

# Contents

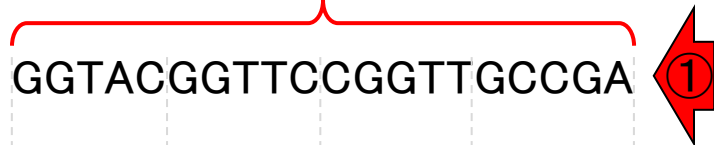
- Introduction、出現頻度解析( $k=2$ )、出現頻度解析( $k=1$ )
- $k=1$ で実践、multi-FASTAファイル、他の例題を実行
- $k=2$ で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8( $k=2$ )、例題9( $k=3$ )、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11( $k=10$ )、例題12( $k=10$ )、シークエンスエラーを含む場合

# Introduction 1

NGS解析で、①のようなL=20塩基からなるリードが得られたとする。

L = 20

GGTACGGTTCGGTTCGCGA



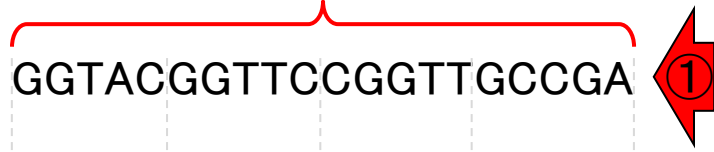
The diagram shows a DNA sequence "GGTACGGTTCGGTTCGCGA" with a red bracket above it indicating a length of L=20. A red arrow with the number 1 inside points to the first base 'G' of the sequence. Below the sequence, there are four vertical dashed lines extending downwards, corresponding to the 4th, 8th, 12th, and 16th positions of the sequence.

# Introduction2

NGS解析で、①のようなL=20塩基からなるリードが得られたとする。k-mer解析は、k塩基の長さからなる部分塩基配列を生成して、出現頻度を調べたりする作業のこと。

L = 20

GGTACGGTTCGGTTGCCGA



The diagram shows a DNA sequence "GGTACGGTTCGGTTGCCGA" with a red bracket above it indicating a length of L=20. A red arrow labeled "1" points to the start of the sequence. Vertical dashed lines are drawn below the sequence, extending downwards.

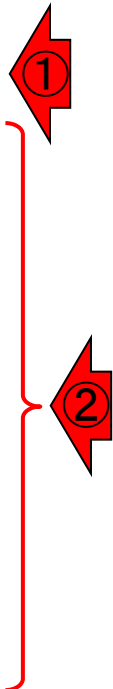
# Introduction3

NGS解析で、①のような $L=20$ 塩基からなるリードが得られたとする。 $k$ -mer解析は、 $k$ 塩基の長さからなる部分塩基配列を生成して、出現頻度を調べたりする作業のこと。 $k$ の値は、 $L$ よりも短い値にします(つまり、 $k < L$ )。例えば $k=6$ の場合は、②15個の $k$ -merを生成可能。

$k=6$ の場合: 15個の $k$ -mer

GGTACGGTTCGGTTGCCGA

GGTACG  
GTACGG  
TACGGT  
ACGGTT  
CGGTTC  
GGTTCC  
GTTCCG  
TTCCGG  
TCCGGT  
CCGGTT  
CGGTTG  
GGTTGC  
GTTGCC  
TTGCCG  
TGCCGA



# Introduction4

同様に、③k=8の場合は、④13個のk-merを生成可能。

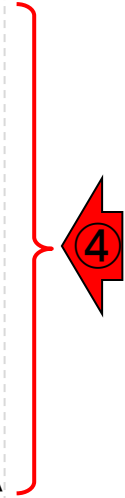
k=6の場合: 15個のk-mer

GGTACGGTTCGGTTGCCGA  
GGTACG  
GTACGG  
TACGGT  
ACGGTT  
CGGTTCC  
GGTTCC  
GTTCCG  
TTCCGG  
TCCGGT  
CCGGTT  
CGGTTG  
GGTTGC  
GTTGCC  
TTGCCG  
TGCCGA



k=8の場合: 13個のk-mer

GGTACGGTTCGGTTGCCGA  
GGTACGGT  
GTACGGTT  
TACGGTTC  
ACGGTTCC  
CGGTTCCG  
GGTTCCGG  
GTTCCGGT  
TTCCGGTT  
TCCGGTTG  
CCGGTTGC  
CGGTTGCC  
GGTTGCCG  
GTTGCCGA



# Introduction5

同様にして、③k=8の場合は、④13個のk-merを生成可能。また、⑤k=10の場合は11個のk-merを生成可能。

k=6の場合: 15個のk-mer

GGTACGGTTCGGTTCGCGA

```
GGTACG
GTACGG
TACGGT
ACGGTT
CGGTTC
GGTTCC
GTTCCG
TTCCGG
TCCGGT
CCGGTT
CGGTTG
GGTTGC
GTTGCC
TTGCCG
TGCCGA
```

k=8の場合: 13個のk-mer

GGTACGGTTCGGTTCGCGA

```
GGTACGGT
GTACGGTT
TACGGTTC
ACGGTTCC
CGGTTCGG
GGTTCGGG
GTTCCGGT
TTCCGGTT
TCCGGTTG
CCGGTTGC
CGGTTGCC
GGTTGCCG
GTTGCCGA
```



k=10の場合: 11個のk-mer

GGTACGGTTCGGTTCGCGA

```
GGTACGGTTC
GTACGGTTCC
TACGGTTCCG
ACGGTTCCGG
CGGTTCGGGT
GGTTCGGGTT
GTTCCGGTTG
TTCCGGTTGC
TCCGGTTGCC
CCGGTTGCCG
CGGTTGCCGA
```

# Introduction6

一般化すると、「L塩基長のリードをk-merで分割すると、 $(L - k + 1)$  個のk-merを生成可能」です。

$$(L - k + 1) = 20 - 6 + 1 = 15$$

k=6の場合: 15個のk-mer

GGTACGGTTCCGGTTGCCGA

```
GGTACG
GTACGG
TACGGT
ACGGTT
CGGTTCC
GGTTCCG
GTTCCGG
TTCCGGG
TCCGGGT
CCGGTTT
CGGTTTG
GGTTTGC
GTTTGCC
TTTGCCG
TGCCCGA
```

$$(L - k + 1) = 20 - 8 + 1 = 13$$

k=8の場合: 13個のk-mer

GGTACGGTTCCGGTTGCCGA

```
GGTACGGT
GTACGGTT
TACGGTTC
ACGGTTCC
CGGTTCCG
GGTTCCGG
GTTCCGGT
TTCCGGTT
TCCGGTTG
CCGGTTGC
CGGTTGCC
GGTTGCCG
GTTGCCGA
```

$$(L - k + 1) = 20 - 10 + 1 = 11$$

k=10の場合: 11個のk-mer

GGTACGGTTCCGGTTGCCGA

```
GGTACGGTTC
GTACGGTTCC
TACGGTTCCG
ACGGTTCCGG
CGGTTCCGGT
GGTTCCGGTT
GTTCCGGTTG
TTCCGGTTGC
TCCGGTTGCC
CCGGTTGCCG
CGGTTGCCGA
```

# Introduction7

分野によって表現方法が異なりますが、k-spectrum kernel(k-スペクトラムカーネル)とか、k-mer出現頻度解析みたいな表現もなされており、様々な局面で利用されています。

k=6の場合: 15個のk-mer

GGTACGGTTCCGGTTGCCGA

```
GGTACG
GTACGG
TACGGT
ACGGTT
CGGTTCC
GGTTCCG
GTTCCGG
TTCCGGG
TCCGGGT
CCGGTTT
CGGTTTG
GGTTTGC
GTTTGCC
TTTGCCG
TGCCCGA
```

k=8の場合: 13個のk-mer

GGTACGGTTCCGGTTGCCGA

```
GGTACGGT
GTACGGTT
TACGGTTC
ACGGTTCC
CGGTTCCG
GGTTCCGG
GTTCCGGT
TTCCGGTT
TCCGGTTG
CCGGTTGC
CGGTTGCC
GGTTGCCG
GTTGCCGA
```

k=10の場合: 11個のk-mer

GGTACGGTTCCGGTTGCCGA

```
GGTACGGTTC
GTACGGTTCC
TACGGTTCCG
ACGGTTCCGG
CGGTTCCGGT
GGTTCCGGTT
GTTCCGGTTG
TTCCGGTTGC
TCCGGTTGCC
CCGGTTGCCG
CGGTTGCCGA
```



# Contents

- Introduction、出現頻度解析(k=2)、出現頻度解析(k=1)
- k=1で実践、multi-FASTAファイル、他の例題を実行
- k=2で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8(k=2)、例題9(k=3)、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11(k=10)、例題12(k=10)、シークエンスエラーを含む場合

# 出現頻度解析 (k=2) 1

例として、k-mer出現頻度解析 (k=2) で、  
①の配列の出現頻度を調べる。ACGT  
の4種類のみで考えると、②k=2の場合  
に可能なk-merの種類数は「AA, AC, AG,  
..., TG, TT」の $4^2 = 16$ 。



GGTACGGTTCCGGTTGCCGA

AA
AC
AG
AT
CA
CC
CG
CT
GA
GC
GG
GT
TA
TC
TG
TT

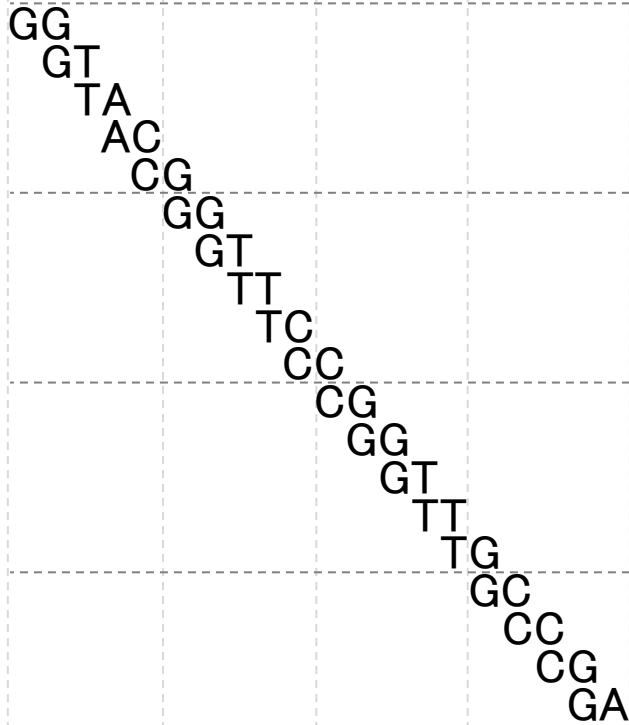


# 出現頻度解析 (k=2) 2

例として、k-mer出現頻度解析 (k=2) で、  
①の配列の出現頻度を調べる。ACGT  
の4種類のみで考えると、②k=2の場合  
に可能なk-merの種類数は「AA, AC, AG,  
..., TG, TT」の $4^2 = 16$ 。③カウント結果。



GGTACGGTTCCGGTTGCCGA



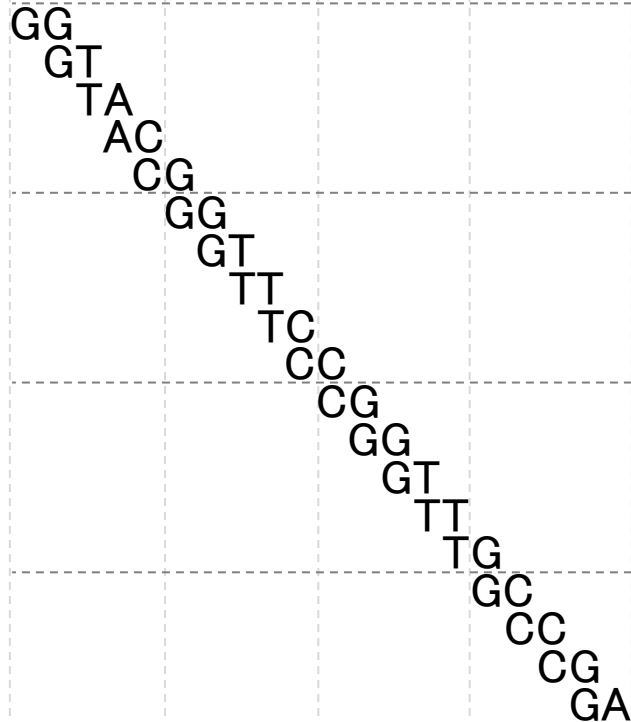
AA	0
AC	1
AG	0
AT	0
CA	0
CC	2
CG	3
CT	0
GA	1
GC	1
GG	3
GT	3
TA	1
TC	1
TG	1
TT	2

# 出現頻度解析 (k=2) 3

例として、k-mer出現頻度解析 (k=2) で、  
①の配列の出現頻度を調べる。ACGT  
の4種類のみで考えると、②k=2の場合  
に可能なk-merの種類数は「AA, AC, AG,  
..., TG, TT」の $4^2 = 16$ 。③カウント結果。  
例えば④は、「CGという2連続塩基が3回  
出現した」という風に解釈する。



GGTACGGTTCGGTTGCCGA

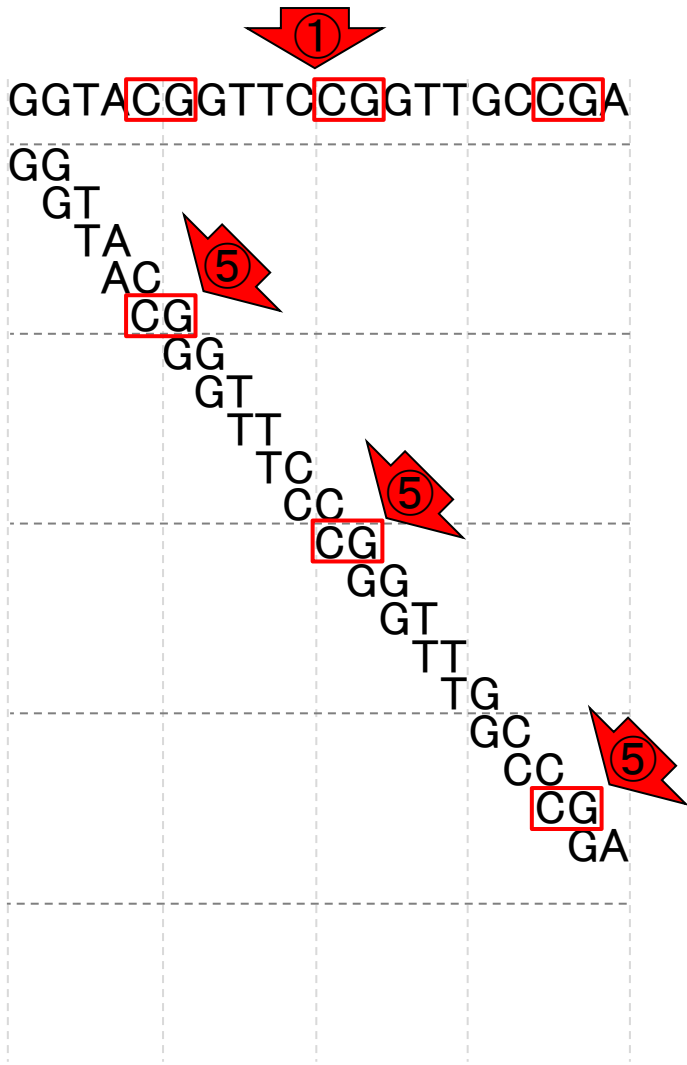


AA	0
AC	1
AG	0
AT	0
CA	0
CC	2
CG	3
CT	0
GA	1
GC	1
GG	3
GT	3
TA	1
TC	1
TG	1
TT	2



# 出現頻度解析 (k=2) 4

例として、k-mer出現頻度解析 (k=2) で、  
 ①の配列の出現頻度を調べる。ACGT  
 の4種類のみで考えると、②k=2の場合  
 に可能なk-merの種類数は「AA, AC, AG,  
 …, TG, TT」の $4^2 = 16$ 。③カウント結果。  
 例えば④は、「CGという2連続塩基が3回  
 出現した」という風に解釈する。確かに、  
 ⑤CGが3個ありますね。



③

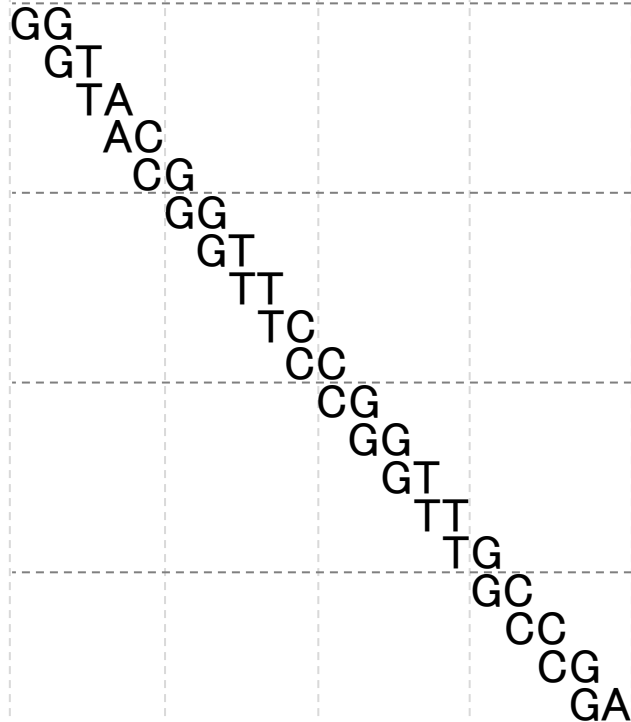
AA	0
AC	1
AG	0
AT	0
CA	0
CC	2
CG	3
CT	0
GA	1
GC	1
GG	3
GT	3
TA	1
TC	1
TG	1
TT	2

④

# 出現頻度解析 (k=2) 5



GGTACGGTTCCGGTTGCCGA



AA	0
AC	1
AG	0
AT	0
CA	0
CC	2
CG	3
CT	0
GA	1
GC	1
GG	3
GT	3
TA	1
TC	1
TG	1
TT	2



例として、k-mer出現頻度解析 (k=2) で、  
①の配列の出現頻度を調べる。ACGT  
の4種類のみで考えると、②k=2の場合  
に可能なk-merの種類数は「AA, AC, AG,  
..., TG, TT」の $4^2 = 16$ 。③カウント結果。  
例えば④は、「CGという2連続塩基が3回  
出現した」という風に解釈する。確かに、  
⑤CGが3個ありますね。k-mer出現頻度  
解析のメリットは、①入力塩基配列の長  
さに関係なく、常に②一定の要素数から  
なる、③数値ベクトルが得られること。

# 出現頻度解析 (k=2) 6

例えば、①と②の2つの配列比較(類似度の評価)が可能。赤下線部分は同一であり、②のほうが5つだけCが多い。



GGTACGGTTCCGGTTGCCGA



GGTACGGTTCCGGTTGCCGACCCCC

# 出現頻度解析 (k=2) 7

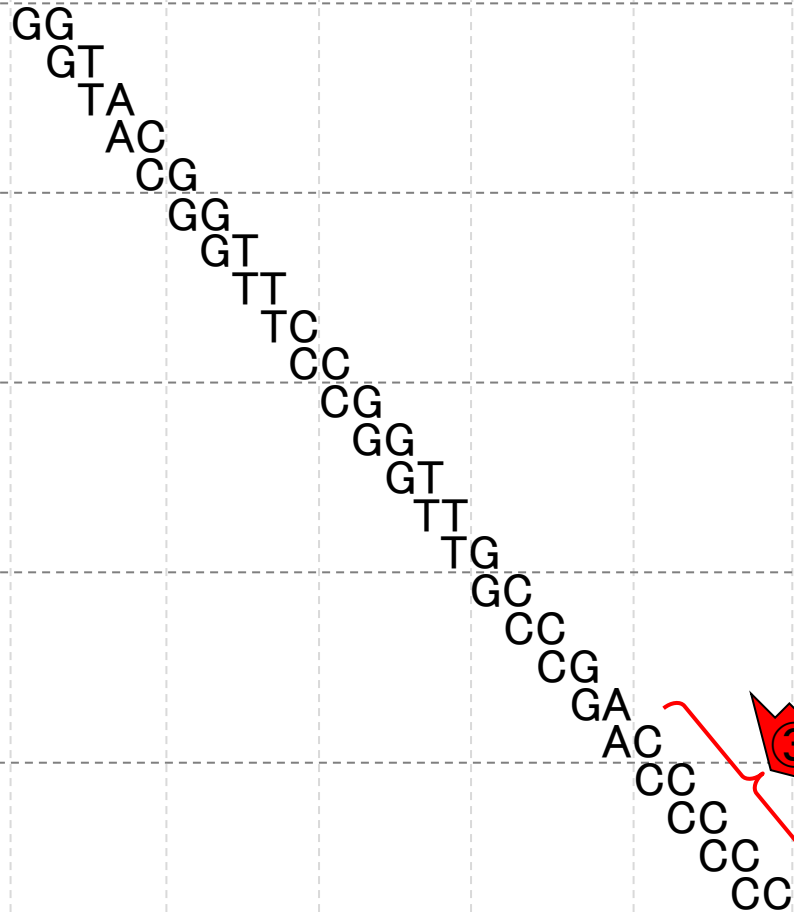
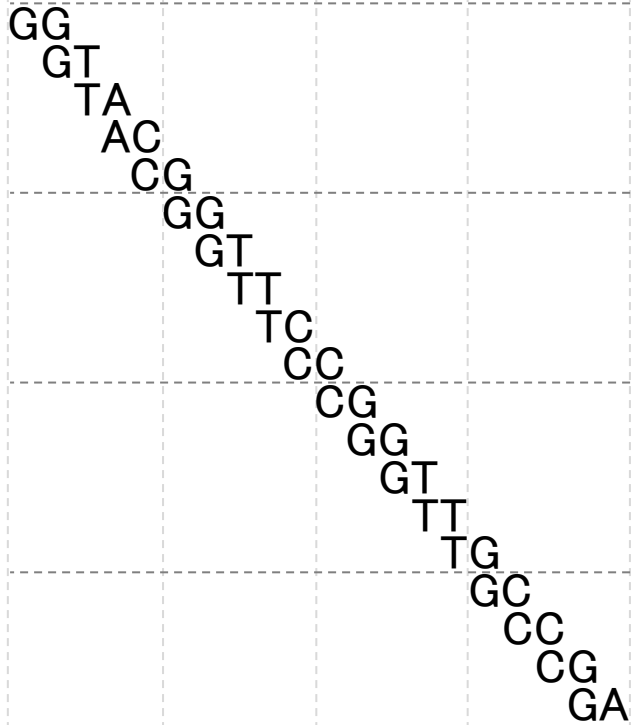
例えば、①と②の2つの配列比較(類似度の評価)が可能。赤下線部分は同一であり、②のほうが5つだけCが多い。それゆえ、③の分だけ出現頻度が上乘せされて…



GGTACGGTTCCGGTTGCCGA



GGTACGGTTCCGGTTGCCGACCCCC





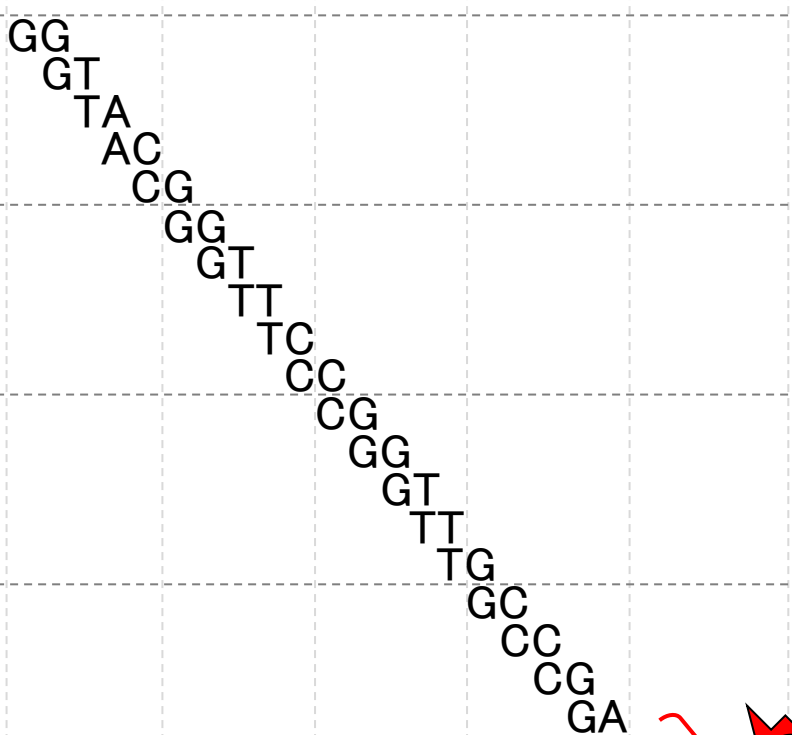
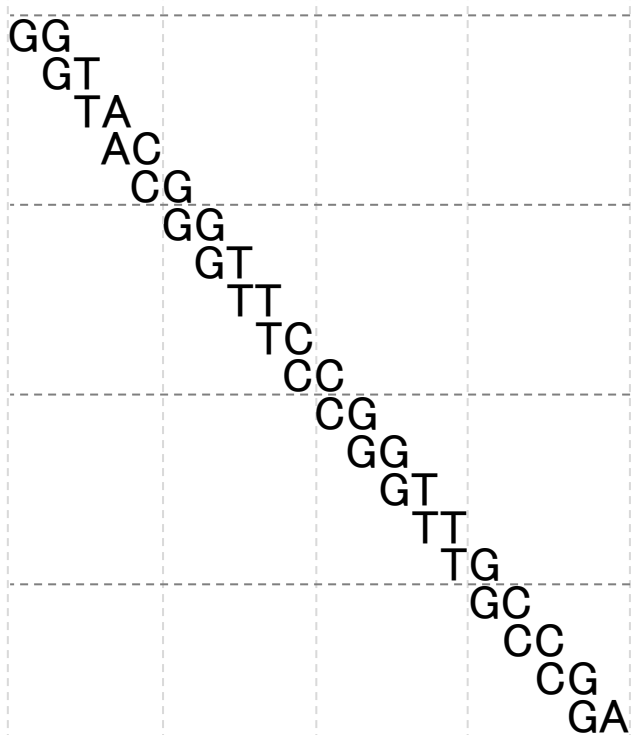
# 出現頻度解析 (k=2) 8

例えば、①と②の2つの配列比較(類似度の評価)が可能。赤下線部分は同一であり、②のほうが5つだけCが多い。それゆえ、③の分だけ出現頻度が上乘せられて、①と②の出現頻度は、それぞれこんな感じになります。



GGTACGGTTCCGGTTGCCGA

GGTACGGTTCCGGTTGCCGACCCCC



AA	0	0
AC	1	2
AG	0	0
AT	0	0
CA	0	0
CC	2	6
CG	3	3
CT	0	0
GA	1	1
GC	1	1
GG	3	3
GT	3	3
TA	1	1
TC	1	1
TG	1	1
TT	2	2

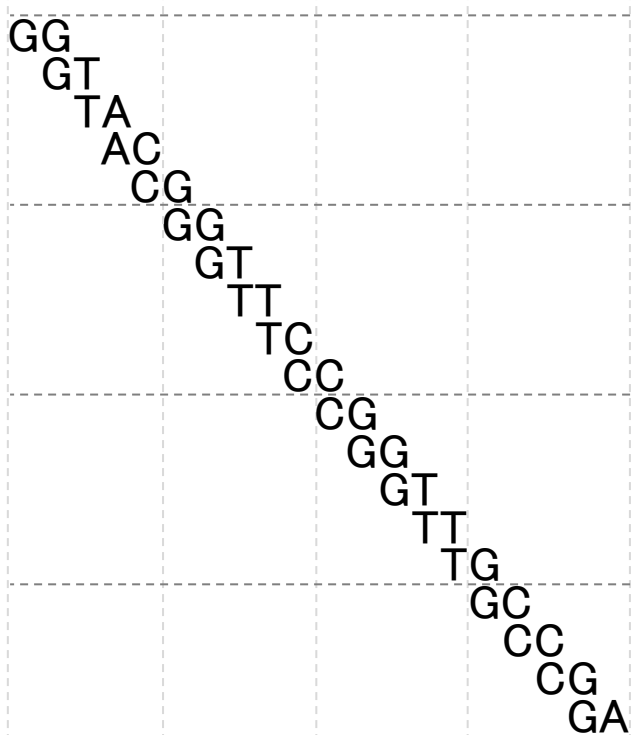


# 出現頻度解析 (k=2) 9

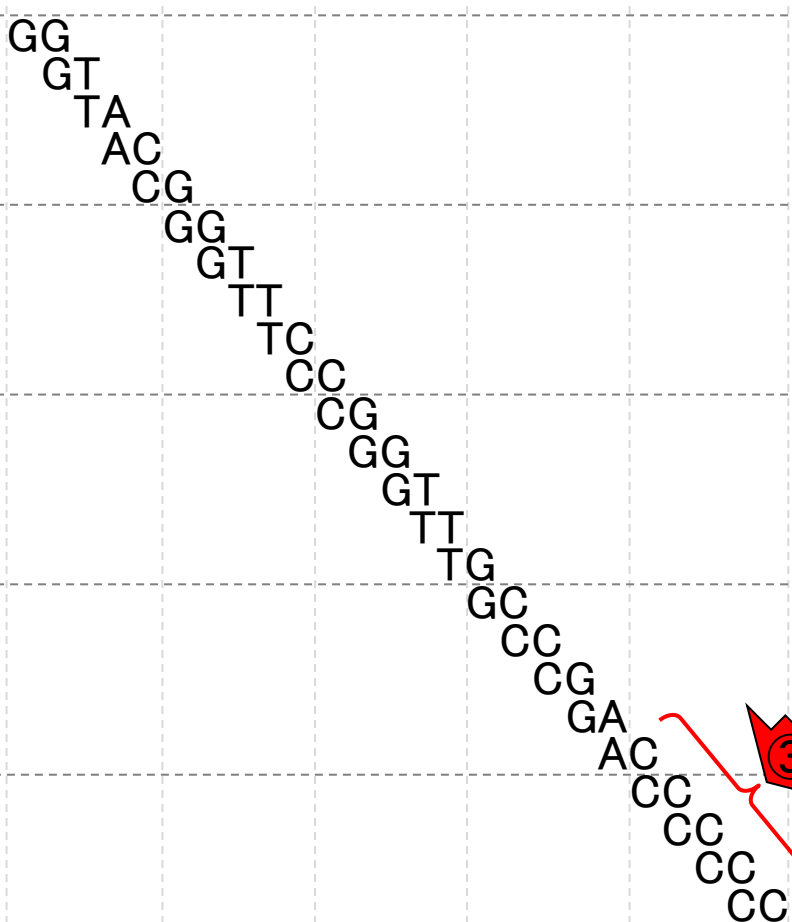
例えば、①と②の2つの配列比較(類似度の評価)が可能。赤下線部分は同一であり、②のほうが5つだけCが多い。それゆえ、③の分だけ出現頻度が上乘せられて、①と②の出現頻度は、それぞれこんな感じになります。③に関連する箇所は、④です。



GGTACGGTTCCGGTTGCCGA



GGTACGGTTCCGGTTGCCGACCCCC

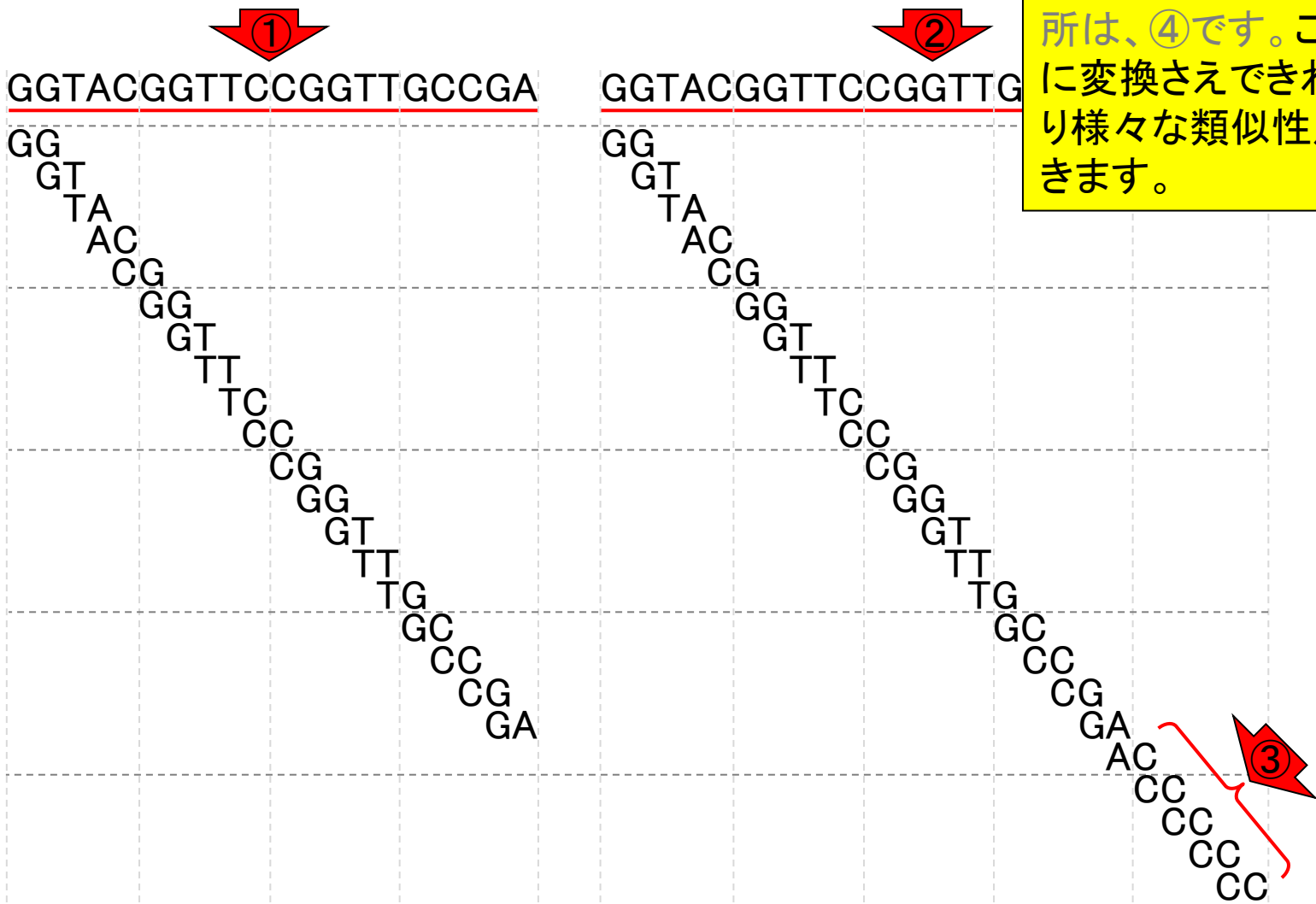


AA	0	0
AC	1	2
AG	0	0
AT	0	0
CA	0	0
CC	2	6
CG	3	3
CT	0	0
GA	1	1
GC	1	1
GG	3	3
GT	3	3
TA	1	1
TC	1	1
TG	1	1
TT	2	2



# 出現頻度解析 (k=2) 10

例えば、①と②の2つの配列比較(類似度の評価)が可能。赤下線部分は同一であり、②のほうが5つだけCが多い。それゆえ、③の分だけ出現頻度が上乗せされて、①と②の出現頻度は、それぞれこんな感じになります。③に関連する箇所は、④です。こんな感じで数値ベクトルに変換さえできれば、あとは相関係数なり様々な類似性尺度を適用することができます。



AG	0	0
AT	0	0
CA	0	0
CC	2	6
CG	3	3
CT	0	0
GA	1	1
GC	1	1
GG	3	3
GT	3	3
TA	1	1
TC	1	1
TG	1	1
TT	2	2

# Contents

- Introduction、出現頻度解析( $k=2$ )、出現頻度解析( $k=1$ )
- $k=1$ で実践、multi-FASTAファイル、他の例題を実行
- $k=2$ で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8( $k=2$ )、例題9( $k=3$ )、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11( $k=10$ )、例題12( $k=10$ )、シークエンスエラーを含む場合

