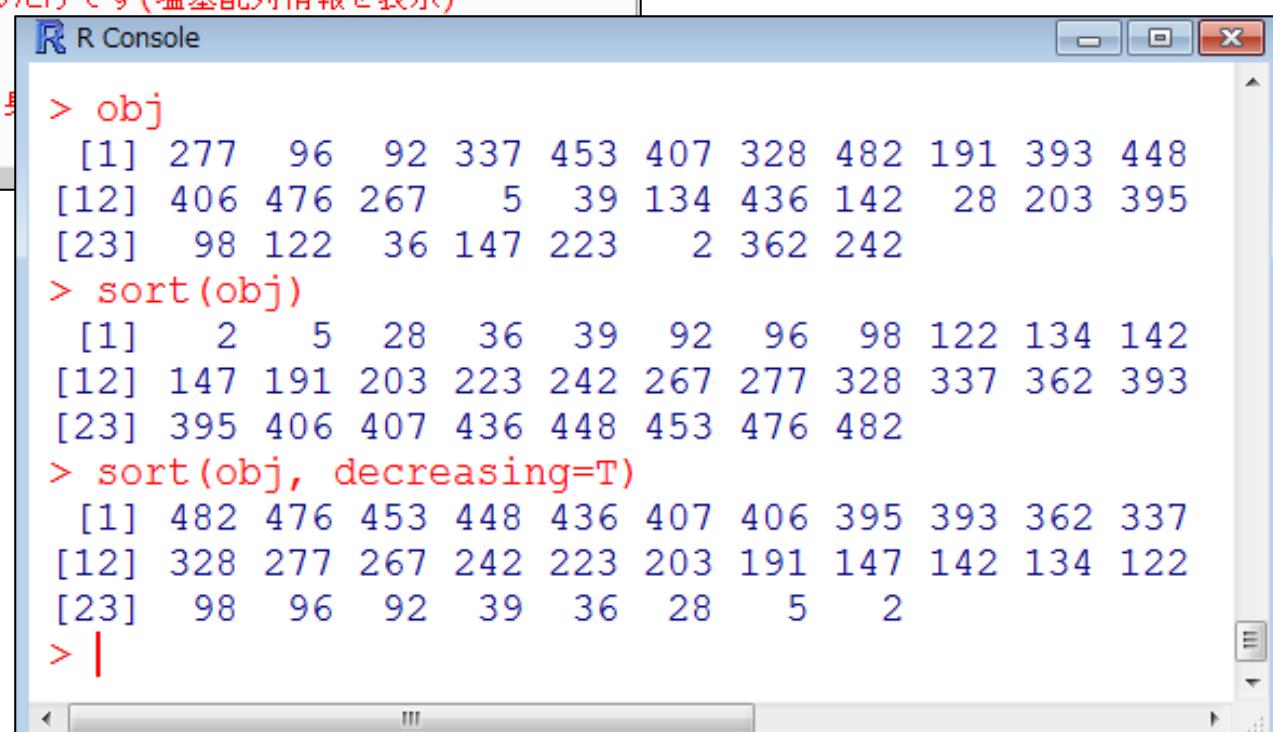
#必要なパッケージをロード
library(ShortRead)                  #パッケージの読み込み

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

#本番
set.seed(1010)
obj <- sample(1:length(fastq), param, replace=F) #おまじない(同じ乱数になるようにするため)
fastq <- fastq[sort(obj)]             #in_fで指定したファイルの読み込み
                                         #確認してるだけです(塩基配列情報を表示)
                                         #objで指定したリードのみソートして抽出した結果をfa
                                         #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
```

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の中身を出力

```

(説明のため)黒枠部分を一旦コピペ

R Console

```

[3]      35 NNNNNNNNNNNNNNNNNNGGTGGGGCGTTGTTCTTG
[4]      35 NNNNNNNNNNNNNNNNCCCCGCCCGCCCCTCCCTC
[5]      35 NNNNNNNNNNNNNNNNGTCGCCCGACGCCACA
...
[496]    35 CTGGACGGCCCCCCCCCACACACACCACCCCCCCC
[497]    35 TGTCACTTGTGCTTGCTCTGTCCCACGGGGCTT
[498]    35 CCGCCCTTTCCAGAAATTTCAGCACAAAAAAA
[499]    35 CGTTCTTGTGCCCCCGGGGCAGGGGGAAAAAAC
[500]    35 GGAGCCTCCCCCCCCCAAGGGGGGGGGGGC

>
> #本番
> set.seed(1010)                      #おまじない
> |