# 出現頻度解析 (k=1) 1

k=1でのk-mer出現頻度解析は、実質的に塩基ごとの出現回数をカウントする作業と同じです。



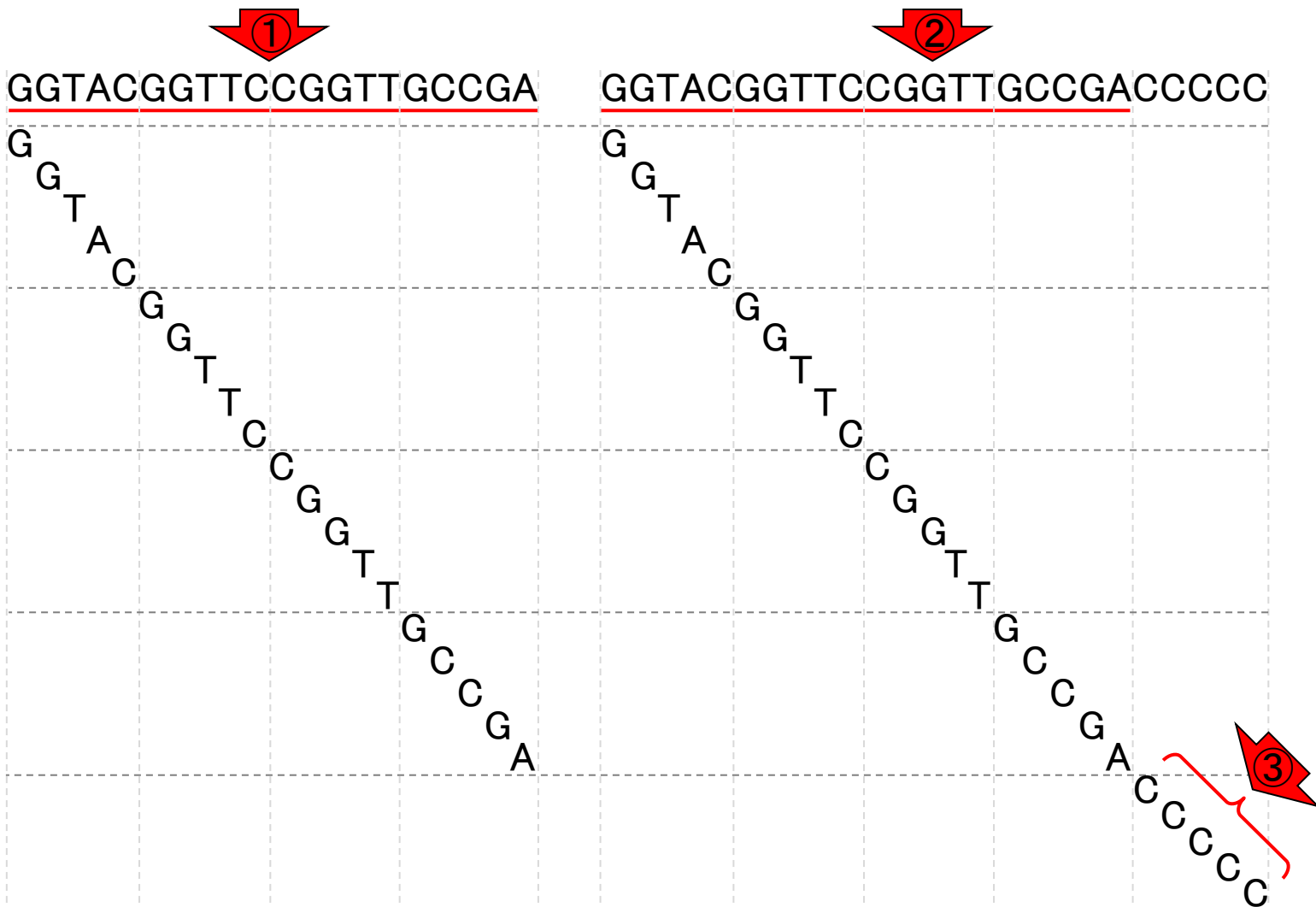
GGTACGGTTCCGGTTGCCGA

GGTACGGTTCCGGTTGCCGACCCCC

A	2	2
C	5	10
G	8	8
T	5	5

# 出現頻度解析 (k=1) 2

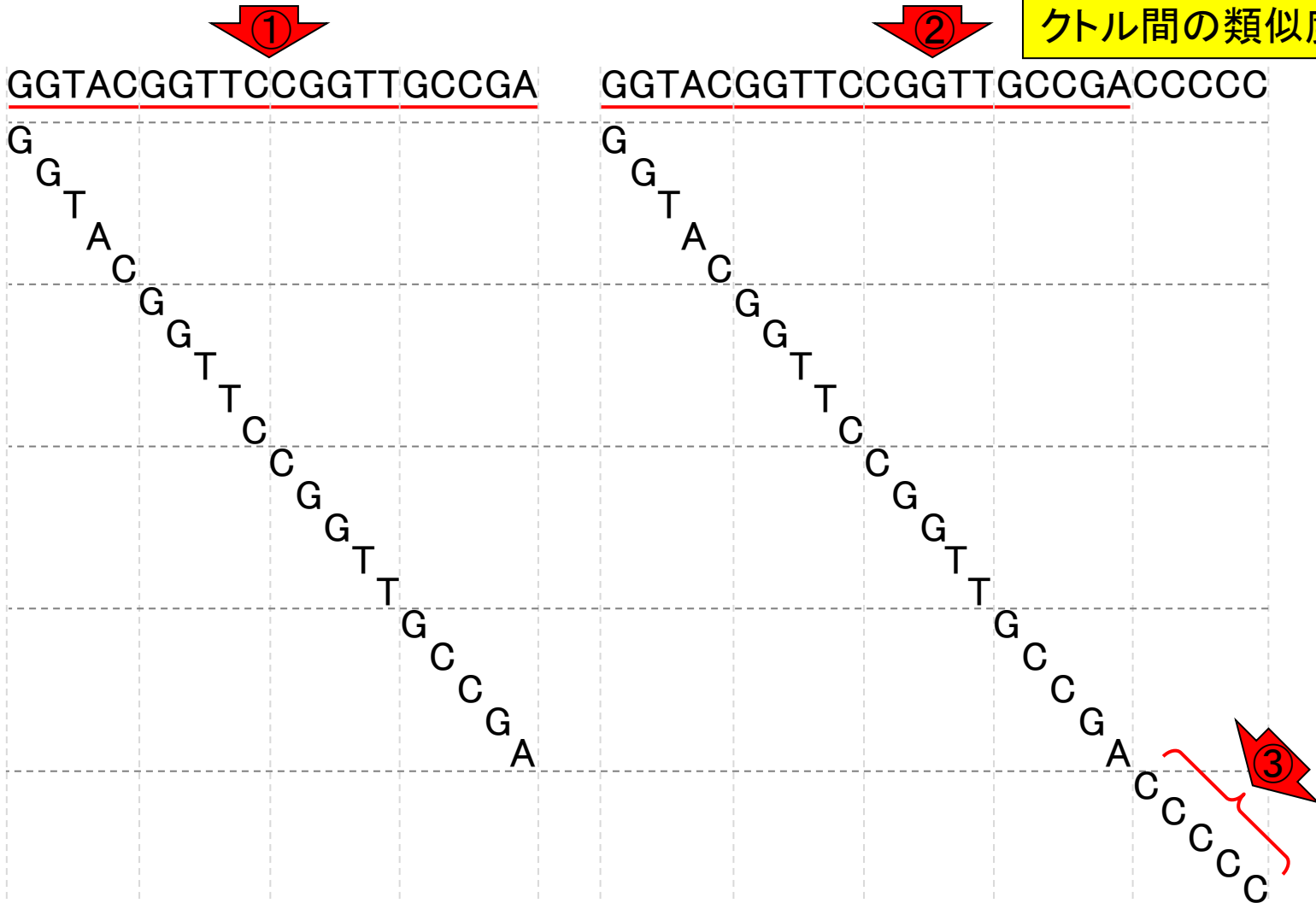
k=1でのk-mer出現頻度解析は、実質的に塩基ごとの出現回数をカウントする作業と同じです。③に関連する箇所は、④です。



	①	②	
A	2	2	④
C	5	10	
G	8	8	
T	5	5	

# 出現頻度解析 (k=1) 3

k=1でのk-mer出現頻度解析は、実質的に塩基ごとの出現回数をカウントする作業と同じです。③に関連する箇所は、④です。この場合も、①と②それぞれ要素数が4で固定の数値ベクトルが生成されるので、配列長が異なっても数値ベクトル間の類似度を評価可能。



① ②

A	2	2
C	5	10
G	8	8
T	5	5

④

# Contents

- Introduction、出現頻度解析( $k=2$ )、出現頻度解析( $k=1$ )
- $k=1$ で実践、multi-FASTAファイル、他の例題を実行
- $k=2$ で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8( $k=2$ )、例題9( $k=3$ )、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11( $k=10$ )、例題12( $k=10$ )、シーケンスエラーを含む場合



# k=1で実践1

①の配列を入力として、②の出現頻度解析結果を得る作業を行ってみましょう。  
①の配列のFASTA形式ファイルは、`seq_20.fasta`です。



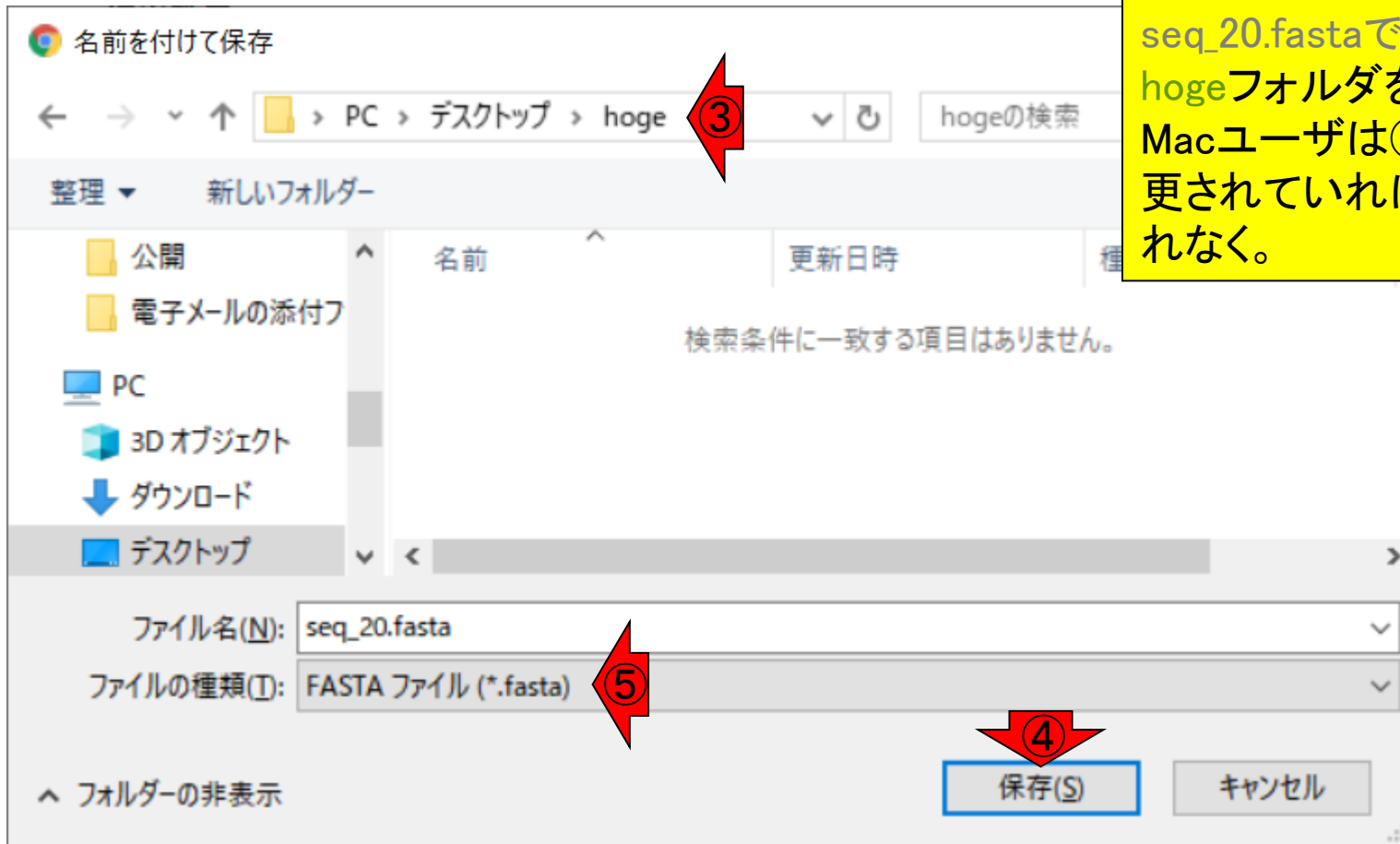
GGTACGGTTCGGTTGCCGA



A	2
C	5
G	8
T	5

# k=1で実践2

①の配列を入力として、②の出現頻度解析結果を得る作業を行ってみましょう。①の配列のFASTA形式ファイルは、seq\_20.fastaです。③デスクトップ上にhogeフォルダを作成して④保存。特にMacユーザは⑤拡張子が.txtに勝手に変更されていけば.fastaに戻すことをお忘れなく。



# k=1で実践3

- ①「デスクトップのhogeフォルダ」にある、
- ②seq\_20.fastaの、③中身。



# k=1で実践4

## (Rで)塩基配列解析



(last modified 2022/05/09, since 2010)

このウェブページの多くは、[インストール](#)についての推奨手順 ([Windows2022.03.31版](#)と[Macintosh2021.04.01版](#))に従って フリーソフトRと必要なパッケージをインストール済みであるという前提で記述しています。 初心者の方は[基本的な利用法](#)(Windows2022.04.03版の[PPTX](#)と[PDF](#) ; Macintosh2020.03.13版の[PPTX](#)と[PDF](#))で自習してください。

@Agribio\_utokyoさんをフォロー

[アグリバイオ](#)、[\(Rで\)塩基配列解析のサブページ](#)、[生命情報解析研究室](#)

### What's new? ([過去のお知らせはこちら](#))

- 「解析 | 一般 | [パターンマッチング](#)」の例題5の入力ファイル名が間違っていたのを訂正しました (data\_seqlogo1.txt -> data\_seqlogo1.fasta)。 (中村 弘太 氏提供情報)(2022/05/09) **NEW**
- [東京大学・大学院農学生命科学研究科・応用生命工学専攻](#)の[令和5\(2023\)年度大学院学生募集公開ガイダンス](#)の第2回目は、5月28日(土)に開催します。(2022/05/08) **NEW** [トップページ](#)

# k=1で実践5

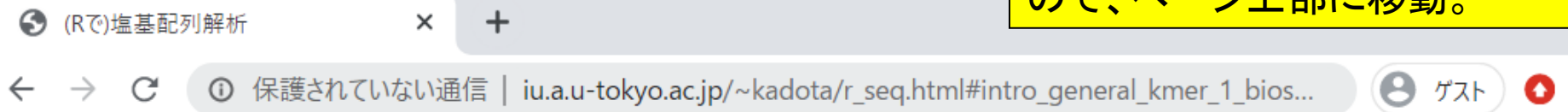
(Rで)塩基配列解析

iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html

- 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 2015/02/19)
- 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 | について](#) (last modified 2019/09/06)
- イントロ | 一般 | 指定した範囲の配列を取得 | [Biostring](#) (last modified 2022/04/26) **NEW**
- イントロ | 一般 | [指定したID\(染色体やdescription\)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 | について](#) (last modified 2019/09/06)
- イントロ | 一般 | 翻訳配列(translate)を取得 | [Biostrings](#) (last modified 2015/09/12)
- イントロ | 一般 | 翻訳配列(translate)を取得 | [seqinr\(Charif\\_2005\)](#) (last modified 2015/03/09)
- イントロ | 一般 | [相補鎖\(complement\)を取得](#) (last modified 2019/03/10)
- イントロ | 一般 | [逆相補鎖\(reverse complement\)を取得](#) (last modified 2019/03/10)
- イントロ | 一般 | [逆鎖\(reverse\)を取得](#) (last modified 2019/03/10)
- イントロ | 一般 | k-mer解析 | k=1(塩基ごとの出現頻度解析) | [Biostrings](#) ② (last modified 2016/04/27)
- イントロ | 一般 | k-mer解析 | k=2(2連続塩基の出現頻度解析) | [Biostrings](#) (last modified 2016/01/28)
- イントロ | 一般 | k-mer解析 | k=3(3連続塩基の出現頻度解析) | [Biostrings](#) (last modified 2016/01/28)
- イントロ | 一般 | k-mer解析 | k=n(n連続塩基の出現頻度解析) | [Biostrings](#) (last modified 2016/05/01)
- 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 [トップページへ](#))
- イントロ | 一般 | 配列取得 | ゲノム配列 | [公共DBから](#) (last modified 2017/04/11)

# k=1で実践6

k-mer出現頻度解析(k=1)を行う項目は、①の中の、②です。こんな感じになります。③例題1をテンプレートとして用いるので、ページ上部に移動。



## イントロ | 一般 | k-mer解析 | k=1(塩基ごとの出現頻度解析) | Biostrings

Biostringsパッケージを用いて、multi-FASTA形式ファイルを読み込んで、"A", "C", "G", "T", ..., "N", ...など塩基ごとの出現頻度を調べるやり方を示します。k-mer解析のk=1の場合に相当します。

「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ー。

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:

配列ごとに出現頻度をカウントした結果を返すやり方です。

```
in_f <- "hoge4.fa"           #入力ファイル名を指定してin_fに格納
out_f <- "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
out <- alphabetFrequency(fasta) #A,C,G,T,..の数を各配列ごとにカウントした結果をトップページへ
out                                #outの中身を表示
```

# k=1で実践7

k-mer出現頻度解析(k=1)を行う項目は、①の中の、②です。こんな感じになります。③例題1をテンプレートとして用いるので、ページ上部に移動。こんな感じ。

(Rで)塩基配列解析

保護されていない通信 | [iu.a.u-tokyo.ac.jp/~kadota/r\\_seq.html#intro\\_general\\_kmer\\_1\\_bios...](http://iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#intro_general_kmer_1_bios...)

## 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:

配列ごとに出現頻度をカウントした結果を返すやり方です。

```
in_f <- "hoge4.fa"           #入力ファイル名を指定してin_fに格納
out_f <- "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
out <- alphabetFrequency(fasta) #A,C,G,T,..の数を各配列ごとにカウントした結果をoutに格納
out                                     #outの中身を表示

#ファイルに保存
tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.names=T)#tmpの中身を指定して
```

[トップページ](#)↑

# k=1で実践8

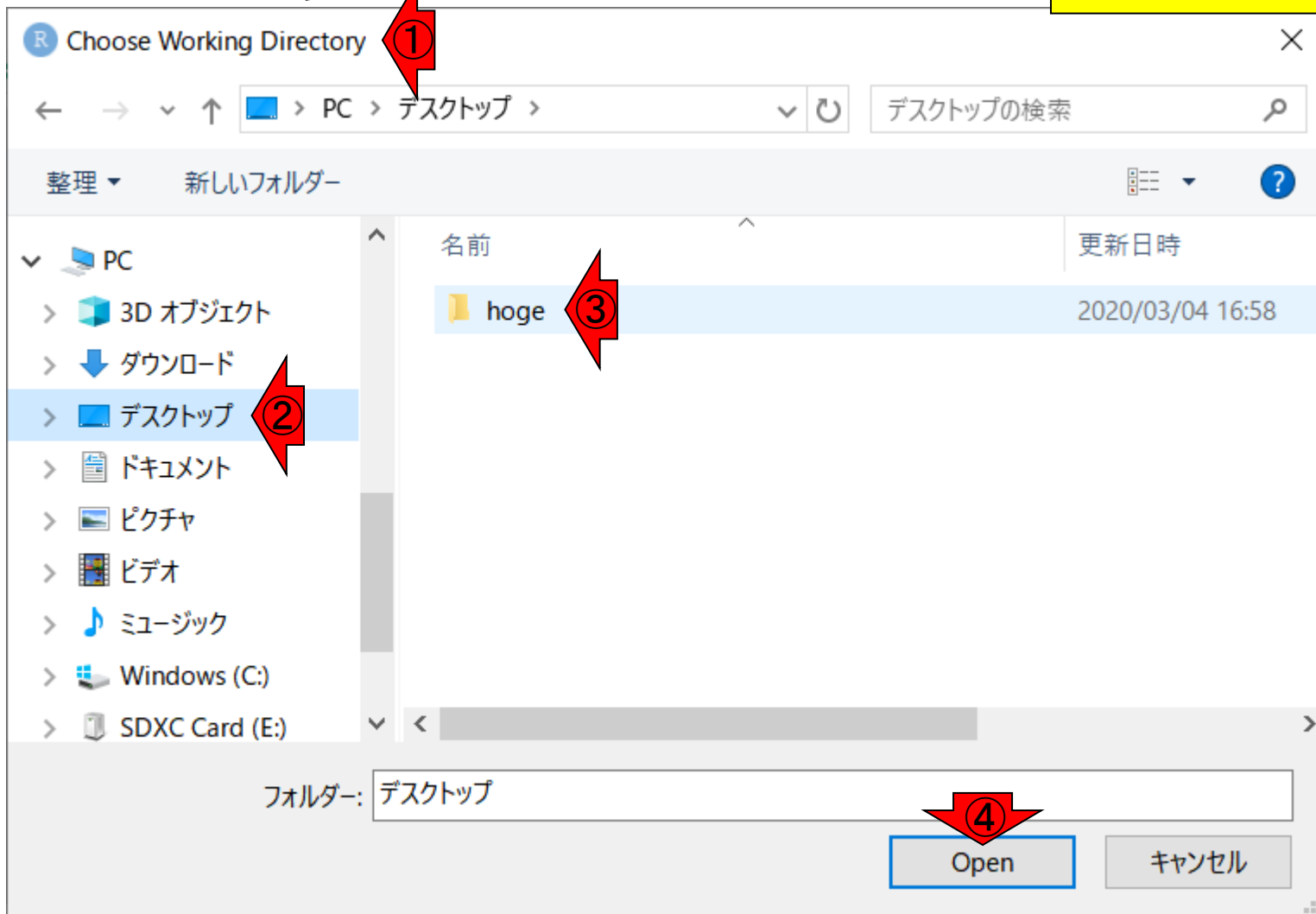
作業ディレクトリを「デスクトップのhoge」  
にすべく、①Session、②Set Working  
Directory、③Choose Directory...

The screenshot shows the RStudio interface. The 'Session' menu is open, and 'Set Working Directory' is selected. A secondary menu is open for 'Set Working Directory', with 'Choose Directory...' selected. Red arrows with numbers 1, 2, and 3 indicate the sequence of actions: 1 points to the 'Session' menu, 2 points to 'Set Working Directory', and 3 points to 'Choose Directory...'. The console shows R version 3.6.1 and some Japanese text. The file explorer on the right shows a directory structure with folders for years 2017-2020 and files like 'hoge1.txt'.



# k=1で実践9

①こんな感じのフォルダ選択画面が出ますので、②③④のような感じで変更を完了してください。



# k=1で実践10

①こんな感じのフォルダ選択画面が出ますので、②③④のような感じで変更を完了してください。こんな感じになります。  
⑤作業ディレクトリが変更され、⑥も変更されて、⑦目的のファイル(seq\_20.fasta)が見られます。

The screenshot shows the RStudio interface. The terminal window displays the R startup message and the command `setwd("C:/Users/kadota/Desktop/hoge")` has been executed. The file explorer on the right shows the directory `C:/Users/kadota/Desktop/hoge` containing files `.Rhistory` and `seq_20.fasta`. Red arrows with circled numbers 5, 6, and 7 point to the terminal prompt, the file explorer, and the `seq_20.fasta` file respectively.

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

Console Terminal x Jobs x

C:/Users/kadota/Desktop/hoge/

Copyright (C) 2019 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 `'licence()'` 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については  
`'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。

```
> setwd("C:/Users/kadota/Desktop/hoge")  
> |
```

Environment History Connections

To Source

setwd("C:/Users/kadota/Desk...

Files Plots Packages Help View

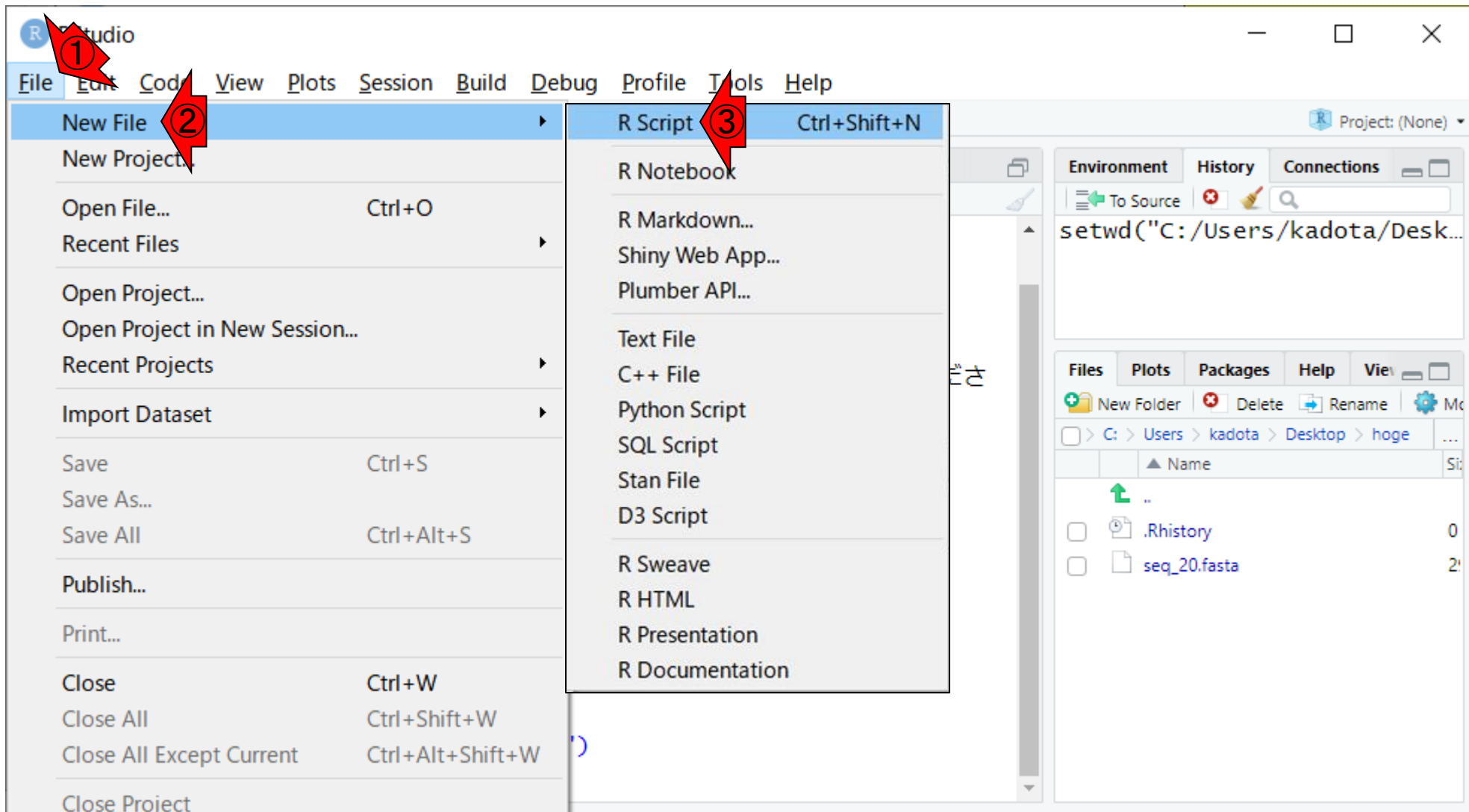
New Folder Delete Rename

C: > Users > kadota > Desktop > hoge

Name	
..	
.Rhistory	0
seq_20.fasta	2

①File、②New File、③R Scriptとして、「新規のRスクリプト」を開く。

# k=1で実践11



# k=1で実践12

①File、②New File、③R Scriptとして、「新規のRスクリプト」を開く。こんな感じになります。

The screenshot shows the RStudio interface. The main editor window contains a single line of code: `1`. The console window at the bottom shows the following text:

```
C:/Users/kadota/Desktop/hoge/
> demo()
'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。

> setwd("C:/Users/kadota/Desktop/hoge")
>
```

The right-hand pane shows the file explorer with the following contents:

Name	Size
..	
.Rhistory	0
seq_20.fasta	21

# k=1で実践13

①File、②New File、③R Scriptとして、「新規のRスクリプト」を開く。こんな感じになります。④例題1のコード全体を、⑤コピーして...

(Rで)塩基配列解析

④ [iu.a.u-tokyo.ac.jp/~kadota/r\\_seq.html#intro\\_general\\_kmer\\_1\\_bios...](http://iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#intro_general_kmer_1_bios...)

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:  
配列ごとに出現頻度をカウントした結果を返すやり方です。

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

#本番
out <- alphabetFrequency(fasta)
out

#ファイルに保存
tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.names=T)#tmpの中身を指定して
```

⑤

コピー(C)	Ctrl+C
Google で「in_f <- "hoge4.fa" #入力ファイル名を...」を検索(S)	
印刷(P)...	Ctrl+P
検証(I)	Ctrl+Shift+I

[トップページ](#)↑

# k=1で実践14

①File、②New File、③R Scriptとして、「新規のRスクリプト」を開く。こんな感じになります。④例題1のコード全体を、⑤コピーして、⑥Paste。

The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu bar is a toolbar with icons for file operations and a search bar. The main editor window shows a file named 'Untitled1' with a single line of code: '1'. A context menu is open over the code, with 'Paste' highlighted in blue. A red arrow with the number '6' points to the 'Paste' option. The console window at the bottom shows the following text:

```
C:/Users/kadota/Desktop/hoge/
> demo()
'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。

> setwd("C:/Users/kadota/Desktop/hoge")
> |
```

The right-hand side of the interface shows the Environment, History, and Connections panels. The Environment panel shows a variable 'setwd' with a value of 'C:/Users/kadota/Desktop/hoge...'. The Files panel shows a directory listing for 'C:/Users/kadota/Desktop/hoge' with files like '.Rhistory' and 'seq\_20.fasta'.

# k=1で実践15

①File、②New File、③R Scriptとして、「新規のRスクリプト」を開く。こんな感じになります。④例題1のコード全体を、⑤コピーして、⑥Paste。こんな感じになります。

The screenshot shows the RStudio environment with the following components:

- Source Editor:** Contains R code for calculating the frequency of each nucleotide (A, C, G, T) in a FASTA file. The code is as follows:

```
10 #本番
11 out <- alphabetFrequency(fasta)      #A,C,G,T,..の数を各配列ご
12 out                                  #outの中身を表示
13
14 #ファイルに保存
15 tmp <- cbind(names(fasta), out)      #保存したい情報をtmpに格
16 write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=1)
17
```
- Environment/History/Connections:** Shows the current working directory set to "C:/Users/kadota/Desktop/hoge...".
- Files Panel:** Displays the file structure of the current directory, including ".Rhistory" and "seq\_20.fasta".
- Console:** Shows the R prompt and the command `setwd("C:/Users/kadota/Desktop/hoge")` being entered.

# k=1で実践16

①File、②New File、③R Scriptとして、「新規のRスクリプト」を開く。こんな感じになります。④例題1のコード全体を、⑤コピーして、⑥Paste。こんな感じになります。解析したいのは⑦seq\_20.fastaなので、⑧スクリプトの上部に移動して、⑨該当箇所を変更...

The screenshot shows the RStudio interface with the following elements:

- Source Editor:** Contains R code for file operations and package loading. Red arrow 9 points to line 1, and red arrow 8 points to the `setwd` function call.
- Files Panel:** Shows the file explorer with `seq_20.fasta` highlighted. Red arrow 7 points to this file.
- Console:** Shows the execution of `setwd("C:/Users/kadota/Desktop/hoge")`.

```
1 in_f <- "hoge4.fa"
2 out_f <- "hoge1.txt"
3
4 #必要なパッケージをロード
5 library(Biostrings)
6
7 #入力ファイルの読み込み
8
```

```
setwd("C:/Users/kadota/Desktop/hoge")
```

```
> setwd("C:/Users/kadota/Desktop/hoge")
> |
```



# k=1で実践17

①File、②New File、③R Scriptとして、「新規のRスクリプト」を開く。こんな感じになります。④例題1のコード全体を、⑤コピーして、⑥Paste。こんな感じになります。解析したいのは⑦seq\_20.fastaなので、⑧スクリプトの上部に移動して、⑨該当箇所を変更した状態。スペルミスに注意！

RStudio interface showing an R script and a file explorer. The script contains R code for reading a FASTA file and using Biostrings. The file explorer shows a directory with files .Rhistory and seq\_20.fasta. Red arrows point to specific elements: 7 points to seq\_20.fasta, 8 points to the setwd function in the script, and 9 points to the file path in the script.

```
1 in_f <- "seq_20.fasta"
2 out_f <- "hoge1.txt"
3
4 #必要なパッケージをロード
5 library(Biostrings)
6
7 #入力ファイルの読み込み
8
```

Console output:

```
> setwd("C:/Users/kadota/Desktop/hoge")
> |
```

変更後のコード全体をコピー実行すべく、  
①左上から…

# k=1で実践18

The screenshot shows the RStudio interface with the following components:

- Code Editor:** Contains R code for file handling and package loading. Lines 1-8 are highlighted in blue. A red arrow with the number '1' points to the top-left corner of the editor.
- Environment/History/Connections:** Shows the current working directory set to "C:/Users/kadota/Desktop/hoge".
- Files:** Shows the directory structure, including "seq\_20.fasta" and ".Rhistory".
- Console:** Shows the execution of `setwd("C:/Users/kadota/Desktop/hoge")`.

```
1 in_f <- "seq_20.fasta" #入力ファイル名を指定
2 out_f <- "hoge1.txt" #出力ファイル名を指定して
3
4 #必要なパッケージをロード
5 library(Biostrings) #パッケージの読み込み
6
7 #入力ファイルの読み込み
8
```

```
> setwd("C:/Users/kadota/Desktop/hoge")
>
```

# k=1で実践19

変更後のコード全体をコピー実行すべく、  
①左上から、②一番下まで反転させて、  
③Run。

The screenshot shows the RStudio interface with the following R code in the editor:

```
11 out <- alphabetFrequency(fasta)      #A,C,G,T,..の数を各配列ご
12 out                                  #outの中身を表示
13
14 #ファイルに保存
15 tmp <- cbind(names(fasta), out)      #保存したい情報をtmpに格
16 write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=
17 存
```

The Run button is highlighted with a red arrow labeled ③. The end of the code block is highlighted with a red arrow labeled ②.

The Console shows the following output:

```
C:\Users\kadota\Desktop\hoge/
> demo()
'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。

> setwd("C:/Users/kadota/Desktop/hoge")
> |
```

The Environment pane shows the current working directory set to "C:/Users/kadota/Desktop/hoge". The Files pane shows the directory contents:

Name	Size
..	
.Rhistory	0
seq_20.fasta	21

# k=1で実践20

変更後のコード全体をコピー実行すべく、  
 ①左上から、②一番下まで反転させて、  
 ③Run。実行完了。

The screenshot shows the RStudio interface with the following code in the editor:

```

11 out <- alphabetFrequency(fasta)      #A,C,G,T,..の数を各配列ご
12 out                                  #outの中身を表示
13
14 #ファイルに保存
15 tmp <- cbind(names(fasta), out)      #保存したい情報をtmpに格納
16 write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=I
17 存
    
```

The console output is as follows:

```

> out
      A C G T M R W S Y K V H D B N - + .
[1,] 2 5 8 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0
>
> #ファイルに保存
> tmp <- cbind(names(fasta), out)      #保存したい情報をtmpに格納
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F,
  col.names=T)#tmpの中身を指定したファイル名で保存
>
    
```

The file explorer on the right shows the directory structure: C:\Users\kadota\Desktop\hoge, containing files like .Rhistory and seq\_20.fasta.

# k=1で実践21

変更後のコード全体をコピー実行すべく、  
①左上から、②一番下まで反転させて、  
③Run。実行完了。私は、④少し右側の  
横幅を広げて、⑤リロードすると、⑥出力  
ファイルが見られました。

The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains R code for calculating nucleotide frequencies and saving the results to a file. Lines 11-17 are highlighted in blue.
- Console:** Shows the execution output, including the frequency table and the execution of the save command.
- Environment/History/Connections:** Shows the execution history of the code.
- Files Panel:** Shows the file explorer view of the current project directory, highlighting the newly created file 'seq\_20.fasta'.

```
11 out <- alphabetFrequency(fasta)      #A,C,G,T,..の数を各
12 out                                  #outの中身を表示
13
14 #ファイルに保存
15 tmp <- cbind(names(fasta), out)      #保存したい情報をtmp
16 write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.names=T) #tmpの中身を指定したファイル名で保存
17
```

```
> out
      A C G T M R W S Y K V H D B N - + .
[1,] 2 5 8 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
> #ファイルに保存
> tmp <- cbind(names(fasta), out)      #保存したい情報をtmpに格納
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.names=T) #tmpの中身を指定したファイル名で保存
>
```

Files Panel:

Name	Size
..	
.Rhistory	0 B
hoge1.txt	80 B
seq_20.fasta	29 B

# k=1で実践22

変更後のコード全体をコピー実行すべく、  
 ①左上から、②一番下まで反転させて、  
 ③Run。実行完了。私は、④少し右側の  
 横幅を広げて、⑤リロードすると、⑥出力  
 ファイルが見られました。⑥をクリックし  
 た結果。⑦だと列名がずれてますが、⑧  
 を眺めると、「Aが2個、Cが5個、Gが8個、  
 Tが5個」となっているのがよくわかります。

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function

hoge1.txt

```

1   A C G T M R W S Y K V H D B N - + .
2   hoge 2 5 8 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3
    
```

3:1 Text File

Console

```

> UU
[1,] 2 5 8 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0
>
> #ファイルに保存
> tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.name
s=F, col.names=T)#tmpの中身を指定したファイル名で保存
>
    
```

```

out #outの中身を表示
#ファイルに保存
tmp <- cbind(names(fasta), out...)
write.table(tmp, out_f, sep="\t"
    
```

Files Plots Packages Help Viewer

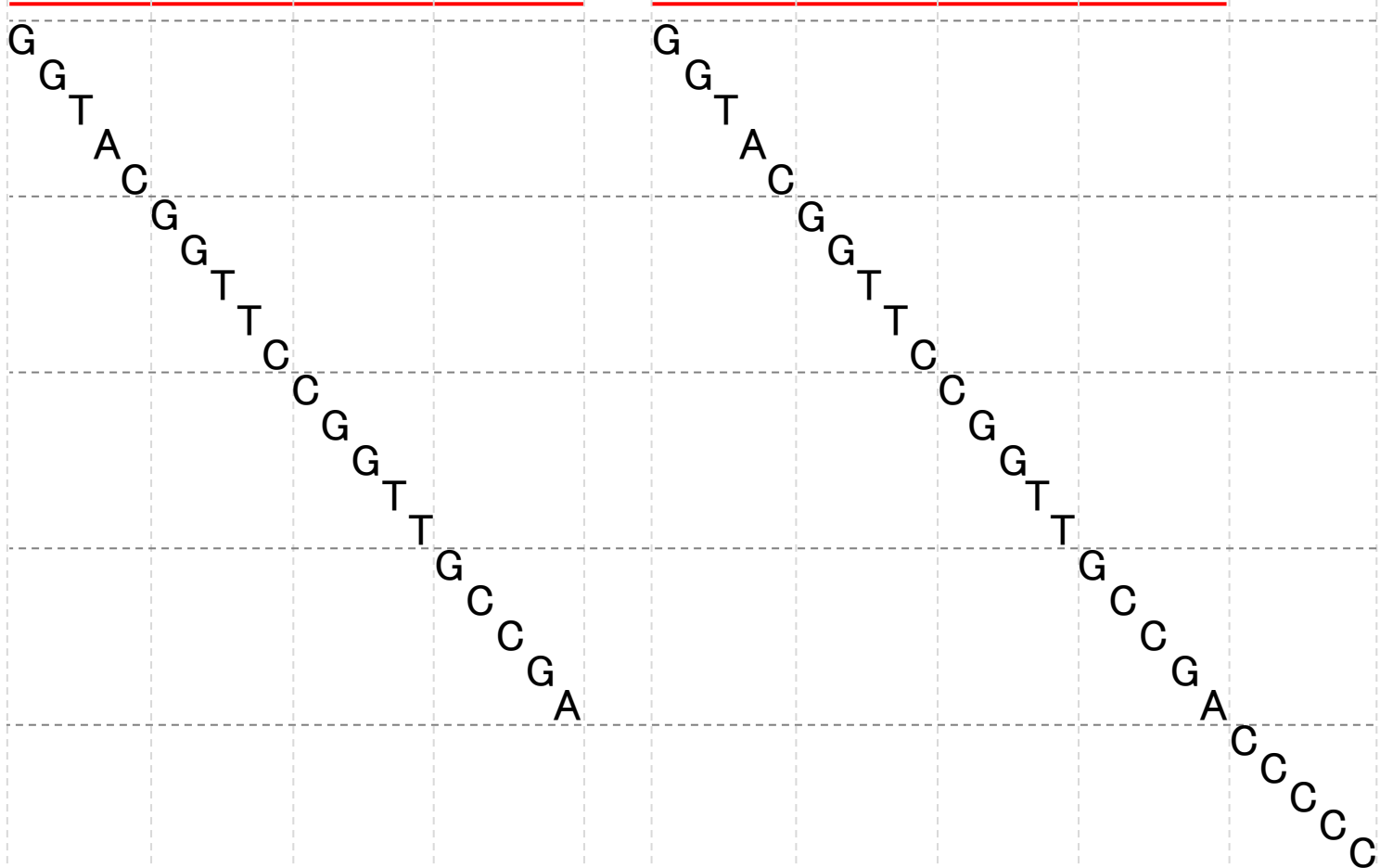
C:\Users\kadota\Desktop\hoge

Name	Size
..	
.Rhistory	0 B
hoge1.txt	80 B
seq_20.fasta	29 B

# k=1で実践23

GGTACGGTTCCGGTTGCCGA

GGTACGGTTCCGGTTGCCGACCCCC



A	2	2
C	5	10
G	8	8
T	5	5

# Contents

- Introduction、出現頻度解析( $k=2$ )、出現頻度解析( $k=1$ )
- $k=1$ で実践、multi-FASTAファイル、他の例題を実行
- $k=2$ で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8( $k=2$ )、例題9( $k=3$ )、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11( $k=10$ )、例題12( $k=10$ )、シークエンスエラーを含む場合



# multi-FASTAファイル1

①を入力として得た、②の出力ファイルの形式が、③配列あたり1行で表現されている理由は、複数の配列を含むmulti-FASTA形式ファイルへの対応を意識しているからです。

The screenshot shows the RStudio interface with the following components:

- Editor:** A text file named 'hoge1.txt' containing:

```
1 A C G T M R W S Y K V H D B N - + .
2 hoge 2 5 8 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3
```
- Console:** Shows the execution of R commands:

```
> hoge
GGTACGGTTCCGGTTGCCGA
```

```
> out
[1,] 2 5 8 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
> #ファイルに保存
> tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.name=s=F, col.names=T)#tmpの中身を指定したファイル名で保存
```
- Environment/Files:** Shows a file explorer view of the 'hoge' directory containing:
  - ..
  - .Rhistory (0 B)
  - hoge1.txt (80 B)
  - seq\_20.fasta (29 B)

# multi-FASTAファイル2

さきほどテンプレートとして用いた①例題1は、②multi-FASTAファイルを入力として実行するものです。③hoge4.faを作業ディレクトリにダウンロード。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#intro\_general\_kmer\_1\_biostrings

1. [イン트로](#) | [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:  
配列ごとに出現頻度をカウントした結果を返すやり方です。

```
in_f <- "hoge4.fa" #入力ファイル名を指定してin_fに格納
out_f <- "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
out <- alphabetFrequency(fasta) #A,C,G,T,..の数を各配列ごとにカウントした結果をoutに格納
out #outの中身を表示

#ファイルに保存
tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.names=T)#tmpの中身を指定して
```

[トップページ](#)

# multi-FASTAファイル3

さきほどテンプレートとして用いた①例題1は、②multi-FASTAファイルを入力として実行するものです。③hoge4.faを作業ディレクトリにダウンロード。③リロードすれば④が見られます。④をクリックして、⑤エディタで眺めているところ。

The screenshot shows the RStudio environment with a text editor open to a multi-FASTA file named 'hoge4.fa'. The file content is as follows:

```
1 >contig_1
2 CGGACAGCTCCTCGGCATCCGGAT
3 >contig_2
4 GTCTGCCTCAAGCGCCCCAAGTGGGTTTGGAGGCCTAACATCGCAAGTCG
5 ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCT
6 GTC
7 >contig_3
8 TGTAGGAGAAGGGCGGTATCAGCGTCCACTTACACGATCCGTTACTAATT
9 GTATGAGGTCGGGCA
10 >contig_4
11 CGTGCTGATTCCACACAGCAGTAAACGCGGACCTCTACCTATGAACATG
12
```

The R console at the bottom shows the following commands and their output:

```
> tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.names=T)#tmpの中身を指定したファイル名で保存
>
```

The file explorer on the right shows the directory structure: C:\Users\kadota\Desktop\hoge. The files listed are:

Name	Size
..	
.Rhistory	0 B
hoge1.txt	80 B
hoge4.fa	299 B
seq_20.fasta	29 B

Red arrows with circled numbers point to specific elements: ⑤ points to the 'Function' button in the toolbar; ③ points to the 'Refresh' button in the file explorer; ④ points to the 'hoge4.fa' file in the file explorer; ⑤ also points to the 'hoge4.fa' file in the file explorer.

# multi-FASTAファイル4

さきほどテンプレートとして用いた①例題1は、②multi-FASTAファイルを入力として実行するものです。③hoge4.faを作業ディレクトリにダウンロード。③リロードすれば④が見られます。④をクリックして、⑤エディタで眺めているところ。⑥「>」から始まる行が4つなので、このファイルは4配列です。Description部分が「contig\_1」や「contig\_2」と書かれているが、人工配列なので特に意味はない。

```
1 >contig_1
2 CGGACAGCTCGGGGCATCCGGAT
3 >contig_2
4 GTCTGCCTCAACCGCCCCAAGTGGGTTTGGAGGCCTAACATCGCAAGTCG
5 ACACTCAGTCCGGCCGCTGGTTGGCAGGGGCAGAGACCCAGCACACCCT
6 GTC
7 >contig_3
8 TGTAGGAGAAGCGCGGTATCAGCGTCCACTTACACGATCCGTTACTAATT
9 GTATGAGGTCCTCGCA
10 >contig_4
11 CGTGCTGATTCACACAGCAGTAAACGCGGACCTCTACCTATGAACATG
12
```

```
> tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.names=T)#tmpの中身を指定したファイル名で保存
>
```

Name	Size
..	
.Rhistory	0 B
hoge1.txt	80 B
hoge4.fa	299 B
seq_20.fasta	29 B

# multi-FASTAファイル5

さきほどテンプレートとして用いた①例題1は、②multi-FASTAファイルを入力として実行するものです。③hoge4.faを作業ディレクトリにダウンロード。③リロードすれば④が見られます。④をクリックして、⑤エディタで眺めているところ。⑥「>」から始まる行が4つなので、このファイルは4配列です。Description部分が「contig\_1」や「contig\_2」と書かれているが、人工配列なので特に意味はない。⑦のカーソル位置が、⑧11行50列目を意味する「11:50」となっていることがわかる。また、⑨の破線は10塩基ごとに示している。このことから、「contig\_1は24塩基、contig\_2は103塩基、contig\_3は65塩基、contig\_4は49塩基」だということが目視でわかります。

```
1 >contig_1
2 CGGACAGCTCCTCGGCATCCGGAT
3 >contig_2
4 GTCTGCCTCAAGCGCCCCAAGTGGGTTTGGAGGCCTAACATCGCAAGTCG
5 ACACTCAGTCCGGCCGCTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCT
6 GTC
7 >contig_3
8 TGTAGGAGAAGGGCGGTATCAGCGTCCACTTACACGATCCGTTACTAATT
9 GTATGAGGTCGGGCA
10 >contig_4
11 CGTGCTGATTCCACACAGCAGTAAACGCGGACCTCTACCTATGAACATG
12
```

```
> tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.name
s=F, col.names=T)#tmpの中身を指定したファイル名で保存
>
```

# multi-FASTAファイル6

例題1のテンプレートをコピー実行でもよいが、ここでは①のタブをアクティブにして、②入力ファイル名を元に戻して(hoge4.faに変更して)、全選択したのち、③再度実行。

The screenshot shows the RStudio interface with the following elements:

- Code Editor:** Contains R code for reading a multi-FASTA file and writing the output. The code is as follows:
 

```

1 in_f <- "hoge4.fa" #入力ファイル名を指
2 out_f <- "hoge1.txt" #出力ファイル名を指
3
4 #必要なパッケージをロード
5 library(Biostrings) #パッケージの読み込
6
7 #入力ファイルの読み込み
8 fasta <- readDNAStringSet(in_f, format="fasta")#in_fで指定
9
10 #本番
11 out <- alphabetFrequency(fasta) #A,C,G,T,...の数を各
12

```
- Console:** Shows the execution of the code:
 

```

> tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.name
s=F, col.names=T)#tmpの中身を指定したファイル名で保存
>

```
- Files Panel:** Shows the file structure in the current directory:
 

Name	Size
..	
.Rhistory	0 B
hoge1.txt	80 B
hoge4.fa	299 B
seq_20.fasta	29 B

# multi-FASTAファイル7

例題1のテンプレートをコピー実行でもよいが、ここでは①のタブをアクティブにして、②入力ファイル名を元に戻して(hoge4.faに変更して)、全選択したのち、③再度実行。実行後の状態。④出力ファイルのサイズが80 Bytesから234 Bytesに増加していることが分かります。

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
hoge1.txt hoge4.fa
Source on Save Run Source
7 #入力ファイルの読み込み
8 fasta <- readDNASTringSet(in_f, format="fasta")#in_fで指定
9
10 #本番
11 out <- alphabetFrequency(fasta) #A,C,G,T,..の数を各
12 out #outの中身を表示
13
14 #ファイルに保存
15 tmp <- cbind(names(fasta), out) #保存したい情報をtmp
16 write.table(tmp, out_f, sep="\t", append=F, quote=F, row.na
17 存
```

```
Console To Source
out #outの中身を表示
#ファイルに保存
tmp <- cbind(names(fasta), out...
write.table(tmp, out_f, sep="\t"
```

Files	Plots	Packages	Help	Viewer
New Folder	Delete	Rename	More	
C: > Users > kadota > Desktop > hoge				
Name		Size		
..		0 B		
.Rhistory		0 B		
hoge1.txt		234 B		
hoge4.fa		299 B		
seq_20.fasta		29 B		

```
Console Terminal Jobs
C:/Users/kadota/Desktop/hoge/
> tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.name
s=F, col.names=T)#tmpの中身を指定したファイル名で保存
> |
```



④

# multi-FASTAファイル8

例題1のテンプレートをコピー実行でもよいが、ここでは①のタブをアクティブにして、②入力ファイル名を元に戻して(hoge4.faに変更して)、全選択したのち、③再度実行。実行後の状態。④出力ファイルのサイズが80 Bytesから234 Bytesに増加していることがわかります。実は、④出力ファイルであるhoge1.txtを、⑤エディタで開いていたのですが、④をクリックすると(しなくても?!)最新の実行結果が反映されていることがわかります。

RStudio interface showing a multi-FASTA file being processed. The console shows R code for writing a table to a file. A red arrow points to the 'hoge1.txt' tab.

```

1   A C G T M R W S Y K V H D B N - + .
2 contig_1  4 9 7 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3 contig_2 20 34 31 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4 contig_3 16 13 20 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5 contig_4 14 15 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6

```

```

> tmp <- cbind(names(fasta), out)           #保存したい情報をtmpに格納
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.names=T)#tmpの中身を指定したファイル名で保存
>

```

Windows file explorer showing the directory 'hoge' with files: .Rhistory (0 B), hoge1.txt (234 B), hoge4.fa (299 B), and seq\_20.fasta (29 B). A red arrow points to hoge1.txt.

Name	Size
..	
.Rhistory	0 B
hoge1.txt	234 B
hoge4.fa	299 B
seq_20.fasta	29 B



# multi-FASTAファイル9

RStudio interface showing a multi-FASTA file being processed. The editor displays the following table:

	A	C	G	T	M	R	W	S	Y	K	V	H	D	B	N	-	+	.
1																		
2	contig_1	4	9	7	4	0	0	0	0	0	0	0	0	0	0	0	0	0
3	contig_2	20	34	31	18	0	0	0	0	0	0	0	0	0	0	0	0	0
4	contig_3	16	13	20	16	0	0	0	0	0	0	0	0	0	0	0	0	0
5	contig_4	14	15	10	10	0	0	0	0	0	0	0	0	0	0	0	0	0
6																		

The console shows the following R code:

```
> tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.names=T)#tmpの中身を指定したファイル名で保存
> |
```

例題1のテンプレートをコピー実行でもよいが、ここでは①のタブをアクティブにして、②入力ファイル名を元に戻して(hoge4.faに変更して)、全選択したのち、③再度実行。実行後の状態。④出力ファイルのサイズが80 Bytesから234 Bytesに増加していることがわかります。実は、④出力ファイルであるhoge1.txtを、⑤エディタで開いていたのですが、④をクリックすると(しなくても?!)最新の実行結果が反映されていることがわかります。実は⑥で見えるように、⑤が編集集中であることを示す「アスタリスク(\*)」がついていないおかげで上書きできるだけかもしれません。いずれにせよ「WinユーザがExcelで開いているときに遭遇したエラー」とは異なります。

# multi-FASTAファイル10

①エディタで眺めると、②列名部分が左にずれて見辛い。慣れれば③Console画面上の、(この場合はoutオブジェクトの)④表示結果を眺めるほうが見やすいかもしれません。

RStudio interface showing the multi-FASTA file processing workflow.

**Environment:** Console, History, Connections

**Files:** New Folder, Delete, Rename, More

**Files Panel:** .., .Rhistory (0 B), hoge1.txt (234 B), hoge4.fa (299 B), seq\_20.fasta (29 B)

**Editor Window (hoge4.fa):**

```

A C G T M R W S Y K V H D B N - + .
contig_1 4 9 7 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
contig_2 20 34 31 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
contig_3 16 13 20 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
contig_4 14 15 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    
```

**Console Window:**

```

> out #outの中身を表示
#ファイルに保存
tmp <- cbind(names(fasta), out...)
write.table(tmp, out_f, sep="\t"
    
```

**Terminal Window:**

```

> out #outの中身を表示
[1,] 4 9 7 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[2,] 20 34 31 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[3,] 16 13 20 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[4,] 14 15 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
>
> #ファイルに保存
> tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.name
s=F, col.names=T)#tmpの中身を指定したファイル名で保存
>
    
```

# Contents

- Introduction、出現頻度解析( $k=2$ )、出現頻度解析( $k=1$ )
- $k=1$ で実践、multi-FASTAファイル、他の例題を実行
- $k=2$ で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8( $k=2$ )、例題9( $k=3$ )、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11( $k=10$ )、例題12( $k=10$ )、シークエンスエラーを含む場合

# 他の例題を実行1

(ごくまれに塩基の表記で「A or Cを意味するM」や「A or Gを意味するR」などを見かけますが実質的に無視でもよいので…) 普段よく取り扱う「A, C, G, T, and N」の5種類に限定した結果を得るやり方が…

**RStudio Environment**

Environment | History | Connections

o Console | To Source

```
out #outの中身を表示
#ファイルに保存
tmp <- cbind(names(fasta), out...)
write.table(tmp, out_f, sep="\t"
```

**Files** | Plots | Packages | Help | Viewer

New Folder | Delete | Rename | More

C: > Users > kadota > Desktop > hoge

Name	Size
..	
.Rhistory	0 B
hoge1.txt	234 B
hoge4.fa	299 B
seq_20.fasta	29 B

**Editor**

```
1  A C G T M R W S Y K V H D B N - + .
2  contig_1 4 9 7 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3  contig_2 20 34 31 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4  contig_3 16 13 20 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5  contig_4 14 15 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

**Console**

```
> out #outの中身を表示
      A  C  G  T M R W S Y K V H D B N - + .
[1,]  4  9  7  4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[2,] 20 34 31 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[3,] 16 13 20 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[4,] 14 15 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
>
> #ファイルに保存
> tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.name
s=F, col.names=T)#tmpの中身を指定したファイル名で保存
> |
```

# 他の例題を実行2

(ごくまれに塩基の表記で「A or Cを意味するM」や「A or Gを意味するR」などを見かけますが実質的に無視でもよいので…) 普段よく取り扱う「A, C, G, T, and N」の5種類に限定した結果を得るやり方が、**①例題2**です。

(Rで)塩基配列解析

x +

← ① ↻ ⓘ 保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#int

## 2. [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:

1と基本的に同じで、出力結果を"A", "C", "G", "T", "N"のみに限定するやり方です。

```

in_f <- "hoge4.fa"           #入力ファイル名を指定してin_fに格納
out_f <- "hoge2.txt"        #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N") #出力させたい塩基を指定

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

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

#本番
hoge <- alphabetFrequency(fasta) #A,C,G,T,..の数を各配列ごとにカウントした結果をhogeに格納
obj <- is.element(colnames(hoge), param_base) #条件を満たすかどうかを判定した結果をobjに格納
obj
out <- hoge[, obj]         #objで指定した列のみ抽出した結果をoutに格納
out                        #outの中身を表示

#ファイルに保存
tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.names=T) #tmpの中身を指定して出力

```

[トップページ](#) ^

# 他の例題を実行3

(ごくまれに塩基の表記で「A or Cを意味するM」や「A or Gを意味するR」などを見かけますが実質的に無視でもよいので…) 普段よく取り扱う「A, C, G, T, and N」の5種類に限定した結果を得るやり方が、①例題2です。②このサイトは、必要に応じて変更する箇所を、③スクリプトの上部に集約している。

(Rで)塩基配列解析

②

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#int

2. 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mult

1と基本的に同じで、出力結果を"A", "C", "G", "T", "N"のみに限定するやり方

```

in_f <- "hoge4.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge2.txt" #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N") #出力させたい塩基を指定

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

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

#本番
hoge <- alphabetFrequency(fasta) #A,C,G,T,..の数を各配列ごとにカウントした結果をhogeに格納
obj <- is.element(colnames(hoge), param_base) #条件を満たすかどうかを判定した結果をobjに格納
obj #objの中身を表示
out <- hoge[, obj] #objで指定した列のみ抽出した結果をoutに格納
out #outの中身を表示

#ファイルに保存
tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.names=T) #tmpの中身を指定し

```

③

[トップページ](#) ^

# 他の例題を実行4

(ごくまれに塩基の表記で「A or Cを意味するM」や「A or Gを意味するR」などを見かけますが実質的に無視でもよいので…) 普段よく取り扱う「A, C, G, T, and N」の5種類に限定した結果を得るやり方が、①例題2です。②このサイトは、必要に応じて変更する箇所を、③スクリプトの上部に集約している。①例題2では、④のところで出力させたい塩基を指定している。

```

(Rで)塩基配列解析
← ①
②
保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#int
2. 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mult
1と基本的に同じで、出力結果を"A", "C", "G", "T", "N"のみに限定するやり方

in_f <- "hoge4.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge2.txt" #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N") #出力させたい塩基を指定

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

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

#本番
hoge <- alphabetFrequency(fasta) #A,C,G,T,..の数を各配列ごとにカウントした結果をhogeに格納
obj <- is.element(colnames(hoge), param_base) #条件を満たすかどうかを判定した結果をobjに格納
obj #objの中身を表示
out <- hoge[, obj] #objで指定した列のみ抽出した結果をoutに格納
out #outの中身を表示

#ファイルに保存
tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.names=T) #tmpの中身を指定して出力

```

④

[トップページ](#)

# 他の例題を実行5

(ごくまれに塩基の表記で「A or Cを意味するM」や「A or Gを意味するR」などを見かけますが実質的に無視でもよいので…) 普段よく取り扱う「A, C, G, T, and N」の5種類に限定した結果を得るやり方が、①例題2です。②このサイトは、必要に応じて変更する箇所を、③スクリプトの上部に集約している。①例題2では、④のところで出力させたい塩基を指定している。もちろん④で指定した文字列ベクトル情報を含む、⑤param\_baseが、⑥の部分で利用されているなどの違いはあります。

```

(Rで)塩基配列解析
< ① ②
保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#int
2. 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mult
1と基本的に同じで、出力結果を"A", "C", "G", "T", "N"のみに限定するやり方

in_f <- "hoge4.fa" #入力ファイル名を指定して読み込み
out_f <- "hoge2.txt" #出力ファイル名を指定して出力
param_base <- c("A", "C", "G", "T", "N") #出力させたい塩基を指定
#パッケージをロード
library(Biostrings) #パッケージの読み込み

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

#本番
hoge <- alphabetFrequency(fasta) #A,C,G,T,..の数を各配列ごとにカウントした結果をhogeに格納
obj <- is.element(colnames(hoge), param_base) #条件を満たすかどうかを判定した結果をobjに格納
obj #objの中身を表示
out <- hoge[, obj] #objで指定した列のみ抽出した結果をoutに格納
out #outの中身を表示

#ファイルに保存
tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
write.table(tmp, out_f, sep="\t", append=F, quote=F, new.names=F, col.names=T) #tmpの中身を指定して出力

```

[トップページ](#) ^



# 他の例題を実行6

(Rで)塩基配列解析

保護されていない通信 | [iu.a.u-tokyo.ac.jp/~kadota/r\\_seq.html#intro\\_general\\_kmer\\_1\\_bios...](http://iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#intro_general_kmer_1_bios...)

## 2. [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:

1と基本的に同じで、出力結果を"A", "C", "G", "T", "N"のみに限定するやり方です。

```

in_f <- "hoge4.fa"           #入力ファイル名を指定してin_fに格納
out_f <- "hoge2.txt"        #出力ファイル名を指定してout_fに格納
param_base <- c("A", "C", "G", "T", "N") #出力させたい塩基を指定

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

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

#本番
hoge <- alphabetFrequency(fasta) #A,C,G,T,..の数を各配列ごとにカウントした結果をhogeに格納
obj <- is.element(colnames(hoge), param_base) #条件を満たすかどうかを判定した結果をobjに格納
obj
out <- hoge[, obj]         #objで指定した列のみ抽出した結果をoutに格納
out                        #outの中身を表示

#ファイルに保存
tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.names=T) #tmpの中身を指定して格納
    
```

[トップページ](#)▲

①例題2をコピー実行。②Console画面  
上にコピー実行した直後の状態。

# 他の例題を実行7

The screenshot shows the RStudio interface. The top-left pane displays a text file with the following content:

```
1   A C G T M R W S Y K V H D B N - + .
2 contig_1  4 9 7 4 0 0 0 0 0 0 0 0 0 0 0 0 0
3 contig_2 20 34 31 18 0 0 0 0 0 0 0 0 0 0 0 0 0
4 contig_3 16 13 20 16 0 0 0 0 0 0 0 0 0 0 0 0 0
5 contig_4 14 15 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0
```

A red arrow with the number '2' points to the console window. The console window shows the following R code and its output:

```
>
> #ファイルに保存
> tmp <- cbind(names(fasta), out...)
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.names=T)#tmpの中身を指定したファイル名で保存
>
> |
```

The right-hand pane shows the file explorer with the following files:

Name	Size
..	
.Rhistory	0 B
hoge1.txt	234 B
hoge4.fa	299 B
seq_20.fasta	29 B

# 他の例題を実行8

①例題2をコピー実行。②Console画面上にコピー実行した直後の状態。③リロードして、④出力ファイル(hoge2.txt)をクリックして、⑤エディタ上で開いたところ。妥当な結果ですね。こんな感じでいくつかの例題を実行して、スクリプト間の違いや「スクリプト内で赤字で書いているコメント」を参考にして、コード内部でどのようなことが行われているかを理解していくとよいでしょう。ここまでがk-mer解析のk=1の場合でした。

The screenshot shows RStudio with a script in the editor and a console window. The script outputs a table of contig statistics. The console shows the execution of the script. The file explorer shows the output files, with 'hoge2.txt' highlighted.

```
1 | A C G T N
2 contig_1 4 9 7 4 0
3 contig_2 20 34 31 18 0
4 contig_3 16 13 20 16 0
5 contig_4 14 15 10 10 0
```

```
[1,] 4 9 7 4 0
[2,] 20 34 31 18 0
[3,] 16 13 20 16 0
[4,] 14 15 10 10 0
>
> #ファイルに保存
> tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.names=T)#tmpの中身を指定したファイル名で保存
>
> |
```

Name	Size
..	
.Rhistory	0 B
hoge1.txt	234 B
hoge2.txt	104 B
hoge4.fa	299 B
seq_20.fasta	29 B

# Contents

- Introduction、出現頻度解析( $k=2$ )、出現頻度解析( $k=1$ )
- $k=1$ で実践、multi-FASTAファイル、他の例題を実行
- $k=2$ で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8( $k=2$ )、例題9( $k=3$ )、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11( $k=10$ )、例題12( $k=10$ )、シークエンスエラーを含む場合

①次はk=2でk-mer出現頻度解析を行ってみます。

# k=2で実践1

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#

- イントロ | 一般 | 翻訳配列(translate)を取得 | [Biostrings](#) (last modified 2015/09/12)
- イントロ | 一般 | 翻訳配列(translate)を取得 | [seqinr\(Charif\\_2005\)](#) (last modified 2015/03/09)
- イントロ | 一般 | [相補鎖\(complement\)](#)を取得 (last modified 2019/03/10)
- イントロ | 一般 | [逆相補鎖\(reverse complement\)](#)を取得 (last modified 2019/03/10)
- イントロ | 一般 | [逆鎖\(reverse\)](#)を取得 (last modified 2019/03/10)
- イントロ | 一般 | k-mer解析 | k=1(塩基ごとの出現頻度解析) | [Biostrings](#) (last modified 2016/04/27)
- イントロ | 一般 | k-mer解析 | k=2(2連続塩基の出現頻度解析) | [Biostrings](#) (last modified 2016/01/28)
- イントロ | 一般 | k-mer解析 | k=3(3連続塩基の出現頻度解析) | [Biostrings](#) (last modified 2016/01/28)
- イントロ | 一般 | k-mer解析 | k=n(n連続塩基の出現頻度解析) | [Biostrings](#) (last modified 2016/05/01)
- 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 2017/04/11)
- イントロ | 一般 | 配列取得 | ゲノム配列 | [BSgenome](#) (last modified 2019/02/22)
- 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 2017/04/11)
- イントロ | 一般 | 配列取得 | プロモーター配列 | [BSgenomeとTxDbから](#) (last modified 2015/02 [トップページへ](#))
- イントロ | 一般 | 配列取得 | プロモーター配列 | [GenomicFeatures\(Lawrence\\_2013\)](#) (last modified

①次はk=2でk-mer出現頻度解析を行ってみます。②例題1をコピー実行。

# k=2で実践2

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#intro\_general\_kmer\_2\_bios...

## イントロ | 一般 | k-mer解析 | k=2(2連続塩基の出現頻度解析) | Biostrings

Biostringsパッケージを用いて、multi-FASTA形式ファイルを読み込んで、"AA", "AC", "AG", "AT", "CA", "CC", "CG", "CT", "GA", "GC", "GG", "GT", "TA", "TC", "TG", "TT" の計 $4^2 = 16$ 通りの2連続塩基の出現頻度を調べるやり方を示します。k-mer解析のk=2の場合に相当します。ヒトゲノムで"CG"の割合が期待値よりも低い(Lander et al., 2001; Saxonov et al., 2006)ですが、それを簡単に検証できます。

「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ー。

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:

タイトル通りの出現頻度です。

```

in_f <- "hoge4.fa"           #入力ファイル名を指定してin_fに格納
out_f <- "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
fasta                          #確認してるだけです

```

[トップページ](#) ^

# k=2で実践3

①次はk=2でk-mer出現頻度解析を行ってみます。②例題1をコピー実行。実行後に、③出力ファイル(hoge1.txt)をクリックして、④エディタ上で開いたところ。

**Environment** | **History** | **Connections**

```

out <- dinucleotideFrequency(fasta)
#ファイルに保存
tmp <- cbind(names(fasta), out)
write.table(tmp, out_f, sep="\t"

```

**Files** | **Plots** | **Packages** | **Help** | **Viewer**

C:\Users\kadota\Desktop\hoge

Name	Size
..	
.Rhistory	0 B
hoge1.txt	221 B
hoge2.txt	104 B
hoge4.fa	299 B
seq_20.fasta	29 B

**Console** | **Terminal** | **Jobs**

```

C:/Users/kadota/Desktop/hoge/
> #本番
> out <- dinucleotideFrequency(fasta) #連続塩基の出現頻度情報をoutに格納
>
> #ファイルに保存
> tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F) #tmpの中身を指定したファイル名で保存
>
>

```

3:1 | Text File

# k=2で実践4

①次はk=2でk-mer出現頻度解析を行ってみます。②例題1をコピー実行。実行後に、③出力ファイル(hoge1.txt)をクリックして、④エディタ上で開いたところ。例えば⑤は、contig\_1の配列中に⑥TCという2連続塩基が3個あった、と解釈する。

RStudio interface showing the results of a k=2 dinucleotide frequency analysis. The main editor displays a matrix of dinucleotide frequencies for four contigs. A red arrow labeled '5' points to the value '3' in the 'GC' column for 'contig\_1'. Another red arrow labeled '6' points to the 'TC' column header.

	AA	AC	AG	AT	CA	CC	CG	CT	GC	GG	GT	TA	TC				
1	AA	AC	AG	AT	CA	CC	CG	CT	GC	GG	GT	TA	TC				
	TG	TT															
2	contig_1	0	1	1	2	2	2	3	2	2	2	3	0	0			
3	contig_2	4	6	9	1	11	11	5	6	4	9	10	8	1	8	6	3
4	contig_3	2	4	5	4	4	2	5	2	4	3	7	6	6	4	3	3
5	contig_4	3	6	2	3	5	3	3	4	3	3	1	2	3	2	4	1

```
> #本番
> out <- dinucleotideFrequency(fasta) #連続塩基の出現頻度情報をoutに格納
>
> #ファイルに保存
> tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)#tmpの中身を指定したファイル名で保存
>
>
```

Environment History Connections  
o Console To Source  
out <- dinucleotideFrequency(f...  
#ファイルに保存  
tmp <- cbind(names(fasta), out...  
write.table(tmp, out\_f, sep="\t...

Files Plots Packages Help Viewer  
New Folder Delete Rename More  
C:\Users\kadota\Desktop\hoge  
Name Size  
..  
.Rhistory 0 B  
hoge1.txt 221 B  
hoge2.txt 104 B  
hoge4.fa 299 B  
seq\_20.fasta 29 B



# k=2で実践5

①次はk=2でk-mer出現頻度解析を行ってみます。②例題1をコピー実行。実行後に、③出力ファイル(hoge1.txt)をクリックして、④エディタ上で開いたところ。例えば⑤は、contig\_1の配列中に⑥TCという2連続塩基が3個あった、と解釈する。⑦入力ファイル(hoge4.fa)中の、⑧contig\_1を眺めると、確かにTCが3つ存在しますね。

```
RStudio  
File Edit Code View Plots Session Build Debug Profile Tools Help  
Go to file/function  
hoge1.txt x hoge2.txt x hoge4.fa x  
1 >contig_1  
2 CGGACAGCTCCTCGGCATCCGGAT  
3 >contig_2  
4 GTCTGCCTCAAGCGCCCCAAGTGGGTTTGGAGGCCTAACATCGCAAGTCG  
5 ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCT  
6 GTC
```