```

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)
sread(fastq)

#本番
set.seed(1010)
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で指定している
sread(fastq)

#ファイルに保存
writeFastq(fastq, out_f, compress=F) #fastqの中身を確認する
  
```

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)                      #おまじない
> 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
> 
  
```

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
rcode_20140909.txt

```
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)
sread(fastq)

#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で指定している
sread(fastq)                       #確認している

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

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)                      #おまじなし
> 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
> |
```

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

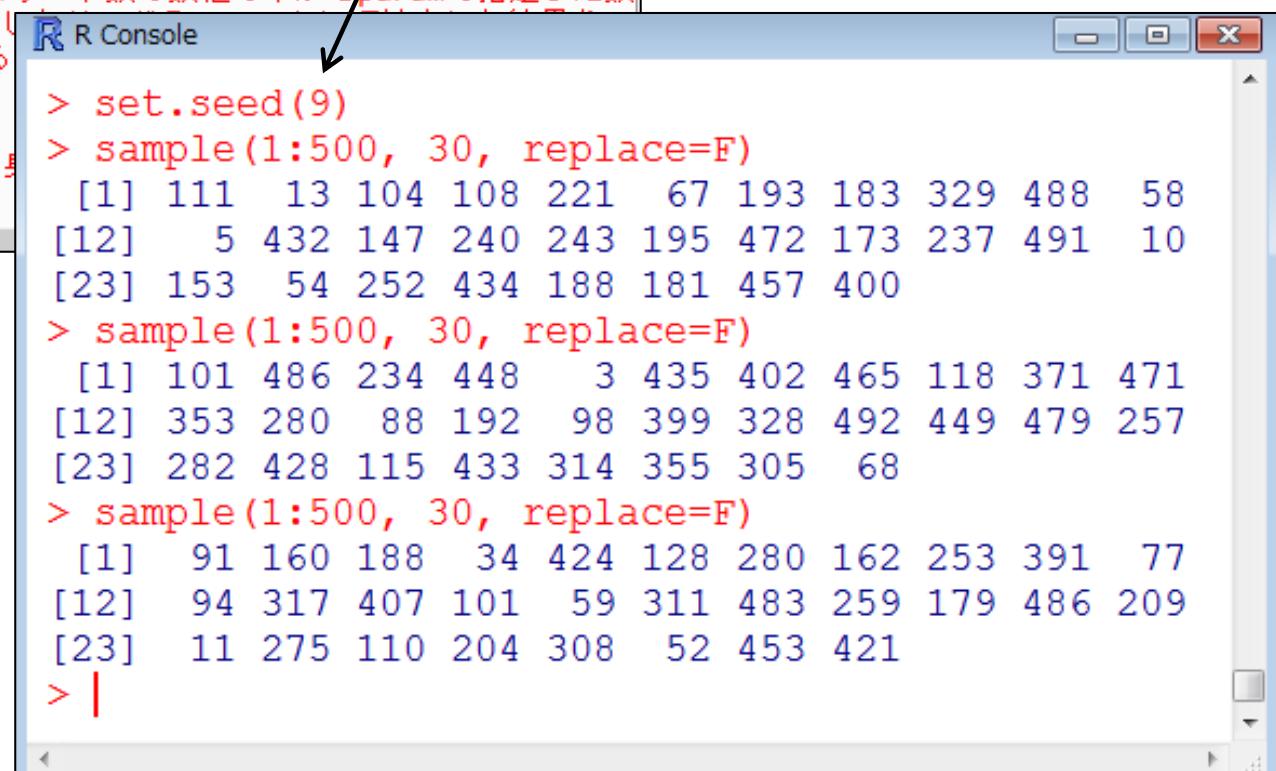
SRR037439から得られるFASTQファイルの最初の2,000行分を抽出したMAQC2 brainデータです ([Bullard et al., 2010](#))。 (全部で500リードからなることが既知という前提で)30リード 分をランダムに非復元抽出するやり方です。 writeFastq関数実行時にcompress=Fとしてgzip圧縮前のファイルを出力しています

処理 | フィルタリング | 任意のリード(サブセット)を抽出

rcode_20140909.txt

```
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で指定している  
sread(fastq)                        #確認してる  
  
#ファイルに保存  
writeFastq(fastq, out_f, compress=F) #fastqの中身を確認
```