```
Console Terminal x Jobs x  
C:/Users/kadota/Desktop/hoge/  
> #本番  
> out <- dinucleotideFrequency(fasta) #連続塩基の出現頻度情報をoutに格納  
>  
> #ファイルに保存  
> tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納  
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F) #tmpの中身を指定したファイル名で保存  
>  
> |
```

```
tmp <- cbind(names(fasta), out...  
write.table(tmp, out_f, sep="\t...  
Files Plots Packages Help Viewer  
New Folder Delete Rename More  
C: > Users > kadota > Desktop > hoge  
Name Size  
..  
.Rhistory 0 B  
hoge1.txt 221 B  
hoge2.txt 104 B  
hoge4.fa 299 B  
seq_20.fasta 29 B
```

# Contents

- Introduction、出現頻度解析( $k=2$ )、出現頻度解析( $k=1$ )
- $k=1$ で実践、multi-FASTAファイル、他の例題を実行
- $k=2$ で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8( $k=2$ )、例題9( $k=3$ )、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11( $k=10$ )、例題12( $k=10$ )、シークエンスエラーを含む場合

# 関数マニュアル1

「①さきほどk=2で実行した例題1 (dinucleotideFrequency関数を使用)」と「その前にk=1で実行した例題1 (alphabetFrequency関数を使用)」とは、②で利用している関数のみが異なります。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#intro\_general\_kmer\_2\_bios...

## 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:

タイトル通りの出現頻度です。

```
in_f <- "hoge4.fa"           #入力ファイル名を指定してin_fに格納
out_f <- "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
fasta                                #確認してるだけです

#本番
out <- dinucleotideFrequency(fasta)  #連続塩基の出現頻度情報をoutに格納

#ファイルに保存
tmp <- cbind(names(fasta), out)      #保存したい情報をtmpに格納
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)#tmpの中身を指定したファイル名で保
```

[トップページ](#)

# 関数マニュアル2

「①さきほどk=2で実行した例題1 (dinucleotideFrequency関数を使用)」と「その前にk=1で実行した例題1 (alphabetFrequency関数を使用)」とは、②で利用している関数のみが異なります。関数が異なると当然役割も異なります。RStudioでは関数名の最初の数文字分を手入力すれば該当する候補が表示されるので、目的の関数のマニュアルを眺めることができます。ここでは②の関数マニュアルを眺めるべく、③「dinu」まで入力してみます。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#int

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mult  
タイトル通りの出現頻度です。

```
in_f <- "hoge4.fa" #入力ファイル名を指定して読み込み
out_f <- "hoge1.txt" #出力ファイル名を指定して書き込み

#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
out <- dinucleotideFrequency(fasta) #連続塩基の出現頻度情報をoutに格納

#ファイルに保存
tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F) #tmpの中身を指定したファイル名で保
```

[トップページ](#)△

# 関数マニュアル3

「①さきほどk=2で実行した例題1 (dinucleotideFrequency関数を使用)」と「その前にk=1で実行した例題1 (alphabetFrequency関数を使用)」とは、②で利用している関数のみが異なります。関数が異なると当然役割も異なります。RStudioでは関数名の最初の数文字分を手入力すれば該当する候補が表示されるので、目的の関数のマニュアルを眺めることができます。ここでは②の関数マニュアルを眺めるべく、③「dinu」まで入力してみます。④Console画面上で、⑤dinuまで打ち込むと…

```
RStudio  
File Edit Code View Plots Session Build Debug Profile Tools Help  
Go to file/function Addins  
Untitled1* x hoge1.txt x hoge2.txt x hoge4.fa x  
1 >contig_1  
2 CGGACAGCTCCTCGGCATCCGGAT  
3 >contig_2  
4 GTCTGCCTCAAGCGCCCCAAGTGGGTTTGGAGGCCTAACATCGCAAGTCG  
5 ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCT  
6 GTC
```

```
Console Terminal x Jobs x  
C:/Users/kadota/Desktop/hoge/ ↗  
> #本番  
> out <- dinucleotideFrequency(fasta) #連続塩基の出現頻度情報をoutに格納  
>  
> #ファイルに保存  
> tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納  
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)#tmpの中身を指定したファイル名で保存  
>  
> dinu|
```

	Name	Size
<input type="checkbox"/>	..	
<input type="checkbox"/>	.Rhistory	0 B
<input type="checkbox"/>	hoge1.txt	221 B
<input type="checkbox"/>	hoge2.txt	104 B
<input type="checkbox"/>	hoge4.fa	299 B
<input type="checkbox"/>	seq_20.fasta	29 B

# 関数マニュアル4

「①さきほどk=2で実行した例題1 (dinucleotideFrequency関数を使用)」と「その前にk=1で実行した例題1 (alphabetFrequency関数を使用)」とは、②で利用している関数のみが異なります。関数が異なると当然役割も異なります。RStudioでは関数名の最初の数文字分を手入力すれば該当する候補が表示されるので、目的の関数のマニュアルを眺めることができます。ここでは②の関数マニュアルを眺めるべく、③「dinu」まで入力してみます。④Console画面上で、⑤dinuまで打ち込むと、こんな感じで(この場合は)2つの候補がリストアップされます。

```

1 >contig_1
2 CGGACAGCTCCTCGGCATCCGGAT
3 >contig_2
4 GTCTGCCTCAAGCGCCCCAAGTGGGTTTGGAGGCCTAACATCGCAAGTCG
5 ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCT
6 GTC
    
```

```

> #本番
> out <- dinucleotideFrequency(fasta) #連続
utに格納
>
> #ファイルに保存
> tmp <- cbind(names(fasta), out) #保存
> write.table(tmp, out, filesep="\t", append=F)
> dinucleotideFrequency {Biostrings}
> dinucleotideFrequencyTest {Biostrings}
> dinu
    
```

```

dinucleotideFrequency(x, step = 1, as.prob = FALSE,
  as.matrix = FALSE, fast.moving.side = "right",
  with.labels = TRUE, ...)
    
```

Given a DNA or RNA sequence (or a set of DNA or RNA sequences), the oligonucleotideFrequency function computes the frequency of all possible oligonucleotides of a given length (called the "width" in this particular context) in a sliding window that is shifted step nucleotides at a time.

Press F1 for additional help

221 B
104 B
299 B
29 B

# 関数マニュアル5

今赤枠内で見えているのは、①でハイライトされているdinucleotideFrequency関数の概要。

```
dinucleotideFrequency(x, step = 1, as.prob = FALSE,
  as.matrix = FALSE, fast.moving.side = "right",
  with.labels = TRUE, ...)
```

Given a DNA or RNA sequence (or a set of DNA or RNA sequences), the oligonucleotideFrequency function computes the frequency of all possible oligonucleotides of a given length (called the "width" in this particular context) in a sliding window that is shifted step nucleotides at a time.

Press F1 for additional help

The screenshot shows the RStudio environment. The console window contains the following R code:

```
out <- dinucleotideFrequency(fasta)
#ファイルに保存
tmp <- cbind(names(fasta), out...)
write.table(tmp, out_f, sep="\\"
```

The file explorer window shows the directory structure: C:\Users\kadota\Desktop\hoge. A file named 'Rhistory' is visible with a size of 29 B.

```
6 GTC
11:50
Console Terminal x Jobs x
C:/Users/kadota/Desktop/hoge/
> #本番
> out <- dinucleotideFrequency(fasta) #連続
utに格納
>
> #ファイルに保存
> tmp <- cbind(names(fasta), out) #保存
> write.table(tmp, out_f, sep="\\"
s: ◆ dinucleotideFrequency {Biostrings}
> ◆ dinucleotideFrequencyTest {Biostrings}
> dinu
```

dinucleotideFrequency(x, step = 1, as.prob = FALSE,
 as.matrix = FALSE, fast.moving.side = "right",
 with.labels = TRUE, ...)

Given a DNA or RNA sequence (or a set of DNA or RNA sequences), the oligonucleotideFrequency function computes the frequency of all possible oligonucleotides of a given length (called the "width" in this particular context) in a sliding window that is shifted step nucleotides at a time.

Press F1 for additional help



# 関数マニュアル6

今赤枠内で見えているのは、①でハイライトされているdinucleotideFrequency関数の概要。②に相当する情報として実際に与えているのが、③fastaというオブジェクト。

```
dinucleotideFrequency(x, step = 1, as.prob = FALSE, as.matrix = FALSE, fast.moving.side = "right", with.labels = TRUE, ...)
```

Given a DNA or RNA sequence (or a set of DNA or RNA sequences), the oligonucleotideFrequency function computes the frequency of all possible oligonucleotides of a given length (called the "width" in this particular context) in a sliding window that is shifted step nucleotides at a time.

Press F1 for additional help

The screenshot shows the RStudio environment. The console window contains the following R code:

```
out <- dinucleotideFrequency(fasta)
#ファイルに保存
tmp <- cbind(names(fasta), out...)
write.table(tmp, out_f, sep="\\"
```

The file explorer window shows a directory structure with files of various sizes:

Name	Size
..	0 B
Rhistory	221 B
	104 B
	299 B
	29 B

```
6 GTC
11:50
Console Terminal x Jobs x
C:/Users/kadota/Desktop/hoge/
> #本番
> out <- dinucleotideFrequency(fasta) #連続
utに格納
>
> #ファイルに保存
> tmp <- cbind(names(fasta), out) #保存
> write.table(tmp, out_f, sep="\\"
s: ◆ dinucleotideFrequency {Biostrings}
> ◆ dinucleotideFrequencyTest {Biostrings}
> dinu
```

```
dinucleotideFrequency(x, step = 1, as.prob = FALSE, as.matrix = FALSE, fast.moving.side = "right", with.labels = TRUE, ...)
```

Given a DNA or RNA sequence (or a set of DNA or RNA sequences), the oligonucleotideFrequency function computes the frequency of all possible oligonucleotides of a given length (called the "width" in this particular context) in a sliding window that is shifted step nucleotides at a time.

Press F1 for additional help



# 関数マニュアル7

今赤枠内で見えているのは、①でハイライトされているdinucleotideFrequency関数の概要。②に相当する情報として実際に与えているのが、③fastaというオブジェクト。③は、④入力ファイル名情報の文字列である、⑤in\_fを与えて読み込んだ情報。つまり実質的に**入力ファイルの中身**だということです。

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mult

⑤ ④ ③

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

#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

```
#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")
fasta
```

#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

```
#本番
```

```
out <- dinucleotideFrequency(fasta)
```

#連続塩基の出現頻度情報をoutに格納

```
#ファイルに保存
```

```
tmp <- cbind(names(fasta), out)
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)
```

#保存したい情報をtmpに格納

#tmpの中身を指定したファイル名で保

[トップページ](#)

# 関数マニュアル8

```
dinucleotideFrequency(x, step = 1, as.prob = FALSE,
  as.matrix = FALSE, fast.moving.side = "right",
  with.labels = TRUE, ...)
```

Given a DNA or RNA sequence (or a set of DNA or RNA sequences), the `oligonucleotideFrequency` function computes the frequency of all possible oligonucleotides of a given length (called the "width" in this particular context) in a sliding window that is shifted `step` nucleotides at a time.

Press F1 for additional help

6 GTC

11:50

Console Terminal x Jobs x

C:/Users/kadota/Desktop/hoge/

```
> #本番
> out <- dinucleotideFrequency(fasta) #連続
outに格納
>
> #ファイルに保存
> tmp <- cbind(names(fasta), out) #保存
> write.table(tmp, out.f.sep="\t", append=F,
s: ◆ dinucleotideFrequency {Biostrings}
> ◆ dinucleotideFrequencyTest {Biostrings}
> dinu
```

今赤枠内で見えているのは、①でハイライトされている`dinucleotideFrequency`関数の概要。②に相当する情報として実際に与えているのが、③`fasta`というオブジェクト。③は、④入力ファイル名情報の文字列ベクトルである、⑤`in_f`を与えて読み込んだ情報。つまり実質的に入力ファイルの中身だということです。例題1のスク립トでは、`dinucleotideFrequency`関数の入力として③`fasta`オブジェクトしか与えていませんが、④の赤枠で示されているように多数のオプションが利用可能であることが分かります。

```
dinucleotideFrequency(x, step = 1, as.prob = FALSE,
  as.matrix = FALSE, fast.moving.side = "right",
  with.labels = TRUE, ...)
```

Given a DNA or RNA sequence (or a set of DNA or RNA sequences), the `oligonucleotideFrequency` function computes the frequency of all possible oligonucleotides of a given length (called the "width" in this particular context) in a sliding window that is shifted `step` nucleotides at a time.

Press F1 for additional help

Name	Size
..	0 B
Rhistory	221 B
	104 B
	299 B
	29 B

# 関数マニュアル9



```
dinucleotideFrequency(x, step = 1, as.prob = FALSE,
  as.matrix = FALSE, fast.moving.side = "right",
  with.labels = TRUE, ...)
```

Given a DNA or RNA sequence (or a set of DNA or RNA sequences), the `oligonucleotideFrequency` function computes the frequency of all possible oligonucleotides of a given length (called the "width" in this particular context) in a sliding window that is shifted `step` nucleotides at a time.

Press F1 for additional help

```
6 GTC
11:50
Console Terminal x Jobs x
C:/Users/kadota/Desktop/hoge/
> #本番
> out <- dinucleotideFrequency(fasta) #連続
utに格納
>
> #ファイルに保存
> tmp <- cbind(names(fasta), out) #保存
> write.table(tmp, out.f, sep="\t", append=F)
> dinucleotideFrequency {Biostrings}
> dinucleotideFrequencyTest {Biostrings}
> dinu
```

今赤枠内で見えているのは、①でハイライトされている `dinucleotideFrequency` 関数の概要。②に相当する情報として実際に与えているのが、③ `fasta` というオブジェクト。③は、④入力ファイル名情報の文字列ベクトルである、⑤ `in_f` を与えて読み込んだ情報。つまり実質的に入力ファイルの中身だということです。例題1のスクリプトでは、`dinucleotideFrequency` 関数の入力として③ `fasta` オブジェクトしか与えていませんが、④の赤枠で示されているように多数のオプションが利用可能であることが分かります。例えば⑤ `as.prob` オプションは「(出現頻度ではなく)出現確率として(`as probability`)結果を返すかどうか」を指定するものです。デフォルトは `FALSE` なので「出現確率として返さない(つまり出現頻度として返す)」と宣言していることに相当します。これを「`as.prob = TRUE` (出現確率として返す)」としたものが、この項目の例題2です。

# Contents

- Introduction、出現頻度解析( $k=2$ )、出現頻度解析( $k=1$ )
- $k=1$ で実践、multi-FASTAファイル、他の例題を実行
- $k=2$ で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8( $k=2$ )、例題9( $k=3$ )、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11( $k=10$ )、例題12( $k=10$ )、シーケンスエラーを含む場合

k=2の①例題2。例題1との違いは、結果を出現確率として返すべく、②as.prob=Tを追加しただけです。コピペ実行。

# 例題2を実行1

(Rで)塩基配列解析

保護されていない通信 | [iu.a.u-tokyo.ac.jp/~kadota/r\\_seq.html#intro\\_general\\_kmer\\_2\\_bios...](http://iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#intro_general_kmer_2_bios...)

2. [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:

出現頻度ではなく、出現確率を得るやり方です。

```
in_f <- "hoge4.fa"           #入力ファイル名を指定してin_fに格納
out_f <- "hoge2.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
fasta                                     #確認してるだけです

#本番
out <- dinucleotideFrequency(fasta, as.prob=T)#連続塩基の出現確率情報をoutに格納

#ファイルに保存
tmp <- cbind(names(fasta), out)          #保存したい情報をtmpに格納
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)#tmpの中身を指定したファイル名で保
```

[トップページ](#)↑

# 例題2を実行2

k=2の①例題2。例題1との違いは、結果を出現確率として返すべく、②as.prob=Tを追加しただけです。コピペ実行。実行後に、③出力ファイル(hoge2.txt)をクリックして、④エディタ上で開いたところ。確かに出現頻度ではなく、出現確率になっていることがわかります。これは配列長の長さの違いを補正していること(つまり正規化)と同義です。

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

function Addins

Untitled1\* x hoge1.txt x hoge2.txt x hoge4.fa x

```

1  AA AC AG AT CA CC CG CT GA GC GG GT TA TC
   TG TT
2  contig_1 0 0.0434782608695652 0.0434782608695652 0
   .0869565217391304 0.0869565217391304 0.0869565217391304
   0.130434782608696 0.0869565217391304 0.0869565217391304
   0.0869565217391304 0.130434782608696 0 0 0.1304347826086
   96 0 0
3  contig_2 0.0392156862745098 0.0588235294117647 0
   .0882352941176471 0.00980392156862745 0.107843137254902 0
   .107843137254902 0.0490196078431373 0.0588235294117647 0
   .0392156862745098 0.0882352941176471 0.0980392156862745
   0.0784313725490196 0.00980392156862745 0.078431372549019
   6 0.0588235294117647 0.0294117647058824
    
```

1:1 Text File

Console Terminal x Jobs x

```

C:/Users/kadota/Desktop/hoge/
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F) #tmpの中身を指定したファイル名で保存
>
>
    
```

```
tmp <- cbind(names(fasta), out...)
write.table(tmp, out_f, sep="\t"
```

Files Plots Packages Help Viewer

New Folder Delete Rename More

C: > Users > kadota > Desktop > hoge

	Name	Size
	..	
	.Rhistory	0 B
	hoge1.txt	221 B
	hoge2.txt	945 B
	hoge4.fa	299 B
	seq_20.fasta	29 B

# 例題2を実行3

k=2の①例題2のコードは、②入力ファイルを変更するだけで、例えばヒトゲノムの染色体ごとの2連続塩基の出現頻度解析を行うことも可能です。

(Rで)塩基配列解析

保護されていない通信 | [iu.a.u-tokyo.ac.jp/~kadota/r\\_seq.html#intro\\_general\\_kmer\\_2\\_bios...](http://iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#intro_general_kmer_2_bios...)

2. [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:

出現頻度ではなく、出現確率を得るやり方です。

```
in_f <- "hoge4.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge2.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
fasta #確認してるだけです

#本番
out <- dinucleotideFrequency(fasta, as.prob=T) #連続塩基の出現確率情報をoutに格納

#ファイルに保存
tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F) #tmpの中身を指定したファイル名で保
```

[トップページ](#)↑

# Contents

- Introduction、出現頻度解析(k=2)、出現頻度解析(k=1)
- k=1で実践、multi-FASTAファイル、他の例題を実行
- k=2で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8(k=2)、例題9(k=3)、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11(k=10)、例題12(k=10)、シークエンスエラーを含む場合
- 課題



# 例題7を実行1

k=2の①例題7。例題2では入力としてFASTA形式ファイルを与えたが、②ここではインストール済みのヒトゲノム情報を含むRパッケージを与えている。手順通りにR本体およびパッケージ群のインストール作業を行っていれば、特に意識せずともインストールされています。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#int

## 7. ヒトゲノム配列パッケージ(BSgenome.Hsapiens.NCBI.GRCh38)の場合

2013年12月にリリースされたGenome Reference Consortium GRCh38です。出力は出現確率です。

```

out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_bsgenome <- "BSgenome.Hsapiens.NCBI.GRCh38" #パッケージ名を指定(BSgenome系のゲノムパッケージ)

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

#前処理(指定したパッケージ中のオブジェクト名をgenomeに統一)
tmp <- ls(paste("package", param_bsgenome, sep=":")) #指定したパッケージで利用可能なオブジェクト名を取
genome <- eval(parse(text=tmp)) #文字列tmpをRオブジェクトとしてgenomeに格納(パッケージ中にオブ
fasta <- getSeq(genome) #ゲノム塩基配列情報を抽出した結果をfastaに格納
names(fasta) <- seqnames(genome) #description情報を追加している
fasta #確認してるだけです

#本番
out <- dinucleotideFrequency(fasta, as.prob=T) #連続塩基の出現確率情報をoutに格納

#ファイルに保存
tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納

```

[トップページ](#)へ

# 例題7を実行2

k=2の①例題7。例題2では入力としてFASTA形式ファイルを与えたが、②ここではインストール済みのヒトゲノム情報を含むRパッケージを与えている。手順通りにR本体およびパッケージ群のインストール作業を行っていれば、特に意識せずともインストールされています。FASTA形式ファイルにしていない理由は、このパッケージは圧縮されているので700MB程度で済んでいますが、非圧縮FASTA形式ファイルだと約3GBだから。

```

(Rで)塩基配列解析
7. ヒトゲノム配列パッケージ(BSgenome.Hsapiens.NCBI.GRCh38)の場合
2013年12月にリリースされたGenome Reference Consortium GRCh38で

out_f <- "hoge7.txt" #出力ファイル名を指定して
param_bsgenome <- "BSgenome.Hsapiens.NCBI.GRCh38" #パッケージ名を

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

#前処理(指定したパッケージ中のオブジェクト名をgenomeに統一)
tmp <- ls(paste("package", param_bsgenome, sep=":")) #指定したパッケージで利用可能なオブジェクト名を取
genome <- eval(parse(text=tmp)) #文字列tmpをRオブジェクトとしてgenomeに格納(パッケージ中にオブ
fasta <- getSeq(genome) #ゲノム塩基配列情報を抽出した結果をfastaに格納
names(fasta) <- seqnames(genome) #description情報を追加している
fasta #確認してるだけです

#本番
out <- dinucleotideFrequency(fasta, as.prob=T) #連続塩基の出現確率情報をoutに格納

#ファイルに保存
tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納

```



[トップページ](#)↑

# 例題7を実行3

```
(Rで)塩基配列解析 × +
← ① 保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#int
7. ヒトゲノム配列パッケージ(BSgenome.Hsapiens.NCBI.GRCh38)の場合
2013年12月にリリースされたGenome Reference Consortium GRCh38で
out_f <- "hoge7.txt" #出力ファイル名を指定して
param_bsgenome <- "BSgenome.Hsapiens.NCBI.GRCh38" #パッケージ名を
#必要なパッケージをロード
library(Biostings) #パッケージの読み込み
library(param_bsgenome, character.only=T) #指定したパッケージの読み込み
#前処理(指定したパッケージ中のオブジェクト名をgenomeに統一)
tmp <- ls(paste("package", param_bsgenome, sep=":")) #指定したパッケージのオブジェクト名を抽出
genome <- eval(parse(text=tmp)) #文字列tmpをRオブジェクトに変換
fasta <- getSeq(genome) #ゲノム塩基配列情報を抽出
names(fasta) <- seqnames(genome) #description情報を追加している
fasta #確認してるだけです
#本番
out <- dinucleotideFrequency(fasta, as.prob=T) #連続塩基の出現確率情報をoutに格納
#ファイルに保存
tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
```

k=2の①例題7。例題2では入力としてFASTA形式ファイルを与えたが、②ここではインストール済みのヒトゲノム情報を含むRパッケージを与えている。手順通りにR本体およびパッケージ群のインストール作業を行っていれば、特に意識せずともインストールされています。FASTA形式ファイルにしていない理由は、このパッケージは圧縮されているので700MB程度で済んでいますが、非圧縮FASTA形式ファイルだと約3GBだから。③赤枠内がヒトゲノムパッケージから目的のゲノム配列情報のfastaオブジェクトを得ているところだが、細かいところは気にしなくてもよい。

[トップページ](#)△

# 例題7を実行4

```

(Rで)塩基配列解析
7. ヒトゲノム配列パッケージ(BSgenome.Hsapiens.NCBI.GRCh38)の場合
2013年12月にリリースされたGenome Reference Consortium GRCh38で

out_f <- "hoge7.txt" #出力ファイル名を指定して
param_bsgenome <- "BSgenome.Hsapiens.NCBI.GRCh38" #パッケージ名を
#必要なパッケージをロー
library(Biostings) #パッケージの読み込み
library(param_bsgenome, character.only=T) #指定したパッケージの読み込み
#前処理(指定したパッケージ中のオブジェクト名をgenomeに統一)
tmp <- ls(paste("package", param_bsgenome, sep=":")) #指定したパッケージのオブジェクト名を抽出
genome <- eval(parse(text=tmp)) #文字列tmpをRオブジェクトに変換
fasta <- getSeq(genome) #ゲノム塩基配列情報を抽出
names(fasta) <- seqnames(genome) #description情報を追加
fasta #確認してるだけです

#本番
out <- dinucleotideFrequency(fasta, as.prob=T) #連続塩基の出現確率を計算

#ファイルに保存
tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納

```

k=2の①例題7。例題2では入力としてFASTA形式ファイルを与えたが、②ここではインストール済みのヒトゲノム情報を含むRパッケージを与えている。手順通りにR本体およびパッケージ群のインストール作業を行っていれば、特に意識せずともインストールされています。FASTA形式ファイルにしていない理由は、このパッケージは圧縮されているので700MB程度で済んでいますが、非圧縮FASTA形式ファイルだと約3GBだから。③赤枠内がヒトゲノムパッケージから目的のゲノム配列情報のfastaオブジェクトを得ているところだが、細かいところは気にしなくてもよい。④BSgenomeとかゲノム配列情報を含むパッケージに関する説明は、⑤2018年の講義資料中にあります。

[http://www.iu.a.u-tokyo.ac.jp/~kadota/20180507\\_kadota.pdf](http://www.iu.a.u-tokyo.ac.jp/~kadota/20180507_kadota.pdf)

k=2の①例題7を実行すると、ヒトゲノムの染色体ごとの2連続塩基の出現確率が、②hoge7.txtに保存されます。コピペ実行。

# 例題7を実行5

(Rで)塩基配列解析 × +

← ① → ↻ ⓘ 保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#intro\_general\_kmer\_2\_bios... ゲスト

## 7. ヒトゲノム配列パッケージ(BSgenome.Hsapiens.NCBI.GRCh38)の場合 :

2013年12月にリリースされたGenome Reference Consortium GRCh38です。出力は出現確率です。

```

out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_bsgenome <- "BSgenome.Hsapiens.NCBI.GRCh38" #パッケージ名を指定(BSgenome系のゲノムパッケージ)

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

#前処理(指定したパッケージ中のオブジェクト名をgenomeに統一)
tmp <- ls(paste("package", param_bsgenome, sep=":")) #指定したパッケージで利用可能なオブジェクト名を取
genome <- eval(parse(text=tmp)) #文字列tmpをRオブジェクトとしてgenomeに格納(パッケージ中にオブ
fasta <- getSeq(genome) #ゲノム塩基配列情報を抽出した結果をfastaに格納
names(fasta) <- seqnames(genome) #description情報を追加している
fasta #確認してるだけです

#本番
out <- dinucleotideFrequency(fasta, as.prob=T) #連続塩基の出現確率情報をoutに格納

#ファイルに保存
tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納

```

[トップページ](#) ↑

# 例題7を実行6

k=2の①例題7を実行すると、ヒトゲノムの染色体ごとの2連続塩基の出現確率が、②hoge7.txtに保存されます。コピペ実行。もしコピペ実行結果の③Console画面の最後のほうが、④のような感じでcontig\_4が見えるヒトは、例題7の実行(ヒトゲノムの解析)がうまくいっていない。

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
Untitled1* x hoge1.txt x hoge2.txt x hoge4.fa x
1 AA AC AG AT CA CC CG CT GA GC GG GT TA TC
  TG TT
2 contig_1 0 0.0434782608695652 0.0434782608695652 0
  .0869565217391304 0.0869565217391304 0.0869565217391304
3] 65 TGTAGGAGAAGGGC...ATGAGGTCGGGCA contig_3
4] 49 CGTGCTGATTCCAC...ACCTATGAACATG contig_4
>
> #本番
> out <- dinucleotideFrequency(fasta, as.prob=T) #連続塩基の出現確率情報をoutに格納
>
> #ファイルに保存
> tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F) #tmpの中身を指定したファイル名で保存
>
>
  
```

```

Console
out <- dinucleotideFrequency(f...
#ファイルに保存
tmp <- cbind(names(fasta), out...
write.table(tmp, out_f, sep="\...
  
```

Name	Size
..	
hoge1.txt	221 B
hoge2.txt	945 B
hoge4.fa	299 B
hoge7.txt	945 B
seq_20.fasta	29 B

# 例題7を実行7

k=2の①例題7を実行すると、ヒトゲノムの染色体ごとの2連続塩基の出現確率が、②hoge7.txtに保存されます。コピペ実行。もしコピペ実行結果の③Console画面の最後のほうが、④のような感じで contig\_4が見えるヒトは、例題7の実行(ヒトゲノムの解析)がうまくいっていない。⑤上のほうにスクロールすると、おそらく⑥のような感じで当該パッケージが存在しない、というエラーメッセージが見られると思います。これは手順通りにインストール作業ができていないことを意味します。

1	AA	AC	AG	AT	CA	CC	CG	CT	GA	GC	GG	GT	TA	TC	TG	TT
2	contig_1	0	0.0434782608695652	0.0434782608695652	0	0.0869565217391304	0.0869565217391304	0.0869565217391304	0.0869565217391304	0.0869565217391304	0.0869565217391304	0.0869565217391304	0.0869565217391304	0.0869565217391304	0.0869565217391304	0.0869565217391304

```
> param_bsgenome <- "BSgenome.Hsapiens.NCBI.GRCh38"#パッケージ名を指定(BSgenome系のゲノムパッケージ)
>
> #必要なパッケージをロード
> library(Biostrings) #パッケージの読み込み
> library(param_bsgenome, character.only=T)#指定したパッケージの読み込み
library(param_bsgenome, character.only = T) でエラー:
'BSgenome.Hsapiens.NCBI.GRCh38' という名前のパッケージはありません
>
> #前処理(指定したパッケージ中のオブジェクト名をgenomeに統一)
> tmp <- ls(paste("package", param_bsgenome, sep="."))#指定したパ
```

Name	Size
..	
hoge1.txt	221 B
hoge2.txt	945 B
hoge4.fa	299 B
seq_20.fasta	29 B



# 例題7を実行8

うまく実行できているヒトは、①のあたりで数分程度かかるとおもいます。ここがヒトゲノムパッケージからヒトゲノムのfastaオブジェクトを作成しているところなので、それなりに時間がかかります。

The screenshot shows the RStudio interface. The top editor pane displays a text file with two rows of genomic data. The first row contains 12 nucleotide pairs (AA, AC, AG, AT, CA, CC, CG, CT, GA, GC, GG, GT, TA, TC) and the second row contains contig identifiers and their corresponding GC content values.

The console pane shows the following R commands and their output:

```

>
要求されたパッケージ BSgenome 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 GenomicRanges をロード中です
要求されたパッケージ rtracklayer をロード中です
>
> #前処理(指定したパッケージ中のオブジェクト名をgenomeに統一)
> tmp <- ls(paste("package", param_bsgenome, sep=":"))#指定したパ
パッケージで利用可能なオブジェクト名を取得した結果をtmpに格納
> genome <- eval(parse(text=tmp)) #文字列tmpをRオブジェクト
としてgenomeに格納(パッケージ中のオブジェクトが一つしかないという前提です)
> fasta <- getSeq(genome) #ゲノム塩基配列情報を抽出し
た結果をfastaに格納
    
```

A red arrow with the number 1 points to the `getSeq` command in the console.

The bottom right pane shows a file explorer view of the project directory, listing files: `hoge1.txt` (221 B), `hoge2.txt` (945 B), `hoge4.fa` (299 B), `hoge7.txt` (945 B), and `seq_20.fasta` (29 B).