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 window showing R code and its output. The code sets a seed of 9 and then runs three calls to sample(1:500, 30, replace=F) to generate three sets of 30 random indices from 1 to 500.

```
> 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
> |
```

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
rcode_20140909.txt

```
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

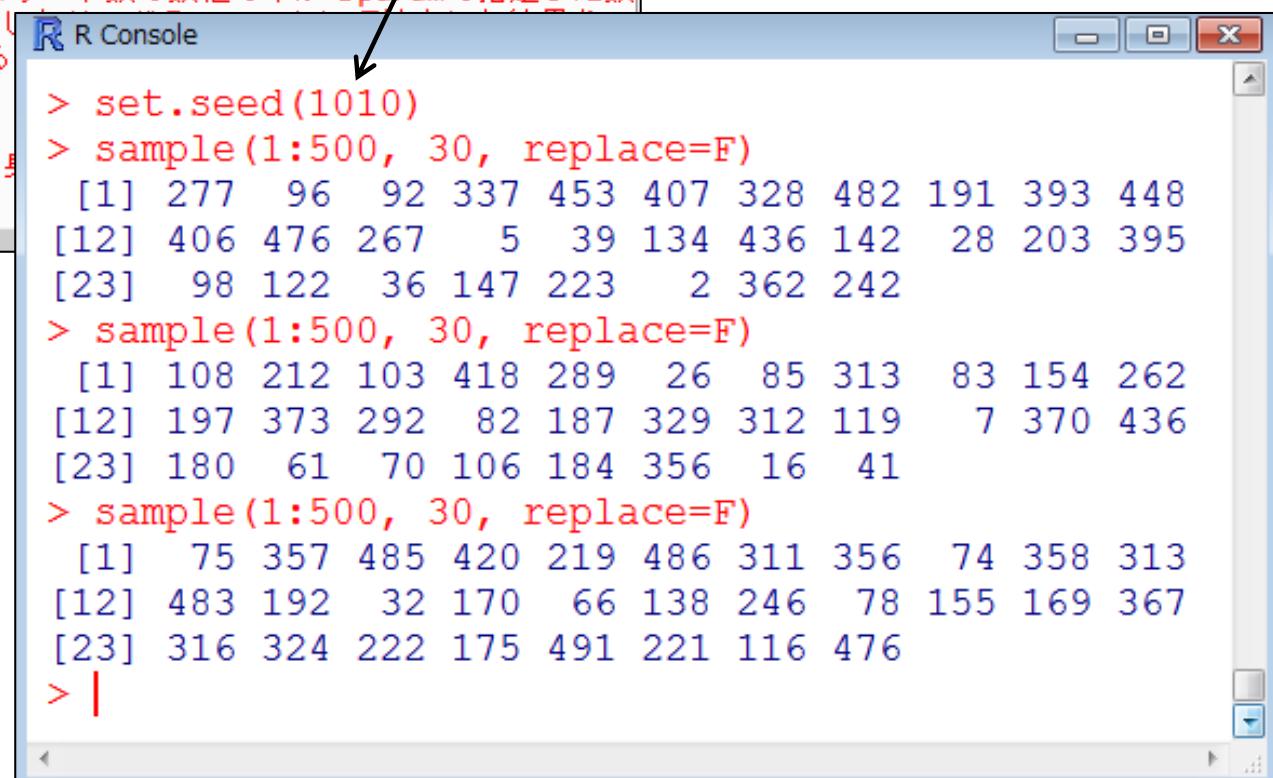
#必要なパッケージをロード
library(ShortRead)                  #パッケージの読み込み

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

#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で指定している順序で並び替える
sread(fastq)

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

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 window showing R code and its execution results. The code uses set.seed(1010) followed by three calls to sample(1:500, 30, replace=F) to generate three different sets of 30 random indices from 1 to 500. The results are displayed as vectors of integers.