# 例題7を実行9

うまく実行できているヒトは、①のあたりで数分程度かかるとおもいます。ここがヒトゲノムパッケージからヒトゲノムのfastaオブジェクトを作成しているところなので、それなりに時間がかかります。うまくいった場合の実行結果の画面。②455という数値が見えていれば正解です。

The screenshot shows the RStudio interface. The top pane displays a text file with sequence data:

```

1 | AA AC AG AT CA CC CG CT GA GC GG GT TA TC
   | TG TT
2 | contig_1 0 0.0434782608695652 0.0434782608695652 0
   | .0869565217391304 0.0869565217391304 0.0869565217391304

```

The bottom-left pane shows the R console with the following code and output:

```

> #本②
> out <- dinucleotideFrequency(fasta, as.prob=T) #連続塩基の出現確
率情報をoutに格納
>
> #ファイルに保存
> tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.name
s=F) #tmpの中身を指定したファイル名で保存
>
>

```

The bottom-right pane shows a file explorer view of the 'hoge' directory:

Name	Size
..	
hoge1.txt	221 B
hoge2.txt	945 B
hoge4.fa	299 B
hoge7.txt	142.1 KB
seq_20.fasta	29 B

# 例題7を実行10

例題7の出力ファイル(hoge7.txt)をExcelで開いて、数値以外の見栄えを加工したもの。

	AA	AC	AG	AT	CA	CC	CG	CT	GA	GC	GG	GT	TA	TC	TG	TT
1	9.5%	5.0%	7.1%	7.4%	7.3%	5.4%	1.0%	7.1%	6.0%	4.4%	5.4%	5.0%	6.3%	6.0%	7.3%	9.6%
2	10.0%	5.0%	7.0%	7.9%	7.2%	5.0%	0.9%	7.0%	5.9%	4.1%	5.0%	5.0%	6.7%	5.9%	7.2%	10.0%
3	10.1%	5.0%	6.9%	8.0%	7.2%	4.9%	0.8%	6.9%	5.9%	4.0%	4.9%	5.0%	6.9%	5.9%	7.2%	10.2%
4	10.6%	5.0%	6.7%	8.5%	7.1%	4.5%	0.8%	6.7%	5.9%	3.8%	4.5%	5.0%	7.3%	5.8%	7.1%	10.6%
5	10.2%	5.0%	6.9%	8.1%	7.2%	4.8%	0.9%	6.9%	5.9%	4.0%	4.8%	5.1%	6.9%	5.9%	7.2%	10.3%
6	10.2%	5.0%	6.9%	8.1%	7.2%	4.8%	0.9%	6.9%	5.9%	4.0%	4.9%	5.0%	6.9%	5.9%	7.2%	10.2%
7	9.8%	5.0%	7.0%	7.7%	7.3%	5.1%	1.0%	7.0%	6.0%	4.2%	5.1%	5.1%	6.5%	5.9%	7.3%	10.0%
8	10.0%	5.1%	6.9%	7.9%	7.2%	5.0%	0.9%	6.9%	6.0%	4.1%	5.0%	5.0%	6.7%	5.9%	7.2%	10.0%
9	9.7%	5.1%	7.0%	7.6%	7.3%	5.3%	1.0%	7.0%	6.0%	4.3%	5.3%	5.0%	6.4%	6.0%	7.3%	9.7%
10	9.6%	5.0%	7.1%	7.5%	7.3%	5.3%	1.0%	7.1%	6.0%	4.4%	5.3%	5.1%	6.3%	6.0%	7.4%	9.7%
11	9.5%	5.1%	7.1%	7.5%	7.3%	5.3%	1.0%	7.1%	6.1%	4.3%	5.4%	5.0%	6.3%	6.0%	7.3%	9.6%
12	9.8%	5.0%	7.0%	7.7%	7.2%	5.1%	1.0%	7.0%	6.0%	4.2%	5.2%	5.1%	6.6%	6.0%	7.2%	9.9%
13	10.5%	5.0%	6.8%	8.4%	7.1%	4.5%	0.9%	6.7%	5.9%	3.8%	4.6%	5.0%	7.2%	5.8%	7.1%	10.6%
14	9.7%	5.0%	7.0%	7.7%	7.2%	5.1%	1.0%	7.0%	6.0%	4.2%	5.2%	5.1%	6.6%	5.9%	7.3%	9.9%
15	9.4%	5.1%	7.1%	7.3%	7.3%	5.4%	1.1%	7.1%	6.0%	4.5%	5.5%	5.1%	6.1%	6.0%	7.4%	9.5%
16	8.6%	5.1%	7.3%	6.7%	7.5%	6.1%	1.4%	7.2%	6.1%	5.0%	6.1%	5.1%	5.4%	6.1%	7.6%	8.8%
17	8.5%	5.1%	7.3%	6.4%	7.4%	6.3%	1.5%	7.4%	6.2%	5.1%	6.4%	5.0%	5.2%	6.1%	7.5%	8.6%
18	10.1%	5.1%	7.0%	7.9%	7.2%	4.7%	0.9%	6.9%	6.1%	4.0%	4.9%	5.1%	6.7%	5.9%	7.3%	10.3%
19	7.7%	5.1%	7.5%	5.7%	7.5%	7.0%	1.9%	7.4%	6.2%	5.6%	7.1%	5.2%	4.5%	6.2%	7.6%	7.9%
20	8.8%	5.0%	7.3%	6.8%	7.5%	5.8%	1.2%	7.3%	6.2%	4.8%	6.0%	5.1%	5.5%	6.1%	7.6%	9.1%
21	9.8%	5.1%	6.9%	7.7%	7.3%	5.1%	1.2%	6.9%	6.0%	4.3%	5.1%	5.1%	6.4%	6.0%	7.3%	9.9%

# 例題7を実行11



例題7の出力ファイル(hoge7.txt)をExcelで開いて、数値以外の見栄えを加工したもの。①ヒトゲノムのドラフト配列決定論文でも言及されているが、確かに②ヒトゲノム中のCpG出現確率は低いですね。

	AA	AC	AG	AT	CA	CC	CG	CT	GA	GC	GG	GT	TA	TC	TG	TT
1	9.5%	5.0%	7.1%	7.4%	7.3%	5.4%	1.0%	7.1%	6.0%	4.4%	5.4%	5.0%	6.3%	6.0%	7.3%	9.6%
2	10.0%	5.0%	7.0%	7.9%	7.2%	5.0%	0.9%	7.0%	5.9%	4.1%	5.0%	5.0%	6.7%	5.9%	7.2%	10.0%
3	10.1%	5.0%	6.9%	8.0%	7.2%	4.9%	0.8%	6.9%	5.9%	4.0%	4.9%	5.0%	6.9%	5.9%	7.2%	10.2%
4	10.6%	5.0%	6.7%	8.5%	7.1%	4.5%	0.8%	6.7%	5.9%	3.8%	4.5%	5.0%	7.3%	5.8%	7.1%	10.6%
5	10.2%	5.0%	6.9%	8.1%	7.2%	4.8%	0.9%	6.9%	5.9%	4.0%	4.8%	5.1%	6.9%	5.9%	7.2%	10.3%
6	10.2%	5.0%	6.9%	8.1%	7.2%	4.8%	0.9%	6.9%	5.9%	4.0%	4.9%	5.0%	6.9%	5.9%	7.2%	10.2%
7	9.8%	5.0%	7.0%	7.7%	7.3%	5.1%	1.0%	7.0%	6.0%	4.2%	5.1%	5.1%	6.5%	5.9%	7.3%	10.0%
8	10.0%	5.1%	6.9%	7.9%	7.2%	5.0%	0.9%	6.9%	6.0%	4.1%	5.0%	5.0%	6.7%	5.9%	7.2%	10.0%
9	9.7%	5.1%	7.0%	7.6%	7.3%	5.3%	1.0%	7.0%	6.0%	4.3%	5.3%	5.0%	6.4%	6.0%	7.3%	9.7%
10	9.6%	5.0%	7.1%	7.5%	7.3%	5.3%	1.0%	7.1%	6.0%	4.4%	5.3%	5.1%	6.3%	6.0%	7.4%	9.7%
11	9.5%	5.1%	7.1%	7.5%	7.3%	5.3%	1.0%	7.1%	6.1%	4.3%	5.4%	5.0%	6.3%	6.0%	7.3%	9.6%
12	9.8%	5.0%	7.0%	7.7%	7.2%	5.1%	1.0%	7.0%	6.0%	4.2%	5.2%	5.1%	6.6%	6.0%	7.2%	9.9%
13	10.5%	5.0%	6.8%	8.4%	7.1%	4.5%	0.9%	6.7%	5.9%	3.8%	4.6%	5.0%	7.2%	5.8%	7.1%	10.6%
14	9.7%	5.0%	7.0%	7.7%	7.2%	5.1%	1.0%	7.0%	6.0%	4.2%	5.2%	5.1%	6.6%	5.9%	7.3%	9.9%
15	9.4%	5.1%	7.1%	7.3%	7.3%	5.4%	1.1%	7.1%	6.0%	4.5%	5.5%	5.1%	6.1%	6.0%	7.4%	9.5%
16	8.6%	5.1%	7.3%	6.7%	7.5%	6.1%	1.4%	7.2%	6.1%	5.0%	6.1%	5.1%	5.4%	6.1%	7.6%	8.8%
17	8.5%	5.1%	7.3%	6.4%	7.4%	6.3%	1.5%	7.4%	6.2%	5.1%	6.4%	5.0%	5.2%	6.1%	7.5%	8.6%
18	10.1%	5.1%	7.0%	7.9%	7.2%	4.7%	0.9%	6.9%	6.1%	4.0%	4.9%	5.1%	6.7%	5.9%	7.3%	10.3%
19	7.7%	5.1%	7.5%	5.7%	7.5%	7.0%	1.9%	7.4%	6.2%	5.6%	7.1%	5.2%	4.5%	6.2%	7.6%	7.9%
20	8.8%	5.0%	7.3%	6.8%	7.5%	5.8%	1.2%	7.3%	6.2%	4.8%	6.0%	5.1%	5.5%	6.1%	7.6%	9.1%
21	9.8%	5.1%	6.9%	7.7%	7.3%	5.1%	1.2%	6.9%	6.0%	4.3%	5.1%	5.1%	6.4%	6.0%	7.3%	9.9%



# Contents

- Introduction、出現頻度解析( $k=2$ )、出現頻度解析( $k=1$ )
- $k=1$ で実践、multi-FASTAファイル、他の例題を実行
- $k=2$ で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8( $k=2$ )、例題9( $k=3$ )、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11( $k=10$ )、例題12( $k=10$ )、シークエンスエラーを含む場合

# 確率の話1



仮想ゲノムのGC含量を50%とすると...

## (1) 1連続塩基の出現確率

A or Tの場合: 0.250

C or Gの場合: 0.250



## (2) 2連続塩基の出現確率

AとTのみから構成される場合(AA, AT, TA, TT):  $0.250 \times 0.250 = 6.25\%$

CとGのみから構成される場合(CC, CG, GC, GG):  $0.250 \times 0.250 = 6.25\%$

それ以外(AC, AG, CA, CT, GA, GT, TC, TG):  $0.250 \times 0.250 = 6.25\%$



確率をおさらい。ゲノム中にA,C,G,Tの4種類しかなく、①それぞれの塩基の出現確率が等しい場合(=  $1/4 = 0.25$ )、②任意の2連続塩基の出現確率は $(0.25)^2$ で表される。つまり、AAの出現確率もCGの出現確率も全て $1/16 = 0.0625$ だということ。③この条件はゲノム中のGC含量(GC contents)が50%だということと等価。

# 確率の話2



ヒトゲノムのGC含量は約41%なので...

## (1) 1連続塩基の出現確率

A or Tの場合: 0.295

C or Gの場合: 0.205



## (2) 2連続塩基の出現確率

AとTのみから構成される場合(AA, AT, TA, TT):  $0.295 \times 0.295 = 8.70\%$

CとGのみから構成される場合(CC, CG, GC, GG):  $0.205 \times 0.205 = 4.20\%$

それ以外(AC, AG, CA, CT, GA, GT, TC, TG):  $0.205 \times 0.295 = 6.05\%$



現実のゲノムはぴったりGC含量 = 50%というわけではない。④例えばヒトゲノムの場合は約41%なので、⑤C or Gの出現確率は $0.41/2 = 0.205$ となる。必然的に残りのA or Tの出現確率は $(1.00 - 0.41)/2 = 0.295$ となる。したがって、2連続塩基の出現確率(期待値)も、⑥構成塩基によって異なる。GC含量が50%よりも低いと、2連続塩基の出現確率は「CとGのみから構成される場合」に最も低くなる。

先ほどのヒトゲノムの染色体ごとの2連続塩基の出現確率(観測値)を、CとGを赤字にして再掲。

# 確率の話3

	AA	AC	AG	AT	CA	CC	CG	CT	GA	GC	GG	GT	TA	TC	TG	TT
1	9.5%	5.0%	7.1%	7.4%	7.3%	5.4%	1.0%	7.1%	6.0%	4.4%	5.4%	5.0%	6.3%	6.0%	7.3%	9.6%
2	10.0%	5.0%	7.0%	7.9%	7.2%	5.0%	0.9%	7.0%	5.9%	4.1%	5.0%	5.0%	6.7%	5.9%	7.2%	10.0%
3	10.1%	5.0%	6.9%	8.0%	7.2%	4.9%	0.8%	6.9%	5.9%	4.0%	4.9%	5.0%	6.9%	5.9%	7.2%	10.2%
4	10.6%	5.0%	6.7%	8.5%	7.1%	4.5%	0.8%	6.7%	5.9%	3.8%	4.5%	5.0%	7.3%	5.8%	7.1%	10.6%
5	10.2%	5.0%	6.9%	8.1%	7.2%	4.8%	0.9%	6.9%	5.9%	4.0%	4.8%	5.1%	6.9%	5.9%	7.2%	10.3%
6	10.2%	5.0%	6.9%	8.1%	7.2%	4.8%	0.9%	6.9%	5.9%	4.0%	4.9%	5.0%	6.9%	5.9%	7.2%	10.2%
7	9.8%	5.0%	7.0%	7.7%	7.3%	5.1%	1.0%	7.0%	6.0%	4.2%	5.1%	5.1%	6.5%	5.9%	7.3%	10.0%
8	10.0%	5.1%	6.9%	7.9%	7.2%	5.0%	0.9%	6.9%	6.0%	4.1%	5.0%	5.0%	6.7%	5.9%	7.2%	10.0%
9	9.7%	5.1%	7.0%	7.6%	7.3%	5.3%	1.0%	7.0%	6.0%	4.3%	5.3%	5.0%	6.4%	6.0%	7.3%	9.7%
10	9.6%	5.0%	7.1%	7.5%	7.3%	5.3%	1.0%	7.1%	6.0%	4.4%	5.3%	5.1%	6.3%	6.0%	7.4%	9.7%
11	9.5%	5.1%	7.1%	7.5%	7.3%	5.3%	1.0%	7.1%	6.1%	4.3%	5.4%	5.0%	6.3%	6.0%	7.3%	9.6%
12	9.8%	5.0%	7.0%	7.7%	7.2%	5.1%	1.0%	7.0%	6.0%	4.2%	5.2%	5.1%	6.6%	6.0%	7.2%	9.9%
13	10.5%	5.0%	6.8%	8.4%	7.1%	4.5%	0.9%	6.7%	5.9%	3.8%	4.6%	5.0%	7.2%	5.8%	7.1%	10.6%
14	9.7%	5.0%	7.0%	7.7%	7.2%	5.1%	1.0%	7.0%	6.0%	4.2%	5.2%	5.1%	6.6%	5.9%	7.3%	9.9%
15	9.4%	5.1%	7.1%	7.3%	7.3%	5.4%	1.1%	7.1%	6.0%	4.5%	5.5%	5.1%	6.1%	6.0%	7.4%	9.5%
16	8.6%	5.1%	7.3%	6.7%	7.5%	6.1%	1.4%	7.2%	6.1%	5.0%	6.1%	5.1%	5.4%	6.1%	7.6%	8.8%
17	8.5%	5.1%	7.3%	6.4%	7.4%	6.3%	1.5%	7.4%	6.2%	5.1%	6.4%	5.0%	5.2%	6.1%	7.5%	8.6%
18	10.1%	5.1%	7.0%	7.9%	7.2%	4.7%	0.9%	6.9%	6.1%	4.0%	4.9%	5.1%	6.7%	5.9%	7.3%	10.3%
19	7.7%	5.1%	7.5%	5.7%	7.5%	7.0%	1.9%	7.4%	6.2%	5.6%	7.1%	5.2%	4.5%	6.2%	7.6%	7.9%
20	8.8%	5.0%	7.3%	6.8%	7.5%	5.8%	1.2%	7.3%	6.2%	4.8%	6.0%	5.1%	5.5%	6.1%	7.6%	9.1%
21	9.8%	5.1%	6.9%	7.7%	7.3%	5.1%	1.2%	6.9%	6.0%	4.3%	5.1%	5.1%	6.4%	6.0%	7.3%	9.9%



# 確率の話4

先ほどのヒトゲノムの染色体ごとの2連続塩基の出現確率(観測値)を、CとGを赤字にして再掲。さらに構成塩基別にソート後。

	AA	AT	TA	TT	AC	AG	CA	CT	GA	GT	TC	TG	CC	CG	GC	GG
1	9.5%	7.4%	6.3%	9.6%	5.0%	7.1%	7.3%	7.1%	6.0%	5.0%	6.0%	7.3%	5.4%	1.0%	4.4%	5.4%
2	10.0%	7.9%	6.7%	10.0%	5.0%	7.0%	7.2%	7.0%	5.9%	5.0%	5.9%	7.2%	5.0%	0.9%	4.1%	5.0%
3	10.1%	8.0%	6.9%	10.2%	5.0%	6.9%	7.2%	6.9%	5.9%	5.0%	5.9%	7.2%	4.9%	0.8%	4.0%	4.9%
4	10.6%	8.5%	7.3%	10.6%	5.0%	6.7%	7.1%	6.7%	5.9%	5.0%	5.8%	7.1%	4.5%	0.8%	3.8%	4.5%
5	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	0.9%	4.0%	4.8%
6	10.2%	8.1%	6.9%	10.2%	5.0%	6.9%	7.2%	6.9%	5.9%	5.0%	5.9%	7.2%	4.8%	0.9%	4.0%	4.9%
7	9.8%	7.7%	6.5%	10.0%	5.0%	7.0%	7.3%	7.0%	6.0%	5.1%	5.9%	7.3%	5.1%	1.0%	4.2%	5.1%
8	10.0%	7.9%	6.7%	10.0%	5.1%	6.9%	7.2%	6.9%	6.0%	5.0%	5.9%	7.2%	5.0%	0.9%	4.1%	5.0%
9	9.7%	7.6%	6.4%	9.7%	5.1%	7.0%	7.3%	7.0%	6.0%	5.0%	6.0%	7.3%	5.3%	1.0%	4.3%	5.3%
10	9.6%	7.5%	6.3%	9.7%	5.0%	7.1%	7.3%	7.1%	6.0%	5.1%	6.0%	7.4%	5.3%	1.0%	4.4%	5.3%
11	9.5%	7.5%	6.3%	9.6%	5.1%	7.1%	7.3%	7.1%	6.1%	5.0%	6.0%	7.3%	5.3%	1.0%	4.3%	5.4%
12	9.8%	7.7%	6.6%	9.9%	5.0%	7.0%	7.2%	7.0%	6.0%	5.1%	6.0%	7.2%	5.1%	1.0%	4.2%	5.2%
13	10.5%	8.4%	7.2%	10.6%	5.0%	6.8%	7.1%	6.7%	5.9%	5.0%	5.8%	7.1%	4.5%	0.9%	3.8%	4.6%
14	9.7%	7.7%	6.6%	9.9%	5.0%	7.0%	7.2%	7.0%	6.0%	5.1%	5.9%	7.3%	5.1%	1.0%	4.2%	5.2%
15	9.4%	7.3%	6.1%	9.5%	5.1%	7.1%	7.3%	7.1%	6.0%	5.1%	6.0%	7.4%	5.4%	1.1%	4.5%	5.5%
16	8.6%	6.7%	5.4%	8.8%	5.1%	7.3%	7.5%	7.2%	6.1%	5.1%	6.1%	7.6%	6.1%	1.4%	5.0%	6.1%
17	8.5%	6.4%	5.2%	8.6%	5.1%	7.3%	7.4%	7.4%	6.2%	5.0%	6.1%	7.5%	6.3%	1.5%	5.1%	6.4%
18	10.1%	7.9%	6.7%	10.3%	5.1%	7.0%	7.2%	6.9%	6.1%	5.1%	5.9%	7.3%	4.7%	0.9%	4.0%	4.9%
19	7.7%	5.7%	4.5%	7.9%	5.1%	7.5%	7.5%	7.4%	6.2%	5.2%	6.2%	7.6%	7.0%	1.9%	5.6%	7.1%
20	8.8%	6.8%	5.5%	9.1%	5.0%	7.3%	7.5%	7.3%	6.2%	5.1%	6.1%	7.6%	5.8%	1.2%	4.8%	6.0%
21	9.8%	7.7%	6.4%	9.9%	5.1%	7.0%	7.3%	7.1%	6.0%	5.1%	6.0%	7.3%	5.3%	1.0%	4.3%	5.3%



# 確率の話5

- ①AとTのみから構成される場合は8.70%、
- ②CとGのみから構成される場合は4.20%、
- ③それ以外は6.05%、というのが期待値。

①

③

②

	AA	AT	TA	TT	AC	AG	CA	CT	GA	GT	TC	TG	CC	CG	GC	GG
1	9.5%	7.4%	6.3%	9.6%	5.0%	7.1%	7.3%	7.1%	6.0%	5.0%	6.0%	7.3%	5.4%	1.0%	4.4%	5.4%
2	10.0%	7.9%	6.7%	10.0%	5.0%	7.0%	7.2%	7.0%	5.9%	5.0%	5.9%	7.2%	5.0%	0.9%	4.1%	5.0%
3	10.1%	8.0%	6.9%	10.2%	5.0%	6.9%	7.2%	6.9%	5.9%	5.0%	5.9%	7.2%	4.9%	0.8%	4.0%	4.9%
4	10.6%	8.5%	7.3%	10.6%	5.0%	6.7%	7.1%	6.7%	5.9%	5.0%	5.8%	7.1%	4.5%	0.8%	3.8%	4.5%
5	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	0.9%	4.0%	4.8%
6	10.2%	8.1%	6.9%	10.2%	5.0%	6.9%	7.2%	6.9%	5.9%	5.0%	5.9%	7.2%	4.8%	0.9%	4.0%	4.9%
7	9.8%	7.7%	6.5%	10.0%	5.0%	7.0%	7.3%	7.0%	6.0%	5.1%	5.9%	7.3%	5.1%	1.0%	4.2%	5.1%
8	10.0%	7.9%	6.7%	10.0%	5.1%	6.9%	7.2%	6.9%	6.0%	5.0%	5.9%	7.2%	5.0%	0.9%	4.1%	5.0%
9	9.7%	7.6%	6.4%	9.7%	5.1%	7.0%	7.3%	7.0%	6.0%	5.0%	6.0%	7.3%	5.3%	1.0%	4.3%	5.3%
10	9.6%	7.5%	6.3%	9.7%	5.0%	7.1%	7.3%	7.1%	6.0%	5.1%	6.0%	7.4%	5.3%	1.0%	4.4%	5.3%
11	9.5%	7.5%	6.3%	9.6%	5.1%	7.1%	7.3%	7.1%	6.1%	5.0%	6.0%	7.3%	5.3%	1.0%	4.3%	5.4%
12	9.8%	7.7%	6.6%	9.9%	5.0%	7.0%	7.2%	7.0%	6.0%	5.1%	6.0%	7.2%	5.1%	1.0%	4.2%	5.2%
13	10.5%	8.4%	7.2%	10.6%	5.0%	6.8%	7.1%	6.7%	5.9%	5.0%	5.8%	7.1%	4.5%	0.9%	3.8%	4.6%
14	9.7%	7.7%	6.6%	9.9%	5.0%	7.0%	7.2%	7.0%	6.0%	5.1%	5.9%	7.3%	5.1%	1.0%	4.2%	5.2%
15	9.4%	7.3%	6.1%	9.5%	5.1%	7.1%	7.3%	7.1%	6.0%	5.1%	6.0%	7.4%	5.4%	1.1%	4.5%	5.5%
16	8.6%	6.7%	5.4%	8.8%	5.1%	7.3%	7.5%	7.2%	6.1%	5.1%	6.1%	7.6%	6.1%	1.4%	5.0%	6.1%
17	8.5%	6.4%	5.2%	8.6%	5.1%	7.3%	7.4%	7.4%	6.2%	5.0%	6.1%	7.5%	6.3%	1.5%	5.1%	6.4%
18	10.1%	7.9%	6.7%	10.3%	5.1%	7.0%	7.2%	6.9%	6.1%	5.1%	5.9%	7.3%	4.7%	0.9%	4.0%	4.9%
19	7.7%	5.7%	4.5%	7.9%	5.1%	7.5%	7.5%	7.4%	6.2%	5.2%	6.2%	7.6%	7.0%	1.9%	5.6%	7.1%
20	8.8%	6.8%	5.5%	9.1%	5.0%	7.3%	7.5%	7.3%	6.2%	5.1%	6.1%	7.6%	5.8%	1.2%	4.8%	6.0%
21	9.8%	7.7%	6.4%	9.9%	5.1%	6.9%	7.3%	6.9%	6.0%	5.1%	6.0%	7.3%	5.1%	1.0%	4.2%	5.1%

④CGの観測値(約1%)は、その期待値(4.20%)より非常に低い!

# 確率の話6



	AA	AT	TA	TT	AC	AG	CA	CT	GA	GT	TC	TG	CC	CG	GC	GG
1	9.5%	7.4%	6.3%	9.6%	5.0%	7.1%	7.3%	7.1%	6.0%	5.0%	6.0%	7.3%	5.4%	1.0%	4.4%	5.4%
2	10.0%	7.9%	6.7%	10.0%	5.0%	7.0%	7.2%	7.0%	5.9%	5.0%	5.9%	7.2%	5.0%	0.9%	4.1%	5.0%
3	10.1%	8.0%	6.9%	10.2%	5.0%	6.9%	7.2%	6.9%	5.9%	5.0%	5.9%	7.2%	4.9%	0.8%	4.0%	4.9%
4	10.6%	8.5%	7.3%	10.6%	5.0%	6.7%	7.1%	6.7%	5.9%	5.0%	5.8%	7.1%	4.5%	0.8%	3.8%	4.5%
5	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	0.9%	4.0%	4.8%
6	10.2%	8.1%	6.9%	10.2%	5.0%	6.9%	7.2%	6.9%	5.9%	5.0%	5.9%	7.2%	4.8%	0.9%	4.0%	4.9%
7	9.8%	7.7%	6.5%	10.0%	5.0%	7.0%	7.3%	7.0%	6.0%	5.1%	5.9%	7.3%	5.1%	1.0%	4.2%	5.1%
8	10.0%	7.9%	6.7%	10.0%	5.1%	6.9%	7.2%	6.9%	6.0%	5.0%	5.9%	7.2%	5.0%	0.9%	4.1%	5.0%
9	9.7%	7.6%	6.4%	9.7%	5.1%	7.0%	7.3%	7.0%	6.0%	5.0%	6.0%	7.3%	5.3%	1.0%	4.3%	5.3%
10	9.6%	7.5%	6.3%	9.7%	5.0%	7.1%	7.3%	7.1%	6.0%	5.1%	6.0%	7.4%	5.3%	1.0%	4.4%	5.3%
11	9.5%	7.5%	6.3%	9.6%	5.1%	7.1%	7.3%	7.1%	6.1%	5.0%	6.0%	7.3%	5.3%	1.0%	4.3%	5.4%
12	9.8%	7.7%	6.6%	9.9%	5.0%	7.0%	7.2%	7.0%	6.0%	5.1%	6.0%	7.2%	5.1%	1.0%	4.2%	5.2%
13	10.5%	8.4%	7.2%	10.6%	5.0%	6.8%	7.1%	6.7%	5.9%	5.0%	5.8%	7.1%	4.5%	0.9%	3.8%	4.6%
14	9.7%	7.7%	6.6%	9.9%	5.0%	7.0%	7.2%	7.0%	6.0%	5.1%	5.9%	7.3%	5.1%	1.0%	4.2%	5.2%
15	9.4%	7.3%	6.1%	9.5%	5.1%	7.1%	7.3%	7.1%	6.0%	5.1%	6.0%	7.4%	5.4%	1.1%	4.5%	5.5%
16	8.6%	6.7%	5.4%	8.8%	5.1%	7.3%	7.5%	7.2%	6.1%	5.1%	6.1%	7.6%	6.1%	1.4%	5.0%	6.1%
17	8.5%	6.4%	5.2%	8.6%	5.1%	7.3%	7.4%	7.4%	6.2%	5.0%	6.1%	7.5%	6.3%	1.5%	5.1%	6.4%
18	10.1%	7.9%	6.7%	10.3%	5.1%	7.0%	7.2%	6.9%	6.1%	5.1%	5.9%	7.3%	4.7%	0.9%	4.0%	4.9%
19	7.7%	5.7%	4.5%	7.9%	5.1%	7.5%	7.5%	7.4%	6.2%	5.2%	6.2%	7.6%	7.0%	1.9%	5.6%	7.1%
20	8.8%	6.8%	5.5%	9.1%	5.0%	7.3%	7.5%	7.3%	6.2%	5.1%	6.1%	7.6%	5.8%	1.2%	4.8%	6.0%
21	9.8%	7.7%	6.4%	9.9%	5.1%	6.9%	7.3%	6.9%	6.0%	5.1%	6.0%	7.3%	5.1%	1.2%	4.3%	5.1%

# 確率の話7

④CGの観測値(約1%)は、その期待値(4.20%)より非常に低い！⑤原著論文中的、⑥の項目中の赤下線部分を実際に確認したことに相当します。

④

	AA	AT	TA	TT	AC	AG	CA	CT	GA	GT	TC	TG	CC	CG	GC	GG
1	9.5%	7.4%	6.3%	9.6%	5.0%	7.1%	7.3%	7.1%	6.0%	5.0%	6.0%	7.3%	5.4%	1.0%	4.4%	5.4%
2	10.0%	7.9%	6.7%	10.0%	5.0%	7.0%	7.2%	7.0%	5.9%	5.0%	5.9%	7.2%	5.0%	0.9%	4.1%	5.0%
3	10.1%	8.0%	6.9%	10.2%	5.0%	6.9%	7.2%	6.9%	5.9%	5.0%	5.9%	7.2%	4.9%	0.8%	4.0%	4.9%
4	10.6%	8.5%	7.3%	10.6%	5.0%	6.7%	7.1%	6.7%	5.9%	5.0%	5.8%	7.1%	4.5%	0.8%	3.8%	4.5%
5	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	0.9%	4.0%	4.8%
6	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	0.9%	4.0%	4.8%
7	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	0.9%	4.0%	4.8%
8	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	0.9%	4.0%	4.8%
9	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	0.9%	4.0%	4.8%
10	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	0.9%	4.0%	4.8%
11	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	0.9%	4.0%	4.8%
12	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	0.9%	4.0%	4.8%
13	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	0.9%	4.0%	4.8%
14	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	0.9%	4.0%	4.8%
15	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	0.9%	4.0%	4.8%
16	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	0.9%	4.0%	4.8%
17	8.5%	6.4%	5.2%	8.6%	5.1%	7.3%	7.4%	7.4%	6.2%	5.0%	6.1%	7.5%	6.3%	1.5%	5.1%	6.4%
18	10.1%	7.9%	6.7%	10.3%	5.1%	7.0%	7.2%	6.9%	6.1%	5.1%	5.9%	7.3%	4.7%	0.9%	4.0%	4.9%
19	7.7%	5.7%	4.5%	7.9%	5.1%	7.5%	7.5%	7.4%	6.2%	5.2%	6.2%	7.6%	7.0%	1.9%	5.6%	7.1%
20	8.8%	6.8%	5.5%	9.1%	5.0%	7.3%	7.5%	7.3%	6.2%	5.1%	6.1%	7.6%	5.8%	1.2%	4.8%	6.0%
21	9.8%	7.7%	6.4%	9.9%	5.1%	6.9%	7.3%	6.9%	6.0%	5.1%	6.0%	7.3%	5.1%	1.2%	4.3%	5.1%

## CpG islands

⑥

A related topic is the distribution of so-called CpG islands across the genome. The dinucleotide CpG is notable because it is greatly under-represented in human DNA, occurring at only about one-fifth of the roughly 4% frequency that would be expected by simply multiplying the typical fraction of Cs and Gs (0.21 × 0.21). The deficit occurs because most

⑤

# 確率の話8

④CGの観測値(約1%)は、その期待値(4.20%)より非常に低い！  
 ⑤原著論文中の、⑥の項目中の赤下線部分を実際に確認したことに相当します。ゲノム中でCとGの割合である⑦0.21を掛け合わせた結果から期待される、⑧約4%という値が

...

	AA	AT	TA	TT	AC	AG	CA	CT	GA	GT	TC	TT	TA	TA	TA	TA	TA
1	9.5%	7.4%	6.3%	9.6%	5.0%	7.1%	7.3%	7.1%	6.0%	5.0%	6.0%	7.2%	4.9%	0.8%	4.0%	4.9%	4.9%
2	10.0%	7.9%	6.7%	10.0%	5.0%	7.0%	7.2%	7.0%	5.9%	5.0%	5.9%	7.2%	4.9%	0.8%	3.8%	4.5%	4.5%
3	10.1%	8.0%	6.9%	10.2%	5.0%	6.9%	7.2%	6.9%	5.9%	5.0%	5.9%	7.2%	4.9%	0.9%	4.0%	4.8%	4.8%
4	10.6%	8.5%	7.3%	10.6%	5.0%	6.7%	7.1%	6.7%	5.9%	5.0%	5.8%	7.1%	4.5%	0.9%	4.0%	4.8%	4.8%
5	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	0.9%	4.0%	4.9%	4.9%
6	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	1.0%	4.2%	5.1%	5.1%
7	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	0.9%	4.1%	5.0%	5.0%
8	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	1.0%	4.3%	5.3%	5.3%
9	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	1.0%	4.4%	5.3%	5.3%
10	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	1.0%	4.3%	5.4%	5.4%
11	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	1.0%	4.2%	5.2%	5.2%
12	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	0.9%	3.8%	4.6%	4.6%
13	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	1.0%	4.2%	5.2%	5.2%
14	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	1.1%	4.5%	5.5%	5.5%
15	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	1.4%	5.0%	6.1%	6.1%
16	10.2%	8.1%	6.9%	10.3%	5.0%	6.9%	7.2%	6.9%	5.9%	5.1%	5.9%	7.2%	4.8%	1.5%	5.1%	6.4%	6.4%
17	8.5%	6.4%	5.2%	8.6%	5.1%	7.3%	7.4%	7.4%	6.2%	5.0%	6.1%	7.5%	6.3%	1.5%	5.1%	6.4%	6.4%
18	10.1%	7.9%	6.7%	10.3%	5.1%	7.0%	7.2%	6.9%	6.1%	5.1%	5.9%	7.3%	4.7%	0.9%	4.0%	4.9%	4.9%
19	7.7%	5.7%	4.5%	7.9%	5.1%	7.5%	7.5%	7.4%	6.2%	5.2%	6.2%	7.6%	7.0%	1.9%	5.6%	7.1%	7.1%
20	8.8%	6.8%	5.5%	9.1%	5.0%	7.3%	7.5%	7.3%	6.2%	5.1%	6.1%	7.6%	5.8%	1.2%	4.8%	6.0%	6.0%
21	9.8%	7.7%	6.4%	9.9%	5.1%	6.9%	7.3%	6.9%	6.0%	5.1%	6.0%	7.3%	5.1%	1.2%	4.3%	5.1%	5.1%

## CpG islands

A related topic is the distribution of so-called CpG islands across the genome. The dinucleotide CpG is notable because it is greatly underrepresented in human DNA, occurring at only about one-fifth of the roughly 4% frequency that would be expected by simply multiplying the typical fraction of Cs and Gs (0.21 × 0.21). The deficit occurs because most

⑧

⑦

# 確率の話9

ヒトゲノムのGC含量は約41%なので...

## (1) 1連続塩基の出現確率

A or Tの場合: 0.295

C or Gの場合: 0.205 

## (2) 2連続塩基の出現確率

AとTのみから構成される場合(AA, AT, TA, TT):  $0.295 \times 0.295 = 8.70\%$

CとGのみから構成される場合(CC, CG, GC, GG):  $0.205 \times 0.205 = 4.20\%$  

それ以外(AC, AG, CA, CT, GA, GT, TC, TG):  $0.205 \times 0.295 = 6.05\%$

④CGの観測値(約1%)は、その期待値(4.20%)より非常に低い! ⑤原著論文での、⑥の項目中の赤下線部分を実際に確認したことに相当します。ゲノム中でCとGの割合である⑦0.21を掛け合わせた結果から期待される、⑧約4%という値が、⑨0.205を掛け合わせた結果から期待される、⑩4.2%という数値に対応します。確率を考える場合はGC含量によって期待値が異なる点に注意すべし、ということ。

# Contents

- Introduction、出現頻度解析(k=2)、出現頻度解析(k=1)
- k=1で実践、multi-FASTAファイル、他の例題を実行
- k=2で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8(k=2)、例題9(k=3)、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11(k=10)、例題12(k=10)、シークエンスエラーを含む場合

# 作図(例題10)1

①例題10。②でも書かれていますが、さきほどの例題7と基本的に同じで、 $4^2 = 16$ 種類の2連続塩基の出現確率の箱ひげ図(box plot)を追加で出力しています。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#intro\_general\_kmer\_2\_bios...

## 10. ヒトゲノム配列パッケージ(BSgenome.Hsapiens.NCBI.GRCh38)の場合 :

7.と基本的に同じですが、box plotのPNGファイルも出力しています。

```

out_f1 <- "hoge10.txt"           #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge10.png"         #出力ファイル名を指定してout_f2に格納
param_bsgenome <- "BSgenome.Hsapiens.NCBI.GRCh38" #パッケージ名を指定(BSgenome系のゲノムパッケージ)
param_fig <- c(700, 400)       #ファイル出力時の横幅と縦幅を指定(単位はピクセル)

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

#前処理(指定したパッケージ中のオブジェクト名をgenomeに統一)
tmp <- ls(paste("package", param_bsgenome, sep=":")) #指定したパッケージで利用可能なオブジェクト名を
genome <- eval(parse(text=tmp)) #文字列tmpをRオブジェクトとしてgenomeに格納(パッケージ中にオ
fasta <- getSeq(genome)       #ゲノム塩基配列情報を抽出した結果をfastaに格納
names(fasta) <- seqnames(genome) #description情報を追加している
fasta                        #確認してるだけです

#本番
out <- dinucleotideFrequency(fasta, as.prob=T) #連続塩基の出現確率情報をoutに格納

```

[トップページ](#) ^

# 作図(例題10)2

①例題10。②でも書かれていますが、さきほどの例題7と基本的に同じで、 $4^2 = 16$ 種類の2連続塩基の出現確率の箱ひげ図(box plot)を追加で出力しています。③box plotはhoge10.pngに保存され、④その大きさは700×400と指定しています。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#int

## 10. ヒトゲノム配列パッケージ(BSgenome.Hsapiens.NCBI.GRCh38)の場合 :

7.と基本的に同じですが、box plotのPNGファイルも出力しています。

```

out_f1 <- "hoge10.txt" #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge10.png" #出力ファイル名を指定してout_f2に格納
param_bsgenome <- "BSgenome.Hsapiens.NCBI.GRCh38" #パッケージ名を指定(BSgenome系のゲノムパッケージ)
param_fig <- c(700, 400) #ファイル出力時の横幅と縦幅を指定(単位はピクセル)

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

#前処理(指定したパッケージ中のオブジェクト名をgenomeに統一)
tmp <- ls(paste("package", param_bsgenome, sep=":")) #指定したパッケージで利用可能なオブジェクト名を
genome <- eval(parse(text=tmp)) #文字列tmpをRオブジェクトとしてgenomeに格納(パッケージ中にオ
fasta <- getSeq(genome) #ゲノム塩基配列情報を抽出した結果をfastaに格納
names(fasta) <- seqnames(genome) #description情報を追加している
fasta #確認してるだけです

#本番
out <- dinucleotideFrequency(fasta, as.prob=T) #連続塩基の出現確率情報をoutに格納

```

[トップページ](#) ^



# 作図(例題10)3

①例題10。②でも書かれていますが、さきほどの例題7と基本的に同じで、 $4^2 = 16$ 種類の2連続塩基の出現確率の箱ひげ図(box plot)を追加で出力しています。③box plotはhoge10.pngに保存され、④その大きさは700×400と指定しています。⑤コード下部を表示。

```
(Rで)塩基配列解析
保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#int
fasta <- getSeq(genome) #ゲノム塩基配列情報を抽出し
names(fasta) <- seqnames(genome) #description情報を追加し
fasta #確認してるだけです

#本番
out <- dinucleotideFrequency(fasta, as.prob=T) #連続塩基の出現確率情報をoutに格納

#ファイルに保存(テキストファイル)
tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
write.table(tmp, out_f1, sep="\t", append=F, quote=F, row.names=F) #tmpの中身を指定したファイル名

#ファイルに保存(pngファイル)
png(out_f2, pointsize=13, width=param_fig[1], height=param_fig[2]) #出力ファイルの各種パラメータを
boxplot(out, ylab="Probability") #描画
grid(col="gray", lty="dotted") #指定したパラメータでグリッドを表示
dev.off() #おまじない
```



## 11. ヒトゲノム配列パッケージ([BSgenome.Hsapiens.NCBI.GRCh38](#))の場合 :

[トップページ](#)△

10.と基本的に同じですが、連続塩基の種類ごとの期待値とボックスプロット(box plot)上での色情報を組み合わせた(human\_2mertyt)を入力として利用し、色情報のみを取り出して利用しています。

# 作図(例題10)4

①の赤枠内が、染色体ごとの2連続塩基の出現確率情報を、②out\_f1で指定したファイル名(hoge10.txt)で出力している部分。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#intro\_general\_kmer\_2\_bios...

```

fasta <- getSeq(genome) #ゲノム塩基配列情報を抽出した結果をfastaに格納
names(fasta) <- seqnames(genome) #description情報を追加している
fasta #確認してるだけです

#本番
out <- dinucleotideFrequency(fasta, as.prob=T) #連続塩基の出現確率情報をoutに格納

#ファイルに保存(テキストファイル)
tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
write.table(tmp, out_f1, sep="\t", append=F, quote=F, row.names=F) #tmpの中身を指定したファイル名

#ファイルに保存(png)
png(out_f2, pointsize=13, width=param_fig[1], height=param_fig[2]) #出力ファイルの各種パラメータを
boxplot(out, ylab="Probability") #描画
grid(col="gray", lty="dotted") #指定したパラメータでグリッドを表示
dev.off() #おまじない

```

## 11. ヒトゲノム配列パッケージ(BSgenome.Hsapiens.NCBI.GRCh38)の場合 :

[トップページ](#)

10.と基本的に同じですが、連続塩基の種類ごとの期待値とボックスプロット(box plot)上での色情報を組み合わせたファイル(human\_2m.txt)を入力として利用し、色情報のみを取り出して利用しています。

# 作図(例題10)5

①の赤枠内が、染色体ごとの2連続塩基の出現確率情報を、②out\_f1で指定したファイル名(hoge10.txt)で出力している部分。③の赤枠内が、箱ひげ図を、④out\_f2で指定したファイル名(hoge10.png)で出力している部分。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#int

```

fasta <- getSeq(genome) #ゲノム塩基配列情報を抽出した結果をfastaに格納
names(fasta) <- seqnames(genome) #description情報を追加している
fasta #確認してるだけです

#本番
out <- dinucleotideFrequency(fasta, as.prob=T) #連続塩基の出現確率情報をoutに格納

#ファイルに保存(テキストファイル)
tmp <- cbind(names(fasta), out) #保存したい情報をtmpに格納
write.table(tmp, out_f1, sep="\t", append=F, quote=F, row.names=F) #tmpの中身を指定したファイル名

#ファイルに保存(pngファイル)
png(out_f2, pointsize=13, width=param_fig[1], height=param_fig[2]) #出力ファイルの各種パラメータを
boxplot(out, ylab="Probability") #描画
grid(col="gray", lty="dotted") #指定したパラメータでグリッドを表示
dev.off() #おまじない
    
```

## 11. ヒトゲノム配列パッケージ(BSgenome.Hsapiens.NCBI.GRCh38)の場合 :

[トップページ](#)

10.と基本的に同じですが、連続塩基の種類ごとの期待値とボックスプロット(box plot)上での色情報を組み合わせたファイル(human\_2m.txt)を入力として利用し、色情報のみを取り出して利用しています。

# 作図(例題10)6

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#intro\_general\_kmer\_2\_bios...

## 10. ヒトゲノム配列パッケージ(BSgenome.Hsapiens.NCBI.GRCh38)の場合 :

7.と基本的に同じですが、box plotのPNGファイルも出力しています。

```

out_f1 <- "hoge10.txt"           #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge10.png"         #出力ファイル名を指定してout_f2に格納
param_bsgenome <- "BSgenome.Hsapiens.NCBI.GRCh38" #パッケージ名を指定(BSgenome系のゲノムパッケージ)
param_fig <- c(700, 400)       #ファイル出力時の横幅と縦幅を指定(単位はピクセル)

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

#前処理(指定したパッケージ中のオブジェクト名をgenomeに統一)
tmp <- ls(paste("package", param_bsgenome, sep=":")) #指定したパッケージで利用可能なオブジェクト名を
genome <- eval(parse(text=tmp)) #文字列tmpをRオブジェクトとしてgenomeに格納(パッケージ中にオ
fasta <- getSeq(genome)       #ゲノム塩基配列情報を抽出した結果をfastaに格納
names(fasta) <- seqnames(genome) #description情報を追加している
fasta                        #確認してるだけです

#本番
out <- dinucleotideFrequency(fasta, as.prob=T) #連続塩基の出現確率情報をoutに格納
    
```

[トップページ](#) ^

# 作図(例題10)7

①例題10のコード全体をコピー実行。例題7と同じく、数分かかります。実行完了後。②エディタ画面上にあるこれらは、適宜終了してください。

The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The toolbar contains icons for file operations and navigation. The editor window shows a text file with the following content:

```

1 | AA AC AG AT CA CC CG CT GA GC GG GT TA TC
   | TG TT
2 | contig_1 0 0.0434782608695652 0.0434782608695652 0
   | .0869565217391304 0.0869565217391304 0.0869565217391304
1:1 |
    
```

The console window shows the following R commands and their output:

```

es=F)#tmpの中身を指定したファイル名で保存
>
> #ファイルに保存(pngファイル)
> png(out_f2, pointsize=13, width=param_fig[1], height=param_fi
g[2])#出力ファイルの各種パラメータを指定
> boxplot(out, ylab="Probability") #描画
> grid(col="gray", lty="dotted") #指定したパラメータでグリッ
ドを表示
> dev.off() #おまじない
null device
      1
>
>
    
```

The file explorer on the right shows the directory structure for the 'hoge' folder:

Name	Size
..	
hoge1.txt	221 B
hoge10.png	6.2 KB
hoge10.txt	142.1 KB
hoge2.txt	945 B
hoge4.fa	299 B
hoge7.txt	142.1 KB
hoge7.xlsx	441.3 KB
seq_20.fasta	29 B

# 作図(例題10)8

①例題10のコード全体をコピー実行。例題7と同じく、数分かかります。実行完了後。②エディタ画面上にあるこれらは、適宜終了してください。③は保存するか聞かれますが、④好きにしてください。

The screenshot shows the RStudio interface. The script editor contains the following R code:

```
7 #入力ファイルを読み込む
8 fasta <- read.fasta("seq_20.fasta")
9
10 #ファイルに保存(pngファイル)
> png(out_f2, pointsize=13, width=param_fig[1], height=param_fig[2])#出力ファイルの各種パラメータを指定
> boxplot(out, ylab="Probability") #描画
> grid(col="gray", lty="dotted") #指定したパラメータでグリッドを表示
> dev.off() #おまじない
null device
1
>
> |
```

The console shows the execution of the code, resulting in a plot and the message "null device".

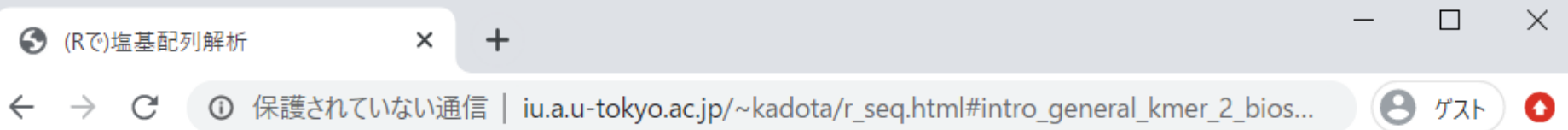
A dialog box titled "Untitled1 - Unsaved Changes" is displayed, asking "Do you want to save these changes?". The "Save" button is highlighted with a red box and a red arrow labeled ④. The "Save" option in the dialog is also highlighted with a red box and a red arrow labeled ③.

The file explorer on the right shows the directory structure: C:\Users\kadota\Desktop\hoge. The files listed are:

Name	Size
..	
hoge1.txt	221 B
hoge10.png	6.2 KB
hoge10.txt	142.1 KB
hoge2.txt	945 B
hoge4.fa	299 B
hoge7.txt	142.1 KB
hoge7.xlsx	441.3 KB
seq_20.fasta	29 B

# 作図(例題10)9

出力ファイルをパワポに貼り付けると、確かに②横幅700 pixels、③縦幅400 pixelsっぽくなっていることがわかる。



## 10. ヒトゲノム配列パッケージ(BSgenome.Hsapiens.NCBI.GRCh38)の場合 :

7.と基本的と同じですが、box plotのPI

```
out_f1 <- "hoge10.txt"
out_f2 <- "hoge10.png"
param_bsgenome <- "BSgenome.Hsapiens.NCBI.GRCh38"
param_fig <- c(700, 400)
```

#必要なパッケージ

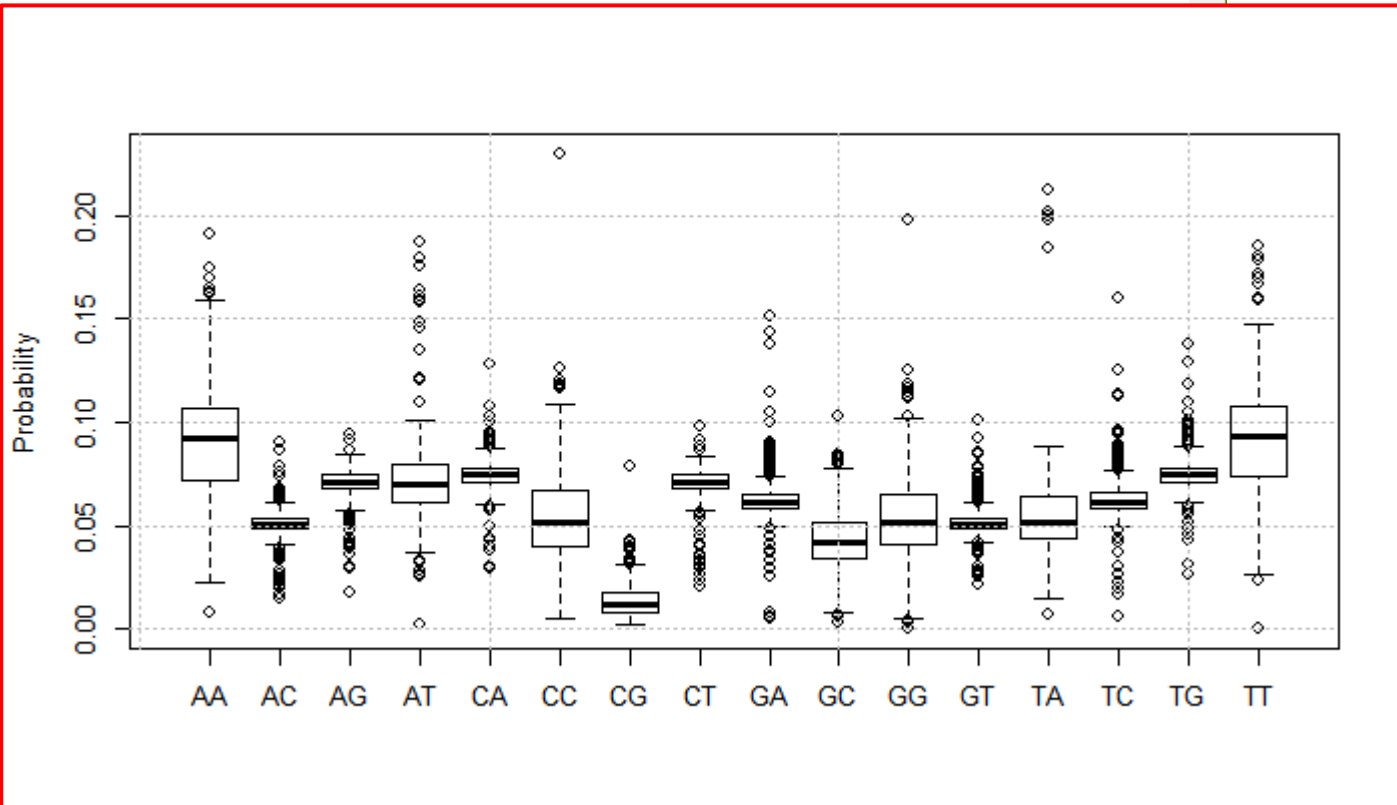
```
library(Biostrings)
library(param_bsgenome, character.only = TRUE)
```

#前処理(指定したパッケージ中のオブジェクトをロード)

```
tmp <- ls(paste("package", param_bsgenome))
genome <- eval(parse(text=tmp))
fasta <- getSeq(genome)
names(fasta) <- seqnames(genome)
fasta
```

#本番

```
out <- dinucleotideFrequency(fasta)
```



# 作図(例題10)10

出力ファイルをパワポに貼り付けると、確かに②横幅700 pixels、③縦幅400 pixelsっぽくなっていることがわかる。④AとTのみから構成される場合の観測値は、各塩基の出現確率(0.295)を掛けた期待値(8.7%)に比較的近い位置にプロットされていることがわかる。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#int

## 10. ヒトゲノム配列パッケージ(BSgenome.Hsapiens.NCBI.GRCh38)の場合

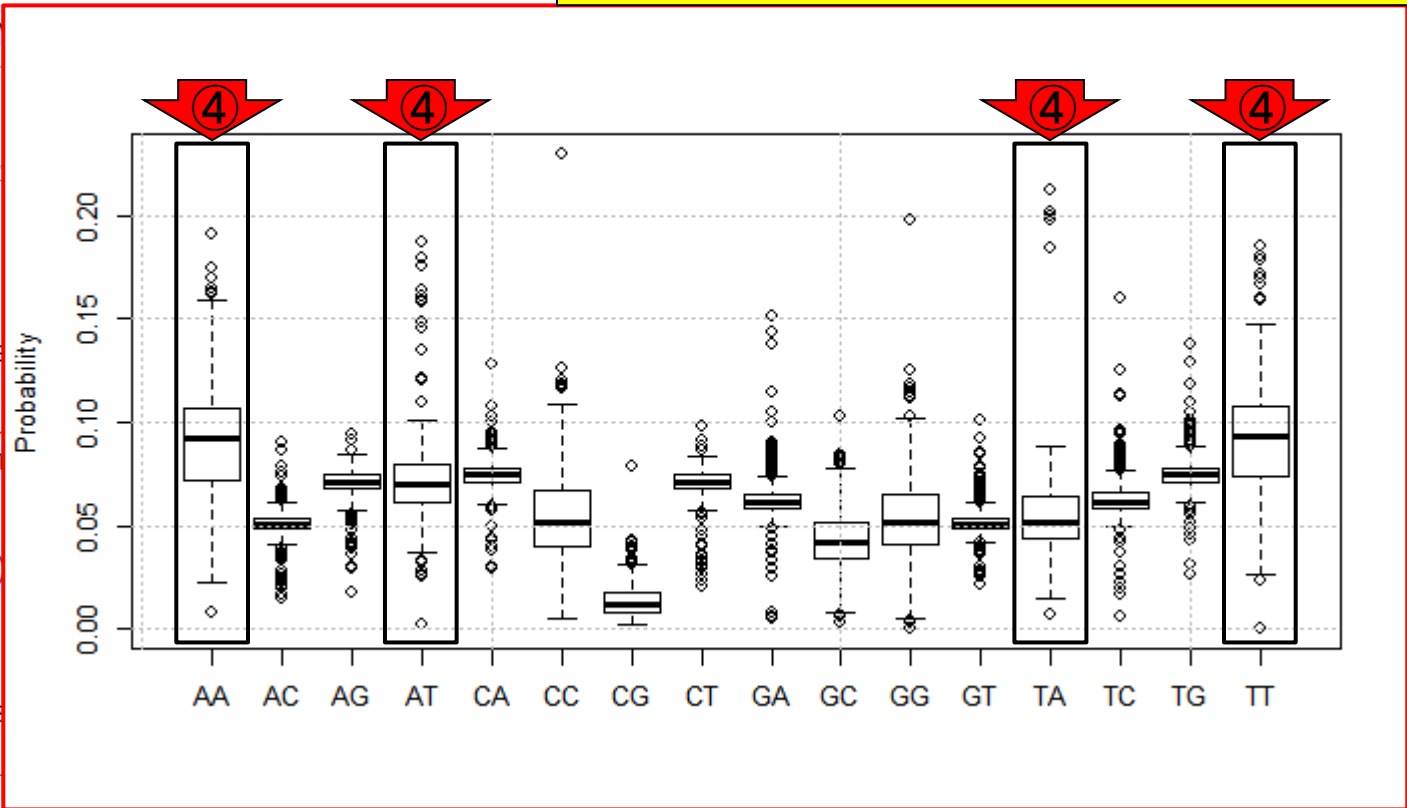
7.と基本的と同じですが、box plotのP

```
out_f1 <- "hoge10.txt"
out_f2 <- "hoge10.png"
param_bsgenome <- "BSgenome.Hsapiens.NCBI.GRCh38"
param_fig <- c(700, 400)
```

```
#必要なパッケージをロード
library(Biostrings)
library(param_bsgenome, character.only = TRUE)
```

```
#前処理(指定したパッケージ中のオブジェクトを読み込む)
tmp <- ls(paste("package", param_bsgenome))
genome <- eval(parse(text=tmp))
fasta <- getSeq(genome)
names(fasta) <- seqnames(genome)
fasta
```

```
#本番
out <- dinucleotideFrequency(fasta)
```





# 作図(例題10)11

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#int

## 10. ヒトゲノム配列パッケージ(BSgenome.Hsapiens.NCBI.GRCh38)の場合

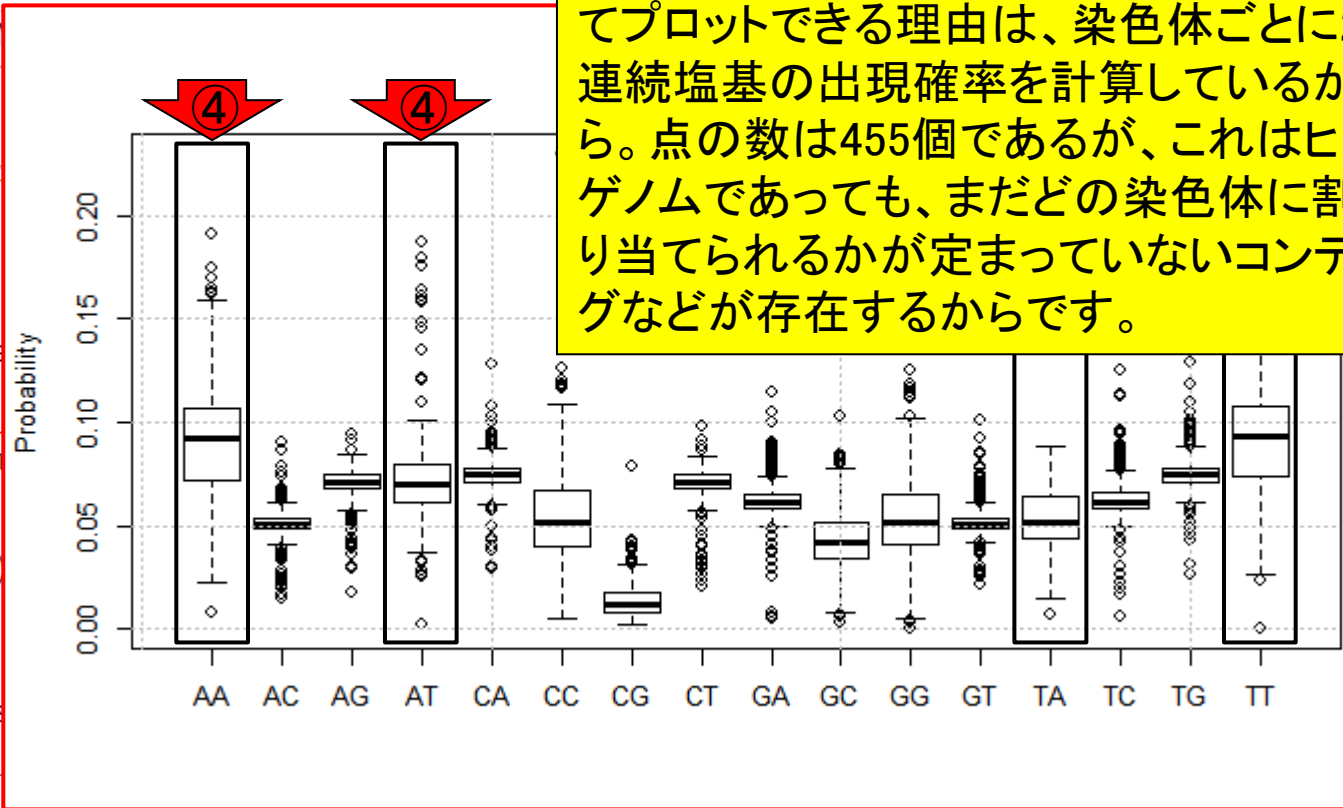
7.と基本的と同じですが、box plotのP

```
out_f1 <- "hoge10.txt"
out_f2 <- "hoge10.png"
param_bsgenome <- "BSgenome.Hsap
param_fig <- c(700, 400)
```

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

```
#前処理(指定したパッケージ中のオブジ
tmp <- ls(paste("package", param
genome <- eval(parse(text=tmp))
fasta <- getSeq(genome)
names(fasta) <- seqnames(genome)
fasta
```

```
#本番
out <- dinucleotideFrequency(fas
```



出力ファイルをパワポに貼り付けると、確かに②横幅700 pixels、③縦幅400 pixelsっぽくなっていることがわかる。④AとTのみから構成される場合の観測値は、各塩基の出現確率(0.295)を掛けた期待値(8.7%)に比較的近い位置にプロットされていることがわかる。尚、箱ひげ図としてプロットできる理由は、染色体ごとに2連続塩基の出現確率を計算しているから。点の数は455個であるが、これはヒトゲノムであっても、まだどの染色体に割り当てられるかが定まっていないコンティグなどが存在するからです。

# Contents

- Introduction、出現頻度解析( $k=2$ )、出現頻度解析( $k=1$ )
- $k=1$ で実践、multi-FASTAファイル、他の例題を実行
- $k=2$ で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8( $k=2$ )、例題9( $k=3$ )、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11( $k=10$ )、例題12( $k=10$ )、シークエンスエラーを含む場合

# 作図(例題11)1

(Rで)塩基配列解析

保護されていない通信 | [iu.a.u-tokyo.ac.jp/~kadota/r\\_seq.html#intro\\_general\\_kmer\\_2\\_bios...](http://iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#intro_general_kmer_2_bios...)

## 11. ヒトゲノム配列パッケージ([BSgenome.Hsapiens.NCBI.GRCh38](#))の場合 :

10.と基本的に同じですが、連続塩基の種類ごとの期待値とボックスプロット(box plot)上での色情報を含むファイル ([human\\_2mer.txt](#))を入力として利用し、色情報のみを取り出して利用しています。

```

in_f <- "human_2mer.txt"           #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge11.txt"            #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge11.png"           #出力ファイル名を指定してout_f2に格納
param_bsgenome <- "BSgenome.Hsapiens.NCBI.GRCh38" #パッケージ名を指定(BSgenome系のゲノムパッケージ)
param_fig <- c(700, 400)         #ファイル出力時の横幅と縦幅を指定(単位はピクセル)

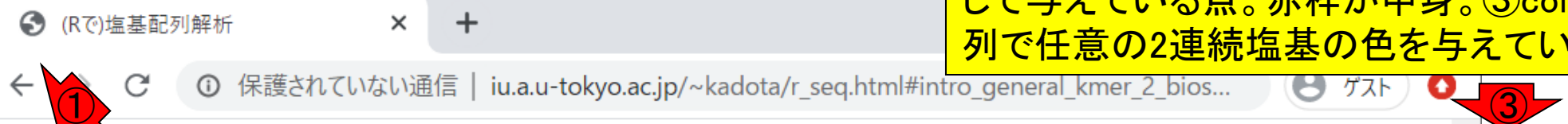
#必要なパッケージをロード
library(Biostrings)              #パッケージの読み込み
library(param_bsgenome, character.only=T) #指定したパッケージの読み込み

#入力ファイルの読み込み
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #in_fで指定したファイルの読み込み

#前処理(指定したパッケージ中のオブジェクト名をgenomeに統一)
tmp <- ls(paste("package", param_bsgenome, sep=":")) #指定したパッケージで利用可能なオブジェクト名を抽出
genome <- eval(parse(text=tmp)) #文字列tmpをRオブジェクトとしてgenomeに格納(パッケージ中のオブジェクト)
fasta <- getSeq(genome)          #ゲノム塩基配列情報を抽出した結果をfastaに格納 トップページ
names(fasta) <- seqnames(genome) #description情報を追加している
    
```

# 作図(例題11)2

①例題11。これは2連続塩基ごとに任意の色で表示させる際に利用するコード。例題10との違いは、②を入力ファイルとして与えている点。赤枠が中身。③color列で任意の2連続塩基の色を与えている。



## 11. ヒトゲノム配列パッケージ([BSgenome.Hsapiens.NCBI.GRCh38](#))の場合 :

10.と基本的に同じですが、連続塩基の種類ごとの期待値とボックスプロット(box plot)上での色情報 ([human\\_2mer.txt](#))を入力として利用し、色情報のみを取り出して利用しています。

```

in_f <- "human_2mer.txt" #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge11.txt" #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge11.png" #出力ファイル名を指定してout_f2に格納
param_bsgenome <- "BSgenome.Hsapiens.NCBI.GRCh38" #パッケージ名を指定(BSgenome系の)
param_fig <- c(700, 400) #ファイル出力時の横幅と縦幅を指定(単位はピクセル)

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

#入力ファイルの読み込み
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #in_fで指定

#前処理(指定したパッケージ中のオブジェクト名をgenomeに統一)
tmp <- ls(paste("package", param_bsgenome, sep=":")) #指定したパッケージで利用可能なオブジェクト
genome <- eval(parse(text=tmp)) #文字列tmpをRオブジェクトとしてgenomeに格納
fasta <- getSeq(genome) #ゲノム塩基配列情報を抽出した結果をfastaに格納
names(fasta) <- seqnames(genome) #description情報を追加している
    
```

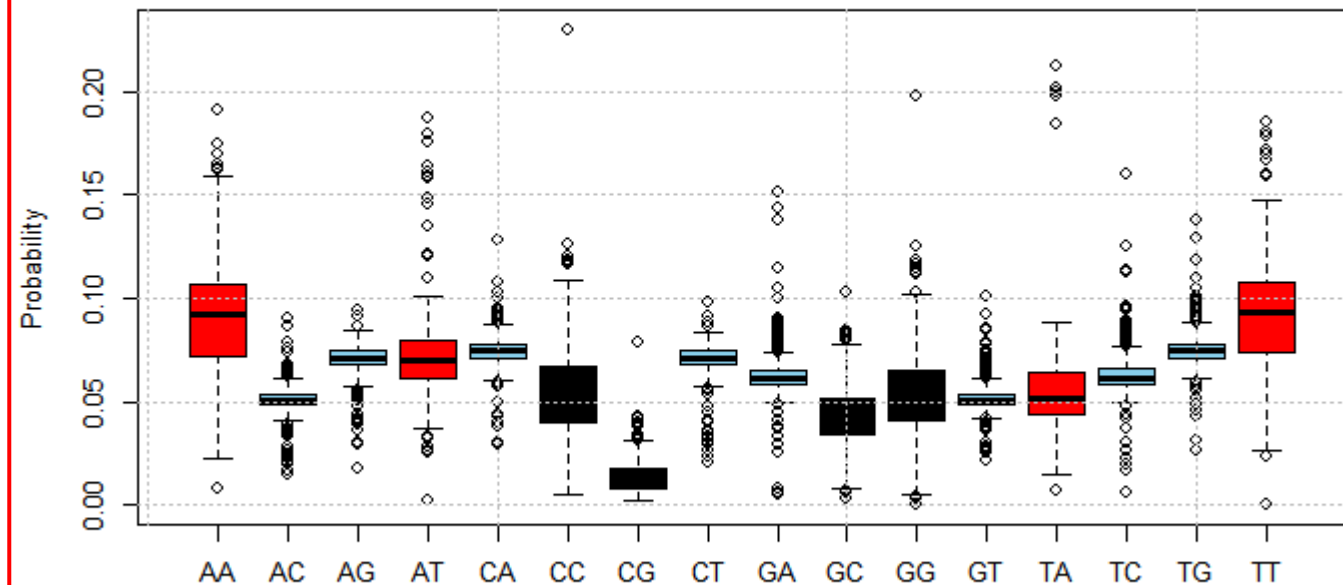
type	expected	color
AA	0.087025	red
AC	0.060475	skyblue
AG	0.060475	skyblue
AT	0.087025	red
CA	0.060475	skyblue
CC	0.042025	black
CG	0.042025	black
CT	0.060475	skyblue
GA	0.060475	skyblue
GC	0.042025	black
GG	0.042025	black
GT	0.060475	skyblue
TA	0.087025	red
TC	0.060475	skyblue
TG	0.060475	skyblue
TT	0.087025	red

# 作図(例題11)3

②

type	expected	color
AA	0.087025	red
AC	0.060475	skyblue
AG	0.060475	skyblue
AT	0.087025	red
CA	0.060475	skyblue
CC	0.042025	black
CG	0.042025	black
CT	0.060475	skyblue
GA	0.060475	skyblue
GC	0.042025	black
GG	0.042025	black
GT	0.060475	skyblue
TA	0.087025	red
TC	0.060475	skyblue
TG	0.060475	skyblue
TT	0.087025	red

①例題11の出力ファイル(hoge11.png)。確かに②で指定した通りの色になっていることがわかる。全体を眺めると、A or Tで構成されるもの(赤色)が全体的に高い値となり、C or Gで構成されるもの(黒色)が全体的に低い値となっていることがわかる。これはGC含量が41%という事実からも妥当。

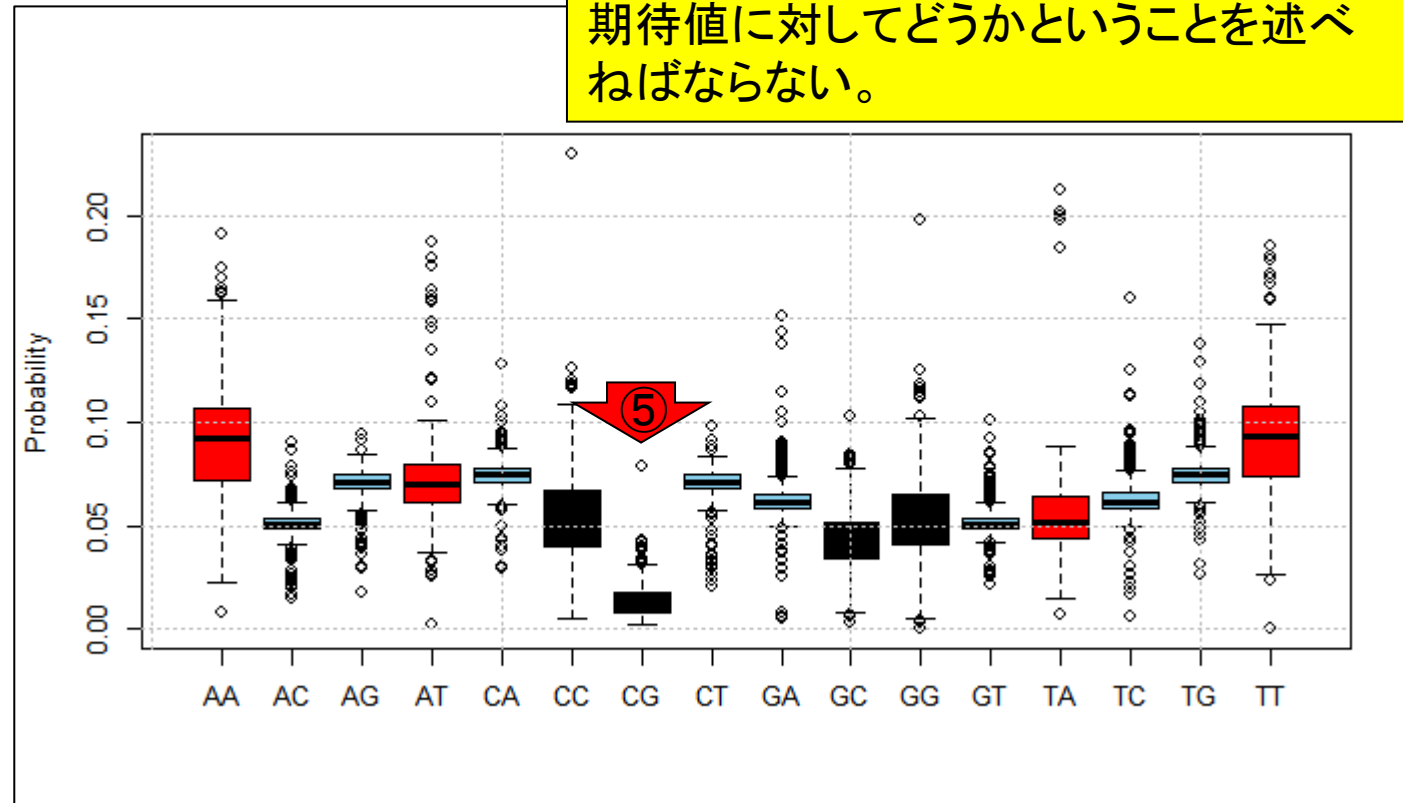


# 作図(例題11)4

④

type	expected	color
AA	0.087025	red
AC	0.060475	skyblue
AG	0.060475	skyblue
AT	0.087025	red
CA	0.060475	skyblue
CC	0.042025	black
CG	0.042025	black
CT	0.060475	skyblue
GA	0.060475	skyblue
GC	0.042025	black
GG	0.042025	black
GT	0.060475	skyblue
TA	0.087025	red
TC	0.060475	skyblue
TG	0.060475	skyblue
TT	0.087025	red

それゆえ、③原著論文でも書かれていたように、ゲノム中の各塩基の出現確率を算出しておき、④「連続塩基の種類ごとに各塩基の出現確率を掛け合わせた期待値(expected)」を考慮に入れねばならない。⑤CGという連続塩基の出現確率がただ低いと主張するのではなく、期待値に対してどうかということを述べねばならない。



③

# Contents

- Introduction、出現頻度解析( $k=2$ )、出現頻度解析( $k=1$ )
- $k=1$ で実践、multi-FASTAファイル、他の例題を実行
- $k=2$ で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8( $k=2$ )、例題9( $k=3$ )、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11( $k=10$ )、例題12( $k=10$ )、シークエンスエラーを含む場合

# 作図(例題12)1

(Rで)塩基配列解析

保護されていない通信 | [iu.a.u-tokyo.ac.jp/~kadota/r\\_seq.html#intro\\_general\\_kmer\\_2\\_bios...](http://iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#intro_general_kmer_2_bios...)