```
> 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
> |
```

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

rcode_20140909.txt

```
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)
sread(fastq)

#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で指定した順序でソート
sread(fastq)

#ファイルに保存
writeFastq(fastq, out_f, compress=F) #fastqをgzip圧縮して出力
```

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して、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。

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.fa"
out_f <- "hoge2.fasta"
param <- 3

#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(fasta)
fasta

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

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

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"
out_f <- "hoge2.fasta"
param_range <- 2:4

#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
fasta <- fasta[obj]
fasta

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

#ファイルに保存
writeXStringSet(fasta, file=out_f, format="fasta") #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成する「ファイル」-「抽出」メニューへ亦重ねます。

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

1. multi-FASTQ

イントロ | 一般

```
in_f <- "hoge8.fastq"
out_f <- "hoge8.fastq.gz"
param <- 3

#必要なパッケージをロード
library(Bio::SeqIO)

#入力ファイルを読み込み
fasta <- readFastq(in_f)
id(fasta)

#本番
obj <- 1:length(fasta)
fastq <- sort(obj)
id(fastq)

#ファイルに保存
writeXString(fasta, out_f)
```

[イントロ](#) | [NGS](#) | [配列取得](#) | [FASTQ or SRALite](#) | [SRAdb\(Zhu, 2013\)](#)

タ(Nie et al., BMC Genomics, 2013)です。入力ファイルサイズは4GB程度で、gzip圧縮なしで出力すると約16MBになります。MacintoshではうまくいくがWindowsではうまくいかないようです。

```
in_f <- "SRR609266.fastq.gz"
out_f <- "hoge8.fastq"
param <- 100000

#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)
id(fastq)

#本番
set.seed(1010)
obj <- sample(1:length(fastq), param, replace=F)
fastq <- fastq[sort(obj)]
id(fastq)

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

#入力ファイル名を指定してin_fに格納
#出力ファイル名を指定してout_fに格納
#ランダム抽出したいリード数を指定

#パッケージの読み込み

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

#おまじない(同じ乱数になるようにするため)
#リード数の数値の中からparamで指定した数だけ抽出
#objで指定したリードのみソートして抽出した結果をfastqに格納
#確認してるだけです(description情報を表示)

#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 modified 2013/06/13)
- ・ 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 modified 2013/06/13)
- ・ 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](#) (last 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 | [FASTQ](#)
- ・ 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 | [FASTA](#)
- ・ イントロ | ファイル形式の変換 | [GenBank](#)
- ・ 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 | [qseqn](#)
- ・ イントロ | ファイル形式の変換 | [qseqn2](#)
- ・ イントロ | ファイル形式の変換 | [qseqn2](#)
- ・ 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](#) (lambda)
- ・ 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](#) | [FastQC](#)

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。



modified 2013/06/13

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 <- " "
param2 <- 1                         #区切り文字を指定
                                      #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)
id(fastq)                           #in_fで指定したファイルの読み込み
                                      #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 <- " "
param2 <- 1                         #区切り文字を指定
                                    #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
NNNNNNNNNNNNNAGACAGTTGATTAGCATAG
+SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35
!!!!!!"+&
@SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35
NNNNNNNNNNNNNGGTGGGCGTTGTTCTTG
+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
NNNNNNNNNNNNNCCCCGCCCGCCCCCTCCCTC
+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
NNNNNNNNNNNNCTACCCCCCCCAGCCGCCGCA
+
@SRR037439.2
NNNNNNNNNNNNNAGACAGTTGATTAGCATAG
+
@SRR037439.3
NNNNNNNNNNNNNGGTGGGCGTTGTTCTTG
+
@SRR037439.4
NNNNNNNNNNNNNCCCCGCCCGCCCCCTCCCTC
+
@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、

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
NNNNNNNNNNNNNAGACAGTTGATTAGCATAG
+SRR037439.2 HWI-E4_6_30ACL:2:1:0:252 length=35
!!!!!!!
@SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35
NNNNNNNNNNNNNGGTGGGGCGTTGTTCTTG
+SRR037439.3 HWI-E4_6_30ACL:2:1:0:1152 length=35
!!!!!!!
@SRR037439.4 HWI-E4_6_30ACL:2:1:0:1349 length=35
NNNNNNNNNNNNNCCCCGCCCGCCCCCTCCCTC
+SRR037439.4 HWI-E4_6_30ACL:2:1:0:1349 length=35
!!!!!!!
@SRR037439.5 HWI-E4_6_30ACL:2:1:0:1669 length=35
```

出力: [hoge5.fastq](#)

```
@1
NNNNNNNNNNNNNCTACCCCCCCCAGCCGCCGCA
+
@2
NNNNNNNNNNNNNAGACAGTTGATTAGCATAG
+
@3
NNNNNNNNNNNNNGGTGGGGCGTTGTTCTTG
+
@4
NNNNNNNNNNNNNCCCCGCCCGCCCCCTCCCTC
+
@5
```