## 12. ヒトゲノム配列パッケージ([BSgenome.Hsapiens.NCBI.GRCh38](#))の場合:

11.と基本的に同じですが、[human\\_2mer.txt](#)というファイルを入力として与えて、連続塩基の種類ごとの期待値とボックスプロット(box plot)上での色情報を利用しています。また、重要なのは期待値からの差分であり、「プロットも期待値(expected)と同程度の観測値(observed)であればゼロ、観測値のほうが大きければプラス、観測値のほうが小さければマイナス」といった具合で表現したほうがスマートです。それゆえ、box plotの縦軸を $\log(\text{observed}/\text{expected})$ として表現しています。CG以外の連続塩基は縦軸上で0近辺に位置していることがわかります。

```
in_f <- "human_2mer.txt"           #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge12.txt"            #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge12.png"           #出力ファイル名を指定してout_f2に格納
param_bsgenome <- "BSgenome.Hsapiens.NCBI.GRCh38" #パッケージ名を指定(BSgenome系のゲノムパッケージ)
param_fig <- c(700, 400)          #ファイル出力時の横幅と縦幅を指定(単位はピクセル)

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

#入力ファイルの読み込み
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #in_fで指定したファイルの読み込み

#前処理(指定したパッケージ中のオブジェクト名をgenomeに統一)
```

[トップページ](#) ↑



# 作図(例題12)2

①例題12は、期待値を考慮に入れてプロットするやり方。縦軸を「観測値」ではなく「 $\log(\text{観測値}/\text{期待値})$ 」とすることで、②のような解釈ができるようにしています。このような期待値で割ってlogをとるのはよくやられる戦略なので感覚をつかんでおくとよいでしょう。

(Rで)塩基配列解析

保護されていない通信 | [iu.a.u-tokyo.ac.jp/~kadota/r\\_seq.html#int](http://iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#int)

## 12. ヒトゲノム配列パッケージ([BSgenome.Hsapiens.NCBI.GRCh38](#))の場合

11.と基本的に同じですが、[human\\_2mer.txt](#)というファイルを入力として与えて、連続塩基の種類ごとの期待値とボックスプロット(box plot)上での色情報を利用しています。また、重要なのは期待値からの差分であり、「プロットも期待値(expected)と同程度の観測値(observed)であればゼロ、観測値のほうが大きければプラス、観測値のほうが小さければマイナス」といった具合で表現したほうがスマートです。それゆえ、box plotの縦軸を $\log(\text{observed}/\text{expected})$ として表現しています。CG以外の連続塩基は縦軸上で0近辺に位置していることがわかります。

```
in_f <- "human_2mer.txt" #入力ファイル名を指定してin_fに格納
out_f1 <- "hoge12.txt" #出力ファイル名を指定してout_f1に格納
out_f2 <- "hoge12.png" #出力ファイル名を指定してout_f2に格納
param_bsgenome <- "BSgenome.Hsapiens.NCBI.GRCh38" #パッケージ名を指定(BSgenome系のゲノムパッケージ)
param_fig <- c(700, 400) #ファイル出力時の横幅と縦幅を指定(単位はピクセル)

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

#入力ファイルの読み込み
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #in_fで指定したファイルの読み込み

#前処理(指定したパッケージ中のオブジェクト名をgenomeに統一)
```

# 作図(例題12)3

例題12のコピペ実行結果の、①Console画面。②警告メッセージがでており、③その理由として外れ値(-Inf)が存在するようだ。-Infというのは「マイナス無限大(infinity)」のこと。log(観測値/期待値)で計算しているので、logの特性をある程度理解していれば、「観測値が0のものがあるのだろう」と簡単に原因の解読ができる。実際に出力ファイル(hoge12.txt)を眺めると、「TT」の列において、「HSCRUN\_RANDOM\_146」というコンティグが-Infになっていることが確認できる。ヒトゲノムの場合は、21番染色体(xとyを含めてもよい)までで同様の計算を行えば、このような警告は出ない筈。

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Console Terminal x Jobs x
C:/Users/kadota/Desktop/hoge/
> tmp <- cbind(names(fasta), logratio) #保存したい情報をtmpに格納
> write.table(tmp, out_f1, sep="\t", append=F, quote=F, row.names=F) #tmpの中身を指定したファイル名で保存
>
> #ファイルに保存(pngファイル)
> png(out_f2, pointsize=13, width=param_fig[1], height=param_fig[2]) #出力ファイルの各種パラメータを指定
> boxplot(logratio, ylab="log2(observed/expected)", #描画
+ col=character(data$color)) #描画
警告メッセージ:
  bplot(at[i], wid = width[i], stats = z$stats[, i], out = z$out
[z$group == 1]で:
  外れ値 (-Inf) は描かれませんが、箱型図 16 中では
> grid(col="gray", lty="dotted") #指定したパラメータでグリッドを表示
> dev.off() #おまじない
null device
      1
>
> |
```

hoge.txt	227 B
hoge10.png	6.2 KB
hoge10.txt	142.1 KB
hoge2.txt	945 B
hoge4.fa	299 B
hoge7.txt	142.1 KB
hoge7.xlsx	441.3 KB
seq_20.fasta	29 B

# 作図(例題12)4

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#intro\_general\_kmer\_2\_bios...

12. ヒトゲノム配列パッケージ([BSgenome.Hsapiens.NCBI.GRCh38](#))の場合:

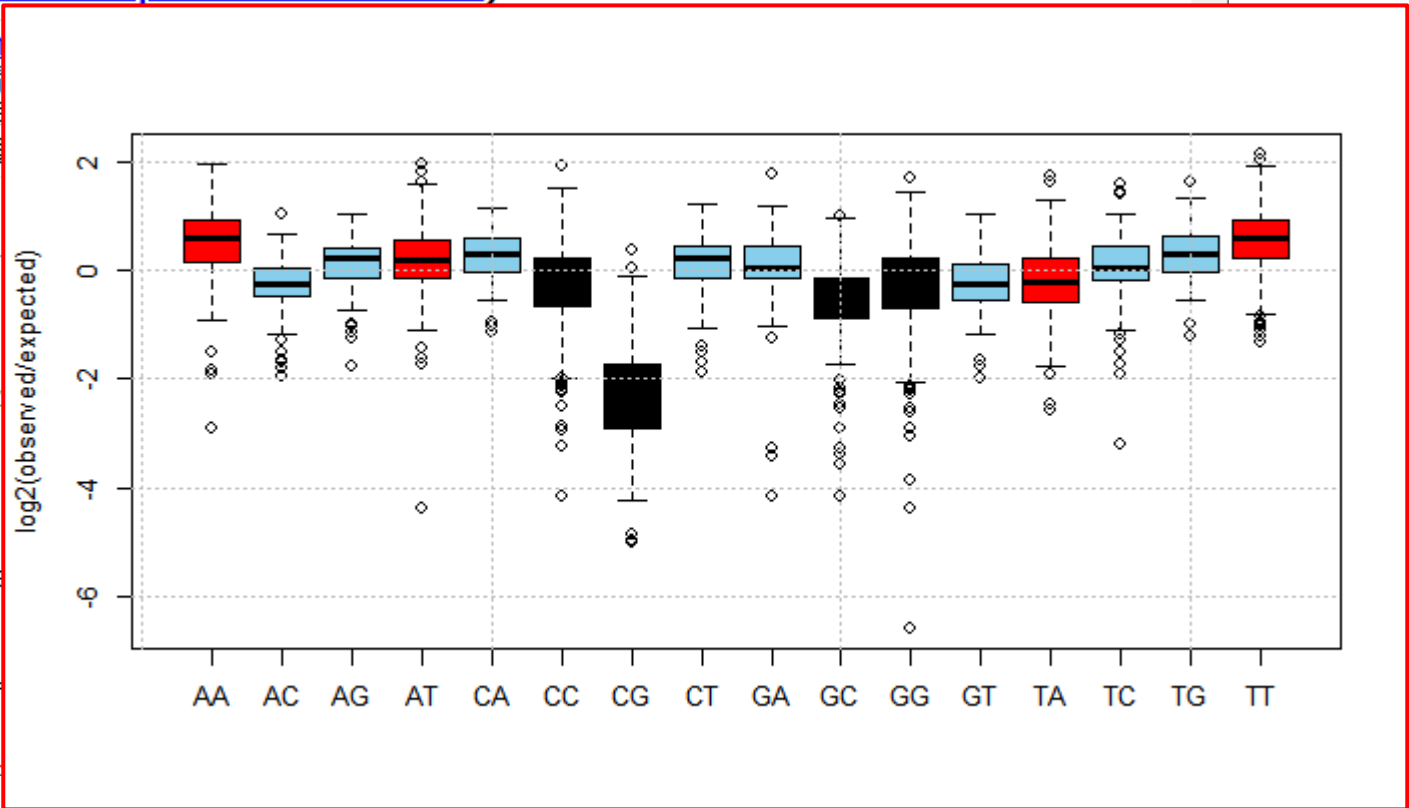
11.と基本的に同じですが、[human\\_2n](#)ボックスプロット(box plot)上での色情も期待値(expected)と同程度の観測(ほうが小さければマイナス)といった具log(observed/expected)として表現します。

```
in_f <- "human_2mer.txt"
out_f1 <- "hoge12.txt"
out_f2 <- "hoge12.png"
param_bsgenome <- "BSgenome.Hsapiens.NCBI.GRCh38"
param_fig <- c(700, 400)
```

```
#必要なパッケージをロード
library(Biostrings)
library(param_bsgenome, character.only=TRUE)
```

```
#入力ファイルの読み込み
data <- read.table(in_f, header=TRUE, as.is=TRUE)
```

```
#前処理(指定したパッケージ中のオブジェクト)
```



# 作図(例題12)5

①例題12の、②出力結果(hoge12.png)。③観測値(実際の2連続塩基の出現確率)と期待値(塩基ごとの出現確率を掛け合わせたもの)が似ている場合は、0付近にプロットされる。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#intro\_general\_kmer\_2\_bios...

## 12. ヒトゲノム配列パッケージ(BSgenome.Hsapiens.NCBI.GRCh38)の場合:

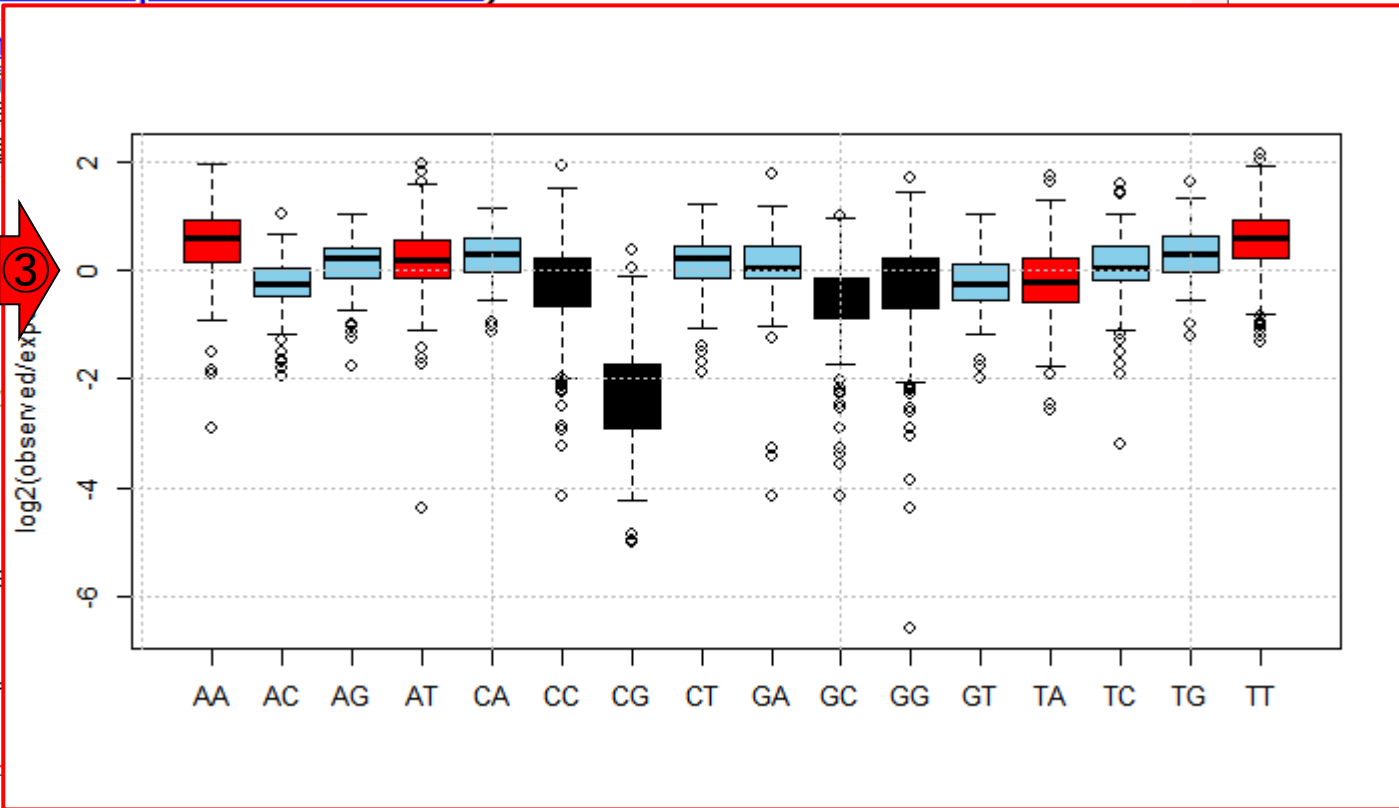
11.と基本的に同じですが、human\_2nボックスプロット(box plot)上での色情も期待値(expected)と同程度の観測(ほうが小さければマイナス)といった具合にlog(observed/expected)として表現されます。

```
in_f <- "human_2mer.txt"
out_f1 <- "hoge12.txt"
out_f2 <- "hoge12.png"
param_bsgenome <- "BSgenome.Hsapiens.NCBI.GRCh38"
param_fig <- c(700, 400)
```

```
#必要なパッケージをロード
library(Biostrings)
library(param_bsgenome, character.only=TRUE)
```

```
#入力ファイルの読み込み
data <- read.table(in_f, header=TRUE, as.is=TRUE)
```

```
#前処理(指定したパッケージ中のオブジェクト)
```



# 作図(例題12)6

①例題12の、②出力結果(hoge12.png)。③観測値(実際の2連続塩基の出現確率)と期待値(塩基ごとの出現確率を掛け合わせたもの)が似ている場合は、0付近にプロットされる。logの底は2で計算しているので、④縦軸が2に相当するのが、観測値のほうが期待値の $2^2 = 4$ 倍出現確率が高いことを意味する。

(Rで)塩基配列解析

保護されていない通信 | [iu.a.u-tokyo.ac.jp/~kadota/r\\_seq.html#int](http://iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#int)

## 12. ヒトゲノム配列パッケージ([BSgenome.Hsapiens.NCBI.GRCh38](#))の場合

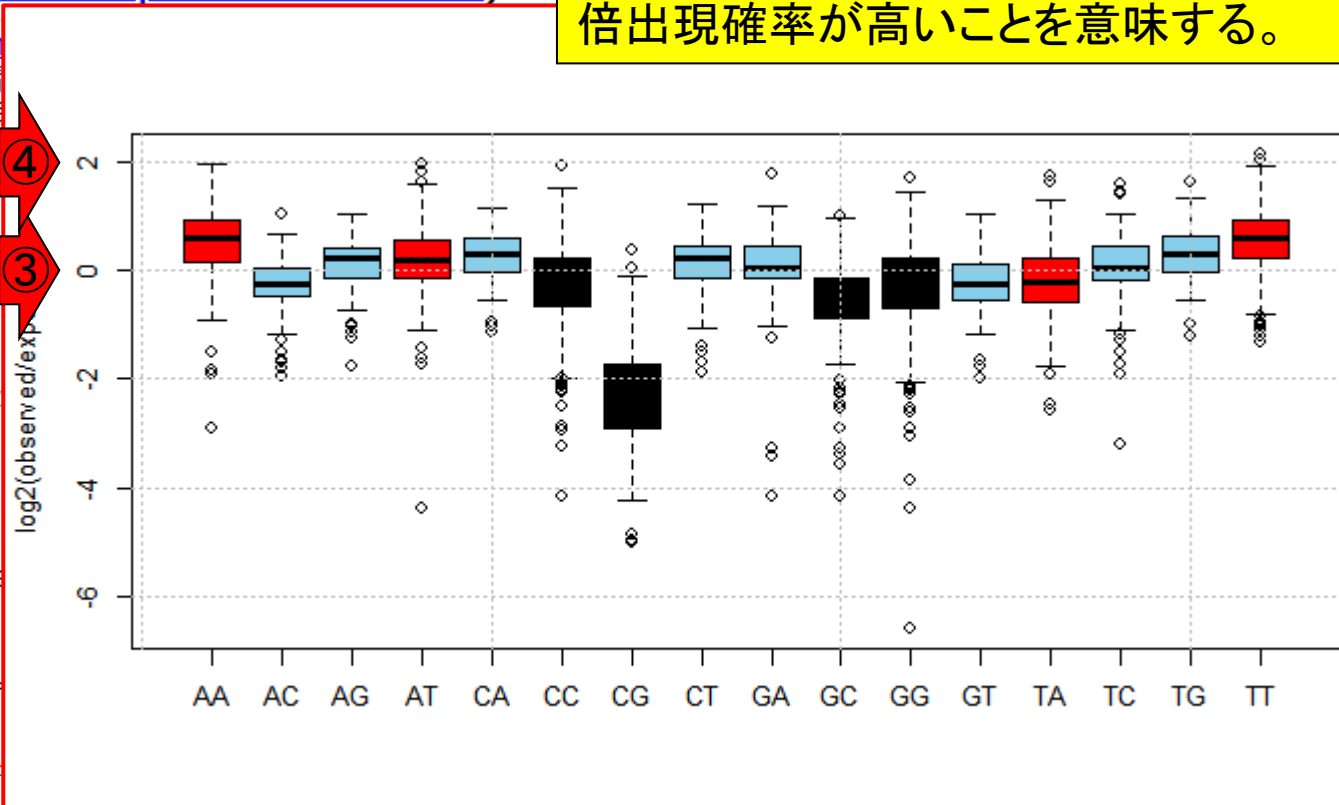
11.と基本的に同じですが、[human\\_2n](#)ボックスプロット(box plot)上での色情プロットも期待値(expected)と同程度の観測値(ほうが小さければマイナス)といった具合に $\log(\text{observed}/\text{expected})$ として表現されます。

```
in_f <- "human_2mer.txt"
out_f1 <- "hoge12.txt"
out_f2 <- "hoge12.png"
param_bsgenome <- "BSgenome.Hsapiens.NCBI.GRCh38"
param_fig <- c(700, 400)
```

```
#必要なパッケージをロード
library(Biostrings)
library(param_bsgenome, character.only = TRUE)
```

```
#入力ファイルの読み込み
data <- read.table(in_f, header = TRUE, as.is = TRUE)
```

```
#前処理(指定したパッケージ中のオブジェクト)
```



# 作図(例題12)7

①例題12の、②出力結果(hoge12.png)。③観測値(実際の2連続塩基の出現確率)と期待値(塩基ごとの出現確率を掛け合わせたもの)が似ている場合は、0付近にプロットされる。logの底は2で計算しているので、④縦軸が2に相当するのが、観測値のほうが期待値の $2^2 = 4$ 倍出現確率が高いことを意味する。同様に、⑤縦軸が-2に相当するのが、観測値のほうが期待値の $2^{-2} = 1/4$ 倍出現確率が高い(つまり4倍出現確率が低い)ことを意味する。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#int

## 12. ヒトゲノム配列パッケージ([BSgenome.Hsapiens.NCBI.GRCh38](#))の場合

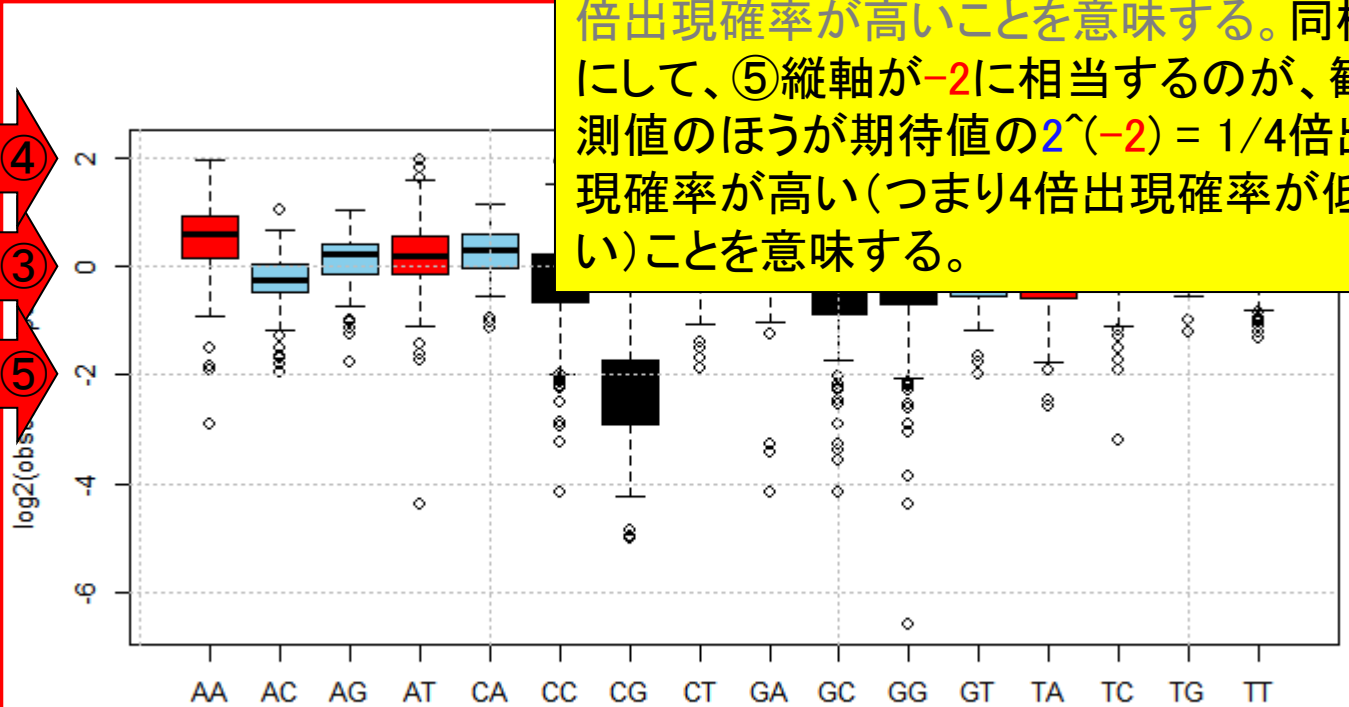
11.と基本的に同じですが、[human\\_2n](#)ボックスプロット(box plot)上での色情報も期待値(expected)と同程度の観測値(観測値のほうが小さければマイナス)といった具合に、 $\log(\text{observed}/\text{expected})$ として表現されます。

```
in_f <- "human_2mer.txt"
out_f1 <- "hoge12.txt"
out_f2 <- "hoge12.png"
param_bsgenome <- "BSgenome.Hsapiens.NCBI.GRCh38"
param_fig <- c(700, 400)
```

```
#必要なパッケージをロード
library(Biostrings)
library(param_bsgenome, character.only = TRUE)
```

```
#入力ファイルの読み込み
data <- read.table(in_f, header = TRUE, as.is = TRUE)
```

```
#前処理(指定したパッケージ中のオブジェクト)
```



# 作図(例題12)8

このように期待値の違いを補正した結果で眺めると、確かに⑥CGの出現確率が期待値よりも大幅に低いことがよくわかりますね。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#intro\_general\_kmer\_2\_bios...

## 12. ヒトゲノム配列パッケージ(BSgenome.Hsapiens.NCBI.GRCh38)の場合:

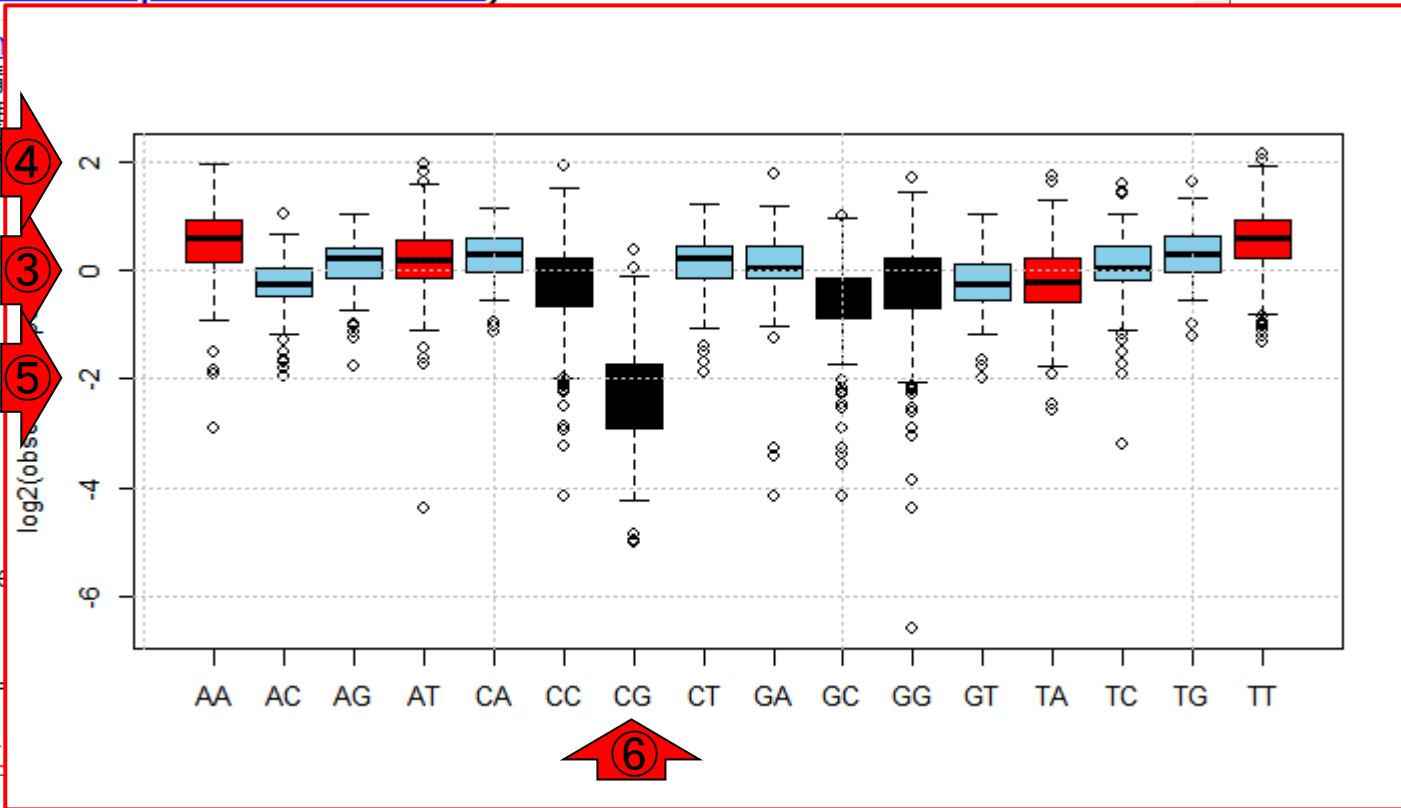
11.と基本的に同じですが、human\_2nボックスプロット(box plot)上での色情も期待値(expected)と同程度の観測(ほうが小さければマイナス)といった具合にlog(observed/expected)として表現されます。

```
in_f <- "human_2mer.txt"
out_f1 <- "hoge12.txt"
out_f2 <- "hoge12.png"
param_bsgenome <- "BSgenome.Hsapiens.NCBI.GRCh38"
param_fig <- c(700, 400)
```

```
#必要なパッケージをロード
library(Biostrings)
library(param_bsgenome, character.only=TRUE)
```

```
#入力ファイルの読み込み
data <- read.table(in_f, headers=TRUE, as.is=TRUE)
```

```
#前処理(指定したパッケージ中のオブジェクト)
```



# Contents

- Introduction、出現頻度解析( $k=2$ )、出現頻度解析( $k=1$ )
- $k=1$ で実践、multi-FASTAファイル、他の例題を実行
- $k=2$ で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8( $k=2$ )、例題9( $k=3$ )、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11( $k=10$ )、例題12( $k=10$ )、シークエンスエラーを含む場合



# 塩基配列解析の基礎

①では塩基配列解析の基礎的な項目を多く含んでいる。先ほどまで解説したGC含量の違いによる出現確率の期待値の違いなどがよくわからなかったヒトも、この後挙げるいくつかの項目を参考にして、仮想データを作成して解析してみると理解が深まるでしょう。

## (Rで)塩基配列解析

(last modified 2022/05/09, since 2010)

このウェブページの多くは、[インストール](#)についての推奨手順 ([Windows2022.03.31版](#)と[Macintosh2021.04.01版](#))に従ってフリーソフトRと必要なパッケージをインストール済みであるという前提で記述しています。初心者の方は[基本的な利用法](#)(Windows2022.04.03版の[PPTX](#)と[PDF](#) ; Macintosh2020.03.13版の[PPTX](#)と[PDF](#))で自習してください。

[@Agribio\\_utokyo](#)さんをフォロー

[アグリバイオ](#)、[\(Rで\)塩基配列解析のサブページ](#)、[生命情報解析研究室](#)

### What's new? ([過去のお知らせはこちら](#))

- 「解析 | 一般 | [パターンマッチング](#)」の例題5の入力ファイル名が間違っていたのを訂正しました (data\_seqlogo1.txt -> data\_seqlogo1.fasta)。 (中村 弘太 氏提供情報)(2022/05/09) **NEW**
- [東京大学・大学院農学生命科学研究科・応用生命工学専攻](#)の[令和5\(2023\)年度大学院学生募集公開ガイダンス](#)の第2回目は、5月28日(土)に開催します。(2022/05/08) **NEW** [トップページ](#)

# GC含量1

①では塩基配列解析の基礎的な項目を多く含んでいる。先ほどまで解説したGC含量の違いによる出現確率の期待値の違いなどがよくわからなかったヒトも、この後挙げるいくつかの項目を参考にして、仮想データを作成して解析してみると理解が深まるでしょう。例えば、②は配列ごとのGC含量を返す項目。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#

- [解析 | 一般 | アラインメント | マルチプル | について](#) (last modified 2019/07/29)
- [解析 | 一般 | アラインメント | マルチプル | DECIPHER\(Wright\\_2015\)](#) (last modified 2019/07/29)
- [解析 | 一般 | アラインメント | マルチプル | msa\(Bodenhofer\\_2015\)](#) (last modified 2016/12/29)
- [解析 | 一般 | Silhouette scores\(シルエットスコア\)](#) (last modified 2019/02/28)
- [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\(GC contents\)](#) ② (last modified 2015/09/12)
- [解析 | 一般 | Sequence logos | について](#) (last modified 2018/06/29)
- [解析 | 一般 | Sequence logos | seqLogo](#) (last modified 2019/04/21)
- [解析 | 一般 | Sequence logos | ggseqlogo\(Wagih\\_2017\)](#) (last modified 2018/06/29)
- [解析 | 一般 | 上流配列解析 | LDSS\(Yamamoto\\_2007\)](#) (last modified 2015/02/19)
- [解析 | 一般 | 上流配列解析 | Relative Appearance Ratio\(Yamamoto\\_2011\)](#) (last modified 2012/07/17)
- [解析 | ゲノム | について](#) (last modified 2019/09/27)
- [解析 | ゲノム | 領域の一致の評価 | regioneR\(Gel\\_2016\)](#) (last modified 2019/09/27)
- [解析 | 基礎 | k-mer | ゲノムサイズ推定\(基礎\) | qrgc](#) (last modified 2016/01/06)
- [解析 | 基礎 | 平均-分散プロット | について](#) (last modified 2015/11/11)
- [解析 | 基礎 | 平均-分散プロット | Technical replicates](#) (last modified 2014/02/18)

[トップページへ](#)

# GC含量2

①では塩基配列解析の基礎的な項目を多く含んでいる。先ほどまで解説したGC含量の違いによる出現確率の期待値の違いなどがよくわからなかったヒトも、この後挙げるいくつかの項目を参考にして、仮想データを作成して解析してみると理解が深まるでしょう。例えば、②は配列ごとのGC含量を返す項目。配列ごとではなく、入力ファイル全体のGC含量を得たい場合は、③の例題3が便利。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#

- イントロ | 一般 | 翻訳配列(translate)を取得 | [Biostrings](#) (last modified 2016/04/27)
- イントロ | 一般 | 翻訳配列(translate)を取得 | [seqinr\(Charif\\_2005\)](#) (last modified 2016/01/28)
- イントロ | 一般 | [相補鎖\(complement\)](#)を取得 (last modified 2019/03/10)
- イントロ | 一般 | [逆相補鎖\(reverse complement\)](#)を取得 (last modified 2019/03/10)
- イントロ | 一般 | [逆鎖\(reverse\)](#)を取得 (last modified 2019/03/10)
- イントロ | 一般 | k-mer解析 | k=1(塩基ごとの出現頻度解析) | [Biostrings](#) (last modified 2016/04/27)
- イントロ | 一般 | k-mer解析 | k=2(2連続塩基の出現頻度解析) | [Biostrings](#) (last modified 2016/01/28)
- イントロ | 一般 | k-mer解析 | k=3(3連続塩基の出現頻度解析) | [Biostrings](#) (last modified 2016/01/28)
- イントロ | 一般 | k-mer解析 | k=n(n連続塩基の出現頻度解析) | [Biostrings](#) (last modified 2016/05/01)
- 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 2017/04/11)
- イントロ | 一般 | 配列取得 | ゲノム配列 | [BSgenome](#) (last modified 2019/02/22)
- 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 2017/04/11)
- イントロ | 一般 | 配列取得 | プロモーター配列 | [BSgenomeとTxDbから](#) (last modified 2015/02) [トップページへ](#)
- イントロ | 一般 | 配列取得 | プロモーター配列 | [GenomicFeatures\(Lawrence\\_2013\)](#) (last modified 2016/05/01)

# GC含量3



①では塩基配列解析の基礎的な項目を多く含んでいる。先ほどまで解説したGC含量の違いによる出現確率の期待値の違いなどがよくわからなかったヒトも、この後挙げるいくつかの項目を参考にして、仮想データを作成して解析してみると理解が深まるでしょう。例えば、②は配列ごとのGC含量を返す項目。配列ごとではなく、入力ファイル全体のGC含量を得たい場合は、③の例題3が便利。③の、④例題3。⑤全配列をまとめて…と書かれているのがわかります。まずはこれをコピペ実行。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#int

## 3. 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mult

配列ごとではなく、全配列をまとめて出現頻度をカウントした結果を返すや

```
in_f <- "hoge4.fa" #入力ファイル名を指定して
out_f <- "hoge3.txt" #出力ファイル名を指定して
param_base <- c("A", "C", "G", "T", "N") #出力させたい塩基を指定

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

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

#本番
hoge <- alphabetFrequency(fasta) #A,C,G,T,..の数を各配列ごとにカウントした結果をhogeに格納
obj <- is.element(colnames(hoge), param_base) #条件を満たすかどうかを判定した結果をobjに格納
#out <- colSums(hoge[, obj]) #列ごとの総和をoutに格納
out <- apply(as.matrix(hoge[, obj]), 2, sum) #列ごとの総和をoutに格納

#ファイルに保存
tmp <- rbind(names(out), out) #保存したい情報をtmpに格納
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.names=F) #tmpの中身を指定して
```

[トップページ](#)

# GC含量4

コピー実行後に、①出力ファイル (hoge3.txt)を、②エディタで概観。③出力ファイルの中身は、④で作成したもののなので…

The screenshot shows the RStudio interface. The top-left pane displays a text file named 'hoge3.txt' with the following content:

	A	C	G	T	N
1	54	71	68	48	0
2					
3					

The bottom-left pane shows the R console with the following code and comments:

```
> hoge <- alphabetFrequency(fasta) #A,C,G,T,..の数を各配列ごとにカウントした結果をhogeに格納
> obj <- is.element(colnames(hoge), param_base)#条件を満たすかどうかを判定した結果をobjに格納
> #out <- colSums(hoge[, obj]) #列ごとの総和をoutに格納
> out <- apply(as.matrix(hoge[, obj]), 2, sum)#列ごとの総和をoutに格納
> #ファイルに保存
> tmp <- rbind(names(out), out) #保存したい情報をtmpに格納
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.names=F)#tmpの中身を指定
>
```

The bottom-right pane shows a file explorer view of the 'hoge' directory. The files listed are:

Name	Size
hoge12.png	5.9 KB
hoge12.txt	138.8 KB
hoge2.txt	945 B
hoge3.txt	26 B
hoge4.fa	299 B
hoge7.txt	142.1 KB
hoge7.xlsx	441.3 KB
human_2mer.txt	333 B
seq_20.fasta	29 B

Red arrows with circled numbers indicate key actions: ② points to the file editor, ④ points to the console code, ③ points to the save command, and ① points to the 'hoge3.txt' file in the explorer.

# GC含量5

コピー実行後に、①出力ファイル (hoge3.txt)を、②エディタで概観。③出力ファイルの中身は、④で作成したものである、⑤のようにoutの中身を表示させると分かり易い。

The screenshot shows the RStudio interface. At the top, the menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and a search bar. The main editor window displays a text file named 'hoge3.txt' with the following content:

1	A	C	G	T	N
2	54	71	68	48	0
3					

The console window shows the following R code and output:

```
> #out <- colSums(hoge[, obj]) #列ごとの総和をoutに格納
> out <- apply(as.matrix(hoge[, obj]), 2, sum)#列ごとの総和をoutに格納
> #ファイルに保存
> tmp <- rbind(names(out), out) #保存したい情報をtmpに格納
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, #tmpの中身を指定
> out
  A C G T N
54 71 68 48 0
> |
```

The file explorer on the right shows the directory 'C:\Users\kadota\Desktop\hoge' containing several files, including 'hoge3.txt' which is highlighted with a red arrow labeled ①.

# GC含量6

コピー実行後に、①出力ファイル (hoge3.txt)を、②エディタで概観。③出力ファイルの中身は、④で作成したものであるため、⑤のようにoutの中身を表示させると分かり易い。GC含量は、この場合は⑥  $(71 + 68) / (54 + 71 + 68 + 48) = 0.5767635$ のように計算すればよいだけです。⑥の数式はNをカウントしていますが、おそらく実際には含めないほうがよいです。

The screenshot shows the RStudio interface. At the top, the menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and a search bar. The main editor window displays a text file named 'hoge3.txt' with the following content:

```
1 A C G T N
2 54 71 68 48 0
3
```

The console window shows the following R code and output:

```
> out <- apply(as.matrix(hoge[, obj]), 2, sum)#列ごとの総和をoutに格納
> #ファイルに保存
> tmp <- rbind(names(out), out) #保存したい情報をtmpに格納
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, #5 names=F)#tmpの中身を指定
> out
  A C G T N
54 71 68 48 0
> (71+68)/sum(out) #6
[1] 0.5767635
> |
```

The file explorer on the right shows the directory 'C:\Users\kadota\Desktop\hoge' containing several files, including 'hoge3.txt' which is highlighted with a red arrow labeled ①.

# GC含量7

コピー実行後に、①出力ファイル (hoge3.txt)を、②エディタで概観。③出力ファイルの中身は、④で作成したものである、⑤のようにoutの中身を表示させると分かり易い。GC含量は、この場合は⑥ $(71 + 68) / (54 + 71 + 68 + 48) = 0.5767635$ のように計算すればよいだけです。⑥の数式はNをカウントするようにしていますが、おそらく実際には含めないほうがよいです。⑦こんな感じで入力すれば、塩基の種類ごとの出現確率情報も取得可能。

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
Console Terminal x Jobs x
C:/Users/kadota/Desktop/hoge/
ごとにカウントした結果をhogeに格納
> obj <- is.element(colnames(hoge), param_base)#条件を満たすかどうかを判定した結果をobjに格納
> #out <- colSums(hoge[, obj]) #列ごとの総和をoutに格納
> out <- apply(as.matrix(hoge[, obj]), 2, sum)#列ごとの総和をoutに格納
> #ファイルに保存
> tmp <- rbind(names(out), out) #保存したい情報をtmpに格納
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.names=F)#tmpの中身を指定
> out
  A  C  G  T  N
54 71 68 48  0
> (71+68)/sum(out)
[1] 0.5767635
> out/sum(out)
      A      C      G      T      N
0.2240664 0.2946058 0.2821577 0.1991701 0.0000000
  
```

Name	Size
hoge12.png	5.9 KB
hoge12.txt	138.8 KB
hoge2.txt	945 B
hoge3.txt	26 B
hoge4.fa	299 B
hoge7.txt	142.1 KB
hoge7.xlsx	441.3 KB
human_2mer.txt	333 B
seq_20.fasta	29 B



# Contents

- Introduction、出現頻度解析( $k=2$ )、出現頻度解析( $k=1$ )
- $k=1$ で実践、multi-FASTAファイル、他の例題を実行
- $k=2$ で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8( $k=2$ )、例題9( $k=3$ )、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11( $k=10$ )、例題12( $k=10$ )、シークエンスエラーを含む場合

# ランダム配列を生成1

(Rで)塩基配列解析

保護されていない通信 | [iu.a.u-tokyo.ac.jp/~kadota/r\\_seq.html#](http://iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#)

- [インストール | Rパッケージ | 個別\(2018年11月以前\)](#) (last modified 2018/11/12)
- [基本的な利用法](#) (last modified 2020/03/06) **NEW**
- [サンプルデータ](#) (last modified 2019/09/06)
- [イントロ | 一般 | ランダムに行を抽出](#) (last modified 2014/07/17)
- [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)
- [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 2019/04/24)
- [イントロ | 一般 | ランダムな塩基配列を生成](#) ① (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 2015/02/19)
- [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 | について](#) (last modified 2019/09/06)
- [イントロ | 一般 | 指定した範囲の配列を取得 | Biostring](#) (last modified 2019/09/06)
- [イントロ | 一般 | 指定したID\(染色体やdescription\)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 | について](#) (last modified 2019/09/06)
- [イントロ | 一般 | 翻訳配列\(translate\)を取得 | Biostrings](#) (last modified 2015/09/12)
- [イントロ | 一般 | 翻訳配列\(translate\)を取得 | seqinr\(Charif\\_2005\)](#) (last modified 2015/03/09) [トップページへ](#)
- [イントロ | 一般 | 相補鎖\(complement\)を取得](#) (last modified 2019/03/10)

# ランダム配列を生成2

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#intro\_general\_randomseq

ゲスト

## イントロ | 一般 | ランダムな塩基配列を生成

タイトル通り、「任意の長さ」で「任意の塩基組成」からなるランダムな塩基配列を生成するやり方を示します。A,C,G,Tの数値を指定することで任意の塩基組成にできるようになっています。指定する数値の合計は別に100にならなくてもかまいません。例えば「全てを1にしておけば、四種類の塩基の出現確率の期待値が25%」になりますし、「A=0, C=705, G=89, T=206みたいな指定法だと、（数値の合計が1000なので）塩基Cの出現確率が70.5%」みたこともできます。

### 1. 50塩基の長さのランダムな塩基配列を生成する場合：

塩基の存在比はAが20%, Cが30%, Gが30%, Tが20%にしています。

```
param_len_ref <- 50 #配列長を指定
narabi <- c("A","C","G","T") #以下の数値指定時にACGTの並びを間違えないようにするために表示(
param_composition <- c(20, 30, 30, 20) #(A,C,G,Tの並びで)各塩基の存在比率を指定

#本番(リファレンス配列生成)
ACGTset <- rep(narabi, param_composition)#narabi中の塩基がparam_compositionで指定した数だけ存在する
reference <- paste(sample(ACGTset, param_len_ref, replace=T), collapse="")#ACGTsetからparam_len_
reference #確認してるだけです
```

[トップページ](#)▲

①任意の存在比を与えて、任意の長さの塩基配列を生成する項目。②例題1。③例題4。

# ランダム配列を生成3

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#intro\_general\_randomseq

4. 配列長情報を含むファイル([seq\\_length.txt](#); 中身は「24, 103, 65, 49」という4行からなる数値情報)を読み込む場合 :

塩基の存在比はAが26%, Cが27%, Gが24%, Tが23%にしています。

```

in_f <- "seq_length.txt"           #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
narabi <- c("A","C","G","T")     #以下の数値指定時にACGTの並びを間違えないようにするために表示
param_composition <- c(26, 27, 24, 23) # (A,C,G,Tの並びで)各塩基の存在比率を指定
param_desc <- "contig"           #FASTA形式ファイルのdescription行に記述する内容

#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
param_len_ref <- readLines(in_f)  #in_fで指定したファイルの読み込み

#本番(配列生成)
ACGTset <- rep(narabi, param_composition) #narabi中の塩基がparam_compositionで指定した数だけ存在す
hoge <- NULL                       #hogeというプレースホルダの作成
for(i in 1:length(param_len_ref)){  #length(param_len_ref)で表現される配列数分だけループを回す
  hoge <- c(hoge, paste(sample(ACGTset, param_len_ref[i], replace=T), collapse="トッブページ"))
}

```

# ランダム配列を生成4

①任意の存在比を与えて、任意の長さの塩基配列を生成する項目。②例題1。③例題4。(A, C, G, Tの並びで)④で与えた各塩基の存在比率(GC含量だと51%)にして、⑤の長さをもつ計4配列からなるランダムな塩基配列を生成し、⑥hoge4.fastaというファイル名で保存するスクリプト。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#int

4. 配列長情報を含むファイル(seq\_length.txt; 中身は「24, 103, 65, 49」)の場合:

塩基の存在比はAが26%, Cが27%, Gが24%, Tが23%にしています。

```

in_f <- "seq_length.txt"
out_f <- "hoge4.fasta"
narabi <- c("A","C","G",")
param_composition <- c(26, 27, 24, 23)
param_desc <- "contig"

#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
param_len_ref <- readLines(in_f)

#本番(配列生成)
ACGTset <- rep(narabi, param_composition)
hoge <- NULL
for(i in 1:length(param_len_ref)){
  hoge <- c(hoge, paste(sample(ACGTset, param_len_ref[i], replace=T), collapse=""))
}
    
```

③

④

⑤

⑥

#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  
 #以下の数値指定時にACGTの並びを間違えないようにするために表示  
 #(A,C,G,Tの並びで)各塩基の存在比率を指定  
 #FASTA形式ファイルのdescription行に記述する内容

#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

#narabi中の塩基がparam\_compositionで指定した数だけ存在す  
 #hogeというプレースホルダの作成  
 #length(param\_len\_ref)で表現される配列数分だけループを回す  
 collapse="トップページ"

# ランダム配列を生成5

①任意の存在比を与えて、任意の長さの塩基配列を生成する項目。②例題1。③例題4。(A, C, G, Tの並びで)④で与えた各塩基の存在比率(GC含量だと51%)にして、⑤の長さをもつ計4配列からなるランダムな塩基配列を生成し、⑥hoge4.fastaというファイル名で保存するスクリプト。今回多くの入力ファイルとして使っているhoge4.faはこの項目の③例題4をコピー実行して得たものです。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#int

4. 配列長情報を含むファイル(seq\_length.txt; 中身は「24, 103, 65, 49」)を含む場合:

塩基の存在比はAが26%, Cが27%, Gが24%, Tが23%にしています。

```

in_f <- "seq_length.txt"
out_f <- "hoge4.fasta"
narabi <- c("A", "C", "G", "T")
param_composition <- c(26, 27, 24, 23)
param_desc <- "contig"

#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
param_len_ref <- readLines(in_f)

#本番(配列生成)
ACGTset <- rep(narabi, param_composition)
hoge <- NULL
for(i in 1:length(param_len_ref)){
  hoge <- c(hoge, paste(sample(ACGTset, param_len_ref[i], replace=T), collapse=""))
}
    
```

③ in\_f <- "seq\_length.txt"

④ param\_composition <- c(26, 27, 24, 23)

⑤ param\_len\_ref <- readLines(in\_f)

⑥ hoge <- c(hoge, paste(sample(ACGTset, param\_len\_ref[i], replace=T), collapse=""))

#入力ファイル名を指定してin\_fに指定したファイルの読み込み

#出力ファイル名を指定してout\_fに指定したファイルの書き込み

#以下の数値指定時にACGTの並びを間違えないようにするために表示(A, C, G, Tの並びで)各塩基の存在比率を指定

#FASTA形式ファイルのdescription行に記述する内容

#パッケージの読み込み

#hogeというプレースホルダの作成

#length(param\_len\_ref)で表現される配列数分だけループを回す

トップページ

# ランダム配列を生成6

①任意の存在比を与えて、任意の長さの塩基配列を生成する項目。②例題1。③例題4。(A, C, G, Tの並びで)④で与えた各塩基の存在比率(GC含量だと51%)にして、⑤の長さをもつ計4配列からなるランダムな塩基配列を生成し、⑥hoge4.fastaというファイル名で保存するスクリプト。今回多くの入力ファイルとして使っているhoge4.faはこの項目の③例題4をコピー実行して得たものです。この例題では常に同じ乱数を発生させるような操作を行っていない(set.seedという関数を利用していない)ため、⑥の出力ファイルとhoge4.faの塩基組成は異なりますが、⑤配列数と配列長は同じです。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#int

4. 配列長情報を含むファイル(seq\_length.txt; 中身は「24, 103, 65, 49」)を含む場合:

塩基の存在比はAが26%, Cが27%, Gが24%, Tが23%にしています。

```
in_f <- "seq_length.txt"
out_f <- "hoge4.fasta"
narabi <- c("A", "C", "G", "T")
param_composition <- c(26, 27, 24, 23)
param_desc <- "contig"
```

#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  
param\_len\_ref <- readLines(in\_f)

#本番(配列生成)

```
ACGTset <- rep(narabi, param_composition)
hoge <- NULL
for(i in 1:length(param_len_ref)){
  hoge <- c(hoge, paste(sample(ACGTset, param_len_ref[i], replace=T), collapse=""))
}
```

#入力ファイル名を指定して  
#出力ファイル名を指定して  
#以下の数値指定時にACGTの並びで各塩基の存在比を指定  
#FASTA形式ファイルのdesc

#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

#narabi中の塩基がparam\_compositionで指定した数だけ存在する  
#hogeというプレースホルダの作成  
#length(param\_len\_ref)で表現される配列数分だけループを回す  
#collapse=" "で空白文字を削除

①の項目の、②例題4を実行して、ある試行でたまたま得た結果ファイルが `hoge4.fa` ということ。それが、例えば…

# ランダム配列を生成7

(Rで)塩基配列解析

保護されていない通信 | [iu.a.u-tokyo.ac.jp/~kadota/r\\_seq.html#intro\\_general\\_randomseq](http://iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#intro_general_randomseq)

4. 配列長情報を含むファイル(`seq_length.txt`; 中身は「24, 103, 65, 49」という4行からなる数値情報)を読み込む場合:

塩基の存在比はAが26%, Cが27%, Gが24%, Tが23%にしています。

```

in_f <- "seq_length.txt"           #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
narabi <- c("A","C","G","T")     #以下の数値指定時にACGTの並びを間違えないようにするために表示
param_composition <- c(26, 27, 24, 23) # (A,C,G,Tの並びで)各塩基の存在比率を指定
param_desc <- "contig"           #FASTA形式ファイルのdescription行に記述する内容

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

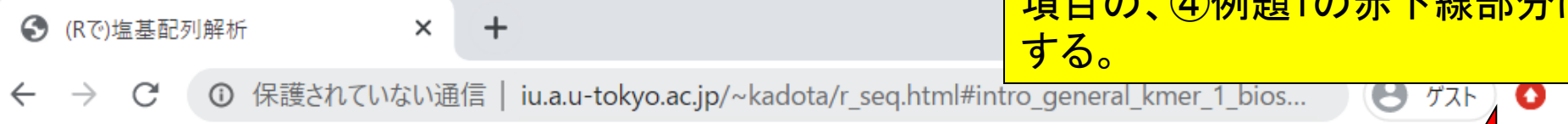
#入力ファイルの読み込み
param_len_ref <- readLines(in_f)  #in_fで指定したファイルの読み込み

#本番(配列生成)
ACGTset <- rep(narabi, param_composition) #narabi中の塩基がparam_compositionで指定した数だけ存在す
hoge <- NULL                       #hogeというプレースホルダの作成
for(i in 1:length(param_len_ref)){  #length(param_len_ref)で表現される配列数分だけループを回す
  hoge <- c(hoge, paste(sample(ACGTset, param_len_ref[i], replace=T), collapse="トッブページ"))
}
    
```



# ランダム配列を生成8

①の項目の、②例題4を実行して、ある試行でたまたま得た結果ファイルが hoge4.fa ということ。それが、例えば③の項目の、④例題1の赤下線部分に対応する。



## イントロ | 一般 | k-mer解析 | k=1(塩基ごとの出現頻度解析) | Biostrings

Biostringsパッケージを用いて、multi-FASTA形式ファイルを読み込んで、"A", "C", "G", "T", ..., "N", ...など塩基ごとの出現頻度を調べるやり方を示します。k-mer解析のk=1の場合に相当します。

「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ー。

### 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:

配列ごとに出現頻度をカウントした結果を返すやり方です。

```

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

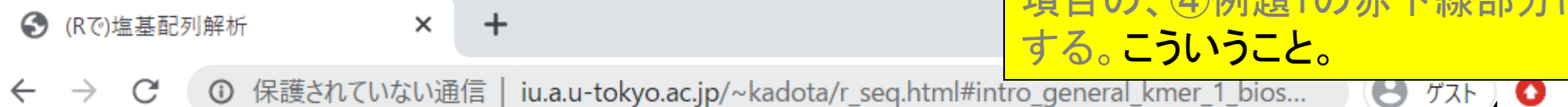
#入力ファイルの読み込み
fasta <- readDNAStringSet(in_f, format="fasta")#in_fで指定したファイルの読み込み

#本番
out <- alphabetFrequency(fasta) #A,C,G,T,..の数を各配列ごとにカウントした結果をトップページへ
out                                #outの中身を表示

```

# ランダム配列を生成9

①の項目の、②例題4を実行して、ある試行でたまたま得た結果ファイルが hoge4.fa ということ。それが、例えば③の項目の、④例題1の赤下線部分に対応する。ということ。



## イントロ | 一般 | k-mer解析 | k=1(塩基ごとの出現頻度解析) | Biostrings

Biostringsパッケージを用いて、multi-FASTA形式ファイルを読み込んで、"A", "C", "G", "T", ..., "N", ...など塩基ごとの出現頻度を調べるやり方を示します。k-mer解析のk=1の場合に相当します。

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

### 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:

配列ごとに出現頻度をカウントした結果を返すやり方です。

```

in_f <- "hoge4.fa"           #入力ファイル名を指定してin_fに格納
out_f <- "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
out <- alphabetFrequency(fasta) #A,C,G,T,..の数を各配列ごとにカウントした結果をトップページへ
out                                #outの中身を表示

```

# Contents

- Introduction、出現頻度解析(k=2)、出現頻度解析(k=1)
- k=1で実践、multi-FASTAファイル、他の例題を実行
- k=2で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8(k=2)、例題9(k=3)、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11(k=10)、例題12(k=10)、シークエンスエラーを含む場合

さきほどまでの①で、任意の塩基組成からなる、任意の長さの仮想ゲノム配列を作成可能になった。次は、②で仮想ゲノム配列から、任意の範囲の部分配列を切り出すやり方を示します。②をクリック。

# 部分配列の切り出し1

(Rで)塩基配列解析

x

+

← → ↻ ⓘ 保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#

ゲスト

- [基本的な利用法](#) (last modified 2020/03/06) **NEW**
- [サンプルデータ](#) (last modified 2019/09/06)
- イントロ | 一般 | [ランダムに行を抽出](#) (last modified 2014/07/17)
- 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)
- 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 2019/04/24)
- イントロ | 一般 | [ランダムな塩基配列を生成](#) ① (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 2015/02/19)
- 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 | について](#) (last modified 2019/09/06)
- イントロ | 一般 | 指定した範囲の配列を取得 | [Biostring](#) ② (last modified 2019/09/06)
- イントロ | 一般 | [指定したID\(染色体やdescription\)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 | について](#) (last modified 2019/09/06)
- イントロ | 一般 | 翻訳配列(translate)を取得 | [Biostrings](#) (last modified 2015/09/12)
- イントロ | 一般 | 翻訳配列(translate)を取得 | [seqinr\(Charif\\_2005\)](#) (last modified 2015/03/09)
- イントロ | 一般 | [相補鎖\(complement\)を取得](#) (last modified 2019/03/10)
- イントロ | 一般 | [逆相補鎖\(reverse complement\)を取得](#) (last modified 2019/03/10)

[トップページ](#) ⤴

# 部分配列の切り出し2

さきほどまでの①で、任意の塩基組成からなる、任意の長さの仮想ゲノム配列を作成可能になった。次は、②で仮想ゲノム配列から、任意の範囲の部分配列を切り出すやり方を示します。②をクリック。こんな感じになります。③例題1の、④入力ファイルの、⑤中身。

(Rで)塩基配列解析

← → ↻ ⓘ 保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#int

## イントロ | 一般 | 指定した範囲の配列を取得 | Biostrings

**Biostrings**パッケージ中のsubseq関数を用いて、single-FASTA形式やmulti-FASTA形式ファイルから様々な部分配列を取得するやり方を示します。この項目は、「この染色体の、ここから、ここまで」という指定の仕方になります。例えば入力ファイルがヒトゲノムだった場合に、chr3の20000から500000の座標の配列取得を行いたい場合などに利用します。したがって、chr4とchr8の配列のみ抽出といったやり方には対応していませんのでご注意ください。また、ファイルダウンロード時に、\*.fastaという拡張子が\*.txtに勝手に変更されることがありますのでご注意ください。ここでは、

「**③**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き、**⑤** `>kadota↓` ペ。

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

任意の範囲 (始点が3, 終点が9)の配列を抽出するやり方です。

```
in_f <- "sample1.fasta"
out_f <- "hoge1.fasta"
param <- c(3, 9)
```

#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  
#抽出したい範囲の始点と終点を指定

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

#パッケージの読み込み

```
#入力ファイルの読み込み
```

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

```
>kadota↓
AGTGACGGTCTT
```

[トップページ](#)△

# 部分配列の切り出し3

さきほどまでの①で、任意の塩基組成からなる、任意の長さの仮想ゲノム配列を作成可能になった。次は、②で仮想ゲノム配列から、任意の範囲の部分配列を切り出すやり方を示します。②をクリック。こんな感じになります。③例題1の、④入力ファイルの、⑤中身。⑥で指定した3-9番目の、⑦部分塩基配列を切り出した結果が…

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#int

## イントロ | 一般 | 指定した範囲の配列を取得 | Biostrings

**Biostrings**パッケージ中のsubseq関数を用いて、single-FASTA形式やmulti-FASTA形式を取得するやり方を示します。この項目は、「この染色体の、ここから、ここまで」の範囲で配列を取得するやり方を示します。例えば入力ファイルがヒトゲノムだった場合に、chr3の20000から500000の座標の配列取得を行いたい場合などに利用します。したがって、chr4とchr8の配列のみ抽出といったやり方には対応していませんのでご注意ください。また、ファイルダウンロード時に、\*.fastaという拡張子が\*.txtに勝手に変更されることがありますのでご注意ください。ここでは、

「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き

```
>kadota↓
AGTGACGGTCCT
```

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

任意の範囲 (始点が3, 終点が9)の配列を抽出するやり方です。

```
in_f <- "sample1.fasta"
out_f <- "hoge1.fasta"
param <- c(3, 9)
```

#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  
#抽出したい範囲の始点と終点を指定

#必要なパッケージをロード  
library(Biostrings)

#パッケージの読み込み

#入力ファイルの読み込み

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

[トップページ](#)△

# 部分配列の切り出し4

さきほどまでの①で、任意の塩基組成からなる、任意の長さの仮想ゲノム配列を作成可能になった。次は、②で仮想ゲノム配列から、任意の範囲の部分配列を切り出すやり方を示します。②をクリック。こんな感じになります。③例題1の、④入力ファイルの、⑤中身。⑥で指定した3-9番目の、⑦部分塩基配列を切り出した結果が、⑧の出力ファイル(hoge1.fasta)。

(Rで)塩基配列解析

x

+

← → ↻ ⓘ 保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#int

## イントロ | 一般 | 指定した範囲の配列を取得 | Biostrings

**Biostrings**パッケージ中のsubseq関数を用いて、single-FASTA形式やmulti-FASTA形式を取得するやり方を示します。この項目は、「この染色体の、ここから、ここまで」の範囲で配列を取得するやり方を示します。例えば入力ファイルがヒトゲノムだった場合に、chr3の20000から500000の座標の配列取得を行いたい場合などに利用します。したがって、chr4とchr8の配列のみ抽出といったやり方には対応していませんのでご注意ください。また、ファイルダウンロード時に、\*.fastaという拡張子が\*.txtに勝手に変更されることがありますのでご注意ください。ここでは、

「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ください。

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

任意の範囲 (始点が3, 終点が9)の配列を抽出するやり方です。

```
in_f <- "sample1.fasta"
out_f <- "hoge1.fasta"
param <- c(3, 9)
```

#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

#抽出したい範囲の始点と終点を指定してparamに格納

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

#パッケージの読み込み

```
#入力ファイルの読み込み
```

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

```
>kadota↓
```

```
AGTGACGGTCTT
```

```
>kadota
```

```
TGACGGT
```

[トップページ](#)△

# 部分配列の切り出し5

さきほどまでの①で、任意の塩基組成からなる、任意の長さの仮想ゲノム配列を作成可能になった。次は、②で仮想ゲノム配列から、任意の範囲の部分配列を切り出すやり方を示します。②をクリック。こんな感じになります。③例題1の、④入力ファイルの、⑤中身。⑥で指定した3-9番目の、⑦部分塩基配列を切り出した結果が、⑧の出力ファイル(hoge1.fasta)。RStudioの実行結果画面でも分かります。

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
Console Terminal x Jobs x
C:/Users/kadota/Desktop/hoge/
> 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
  A DNAStringSet instance of length 1
  width seq                                names
[1]    12 AGTGACGGTCTT                    kadota
>
> #本番
> fasta <- subseq(fasta, param[1], param[2])#paramで指定した
終点の範囲の配列を抽出した結果をfastaに格納
> fasta                                     #確認してるだけです
  A DNAStringSet instance of length 1
  width seq                                names
[1]     7 TGACGGT                          kadota
>
> #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イル名で保存
>
> |
```

```
>kadota↓
AGTGACGGTCTT
```

```
>kadota
TGACGGT
```

File Name	Size
.Rhistory	14.8 KB
hoge1.txt	221 B
hoge10.png	6.2 KB
hoge10.txt	142.1 KB
hoge11.png	6.3 KB
hoge11.txt	142.1 KB
hoge12.png	5.9 KB
hoge12.txt	138.8 KB



①例題4が、②よく知っている入力ファイル(hoge4.fa)を使っており、③任意の配列の任意の領域情報を含むリストファイル(list\_sub2.txt)を与えて実行する、より実践的なスクリプト。

# 部分配列の切り出し6

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#intro\_general\_subseq\_biost... ゲスト

## 4. [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:

目的のaccession番号が複数ある場合に対応したものです。予め用意しておいた「1列目: accession, 2列目: start位置, 3列目: end位置」からなるリストファイル(list\_sub2.txt)を読み込ませて、目的の配列のmulti-FASTAファイルをゲットするやり方です。

```

in_f1 <- "hoge4.fa"
in_f2 <- "list_sub2.txt"
out_f <- "hoge4.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_f1, format="fasta")#in_f1で指定したファイルの読み込み
posi <- read.table(in_f2)
fasta

#本番
hoge <- NULL
for(i in 1:nrow(posi)){
  obj <- names(fasta) == posi[i,1]
  hoge <- append(hoge, subseq(fasta[obj], start=posi[i,2], end=posi[i,3]))#subseq関数を用いて
}

```

#入力ファイル名を指定してin\_f1に格納(multi-FASTAファイル)  
 #入力ファイル名を指定してin\_f2に格納(リストファイル)  
 #出力ファイル名を指定してout\_fに格納

#パッケージの読み込み

#in\_f2で指定したファイルの読み込み  
 #確認してるだけです

#最終的に得る結果を格納するためのプレースホルダhogeを作成して  
 #length(posi)回だけループを回す  
 #条件を満たすかどうかを判定した結果をobjに格納 [トップページ](#)  
 #subseq関数を用いて

# Contents

- Introduction、出現頻度解析(k=2)、出現頻度
- k=1で実践、multi-FASTAファイル、他の例
- k=2で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8(k=2)、例題9(k=3)、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11(k=10)、例題12(k=10)、シークエンスエラーを含む場合

K-mer解析の応用編で、NGSデータのみを入力として、ゲノムサイズ推定までできちゃうんですよ、という話です。基本的な考え方をRStudio上でシミュレーションデータで示していく内容で、「おお！確かにできそうだね！」と楽しんでもらうのが目的です。レポートとは無関係部分です。

# サンプルデータ (例題32)

ここは見るだけ

(Rで)塩基配列解析 × +

← → ↻ ⓘ 保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#

- [基本的な利用法](#) (last modified 2020/03/06) NEW
- [サンプルデータ](#) (last modified 2019/09/06)
- イントロ | 一般 | [ランダムに行を抽出](#) (last modified 2014/07/17)
- 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)
- 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 2019/04/24)
- 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 2015/02/19)
- 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 | について](#) (last modified 2019/09/06)
- イントロ | 一般 | 指定した範囲の配列を取得 | [Biostring](#) (last modified 2019/09/06)
- イントロ | 一般 | [指定したID\(染色体やdescription\)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 | について](#) (last modified 2019/09/06)
- イントロ | 一般 | 翻訳配列(translate)を取得 | [Biostrings](#) (last modified 2015/09/12)
- イントロ | 一般 | 翻訳配列(translate)を取得 | [seqinr\(Charif\\_2005\)](#) (last modified 2015/03/09)
- イントロ | 一般 | [相補鎖\(complement\)を取得](#) (last modified 2019/03/10) [トップページへ](#)
- イントロ | 一般 | [逆相補鎖\(reverse complement\)を取得](#) (last modified 2019/03/10)

# サンプルデータ(例題32)

①サンプルデータの、②例題32は、塩基配列解析の基本テクニックのみを用いて、③仮想ゲノムと仮想NGSリードのファイルを作成するスクリプトです。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#sample\_data

32. k-mer解析用のランダム配列から生成したFASTA形式ファイル([sample32\\_ref.fasta](#)と[sample32\\_ngs.fasta](#))です。

50塩基の長さのリファレンス配列を生成したのち、20塩基長の部分配列を10リード分だけランダム抽出したものです。塩基の存在比はAが22%, Cが28%, Gが28%, Tが22%にしています。リファレンス配列(仮想ゲノム配列)が[sample32\\_ref.fasta](#)で、10リードからなる仮想NGSデータが[sample32\\_ngs.fasta](#)です。リード長20塩基で10リードなのでトータル200塩基となり、50塩基からなる元のゲノム配列の4倍シーケンスしていることとなります(4X coverageに相当)。[イントロ](#) | [NGS](#) | [配列取得](#) | [シミュレーションデータ](#) | [ランダムな塩基配列の生成から](#)と基本的に同じです。

```

out_f1 <- "sample32_ref.fasta" }
out_f2 <- "sample32_ngs.fasta" }
param_len_ref <- 50
narabi <- c("A","C","G","T")
param_composition <- c(22, 28, 28, 22)
param_len_ngs <- 20
param_num_ngs <- 10
param_desc <- "kkk"

#必要なパッケージをロード
library(Biostrings)

#本番(リファレンス配列生成)
set.seed(1010)
    
```

③

#出力ファイル名を指定してout\_f1に格納  
 #出力ファイル名を指定してout\_f2に格納  
 #リファレンス配列の長さを指定  
 #以下の数値指定時にACGTの並びを間違えないようにするために  
 #(A,C,G,Tの並びで)各塩基の存在比率を指定  
 #リード長を指定  
 #リード数を指定  
 #FASTA形式ファイルのdescription行に記述する内容

#パッケージの読み込み

#おまじない(同じ乱数になるようにするため)

[トップページへ](#)

# サンプルデータ(例題32)

①サンプルデータの、②例題32は、塩基配列解析の基本テクニックのみを用いて、③仮想ゲノムと仮想NGSリードのファイルを作成するスクリプトです。④で示すA, C, G, Tの存在比率で、⑤50塩基の長さの、⑥仮想ゲノム(sample32\_ref.fasta)を、まず⑦の部分で作成しています。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#sa

32. k-mer解析用のランダム配列から生成したFASTA形式ファイル(sample32\_ref.fasta)です。

50塩基の長さのリファレンス配列を生成したのち、20塩基長の部分配列を10リード分だけランダム抽出したものです。塩基の存在比はAが22%, Cが28%, Gが28%, Tが22%にしています。リファレンス配列(仮想ゲノム配列)がsample32\_ref.fastaで、10リードからなる仮想NGSデータがsample32\_ngs.fastaです。リード長20塩基で10リードなのでトータル200塩基となり、50塩基からなる元のゲノム配列の4倍シーケンスしていることとなります(4X coverageに相当)。 [イントロ](#) | [NGS](#) | [配列取得](#) | [シミュレーションデータ](#) | [ランダムな塩基配列の生成から](#)と基本的に同じです。

```

out_f1 <- "sample32_ref.fasta"
out_f2 <- "sample32_ngs.fasta"
param_len_ref <- 50
narabi <- c("A", "C", "G", "T")
param_composition <- c(22, 28, 28, 22)
param_len_ngs <- 20
param_num_ngs <- 10
param_desc <- "kkk"

#必要なパッケージをロード
library(Biostrings)

#本番(リファレンス配列生成)
set.seed(1010)
    
```

⑤

⑥

④

⑦

```

#出力ファイル名を指定してout_f1に格納
#出力ファイル名を指定してout_f2に格納
#リファレンス配列の長さを指定
#以下の数値指定時にACGTの並びを間違えないようにするために
#A, C, G, Tの並びで)各塩基の存在比率を指定
#リード長を指定
#リード数を指定
#FASTA形式ファイルのdescription行に記述する内容

#パッケージの読み込み

#おまじない(同じ乱数になるようにするため)
    
```

[トップページへ](#)

# サンプルデータ(例題32)

(Rで)塩基配列解析



```
>kkk
GAAGTTGTCTTTTGAACCTCACTACACCAGGACATGAAGACGCGGACGGCA
```

す。  
50塩基の長さのリファレンス配列を生成したのち、20塩基長の部分配列  
のです。塩基の存在比はAが22%, Cが28%, Gが28%, Tが22%にしています。リファレンス配列(仮想ゲノム  
配列)が[sample32\\_ref.fasta](#)で、10リードからなる仮想NGSデータが[sample32\\_ngs.fasta](#)です。リード長20  
塩基で10リードなのでトータル200塩基となり、50塩基からなる元のゲノム配列の4倍シーケンスしていること  
になります(4X coverageに相当)。[イントロ | NGS | 配列取得 | シミュレーションデータ | ランダムな塩基配  
列の生成から](#)と基本的に同じです。

①サンプルデータの、②例題32は、塩基配列解析の基本テクニックのみを用いて、③仮想ゲノムと仮想NGSリードのファイルを作成するスクリプトです。④で示すA, C, G, Tの存在比率で、⑤50塩基の長さの、⑥仮想ゲノム(sample32\_ref.fasta)を、まず⑦の部分で作成しています。⑥の中身が、⑧。

```

out_f1 <- "sample32_ref.fasta"
out_f2 <- "sample32_ngs.fasta"
param_len_ref <- 50
narabi <- c("A", "C", "G", "T")
param_composition <- c(22, 28, 28, 22)
param_len_ngs <- 20
param_num_ngs <- 10
param_desc <- "kkk"

#必要なパッケージをロード
library(Biostrings)

#本番(リファレンス配列生成)
set.seed(1010)
    
```

⑤

⑥

④

⑦

```

#出力ファイル名を指定してout_f1に格納
#出力ファイル名を指定してout_f2に格納
#リファレンス配列の長さを指定
#以下の数値指定時にACGTの並びを間違えないようにするために
#(A, C, G, Tの並びで)各塩基の存在比率を指定
#リード長を指定
#リード数を指定
#FASTA形式ファイルのdescription行に記述する内容

#パッケージの読み込み

#おまじない(同じ乱数になるようにするため)
    
```

[トップページへ](#)

# サンプルデータ (例題32)

①サンプルデータの、②例題32は、塩基配列解析の基本テクニックのみを用いて、③仮想ゲノムと仮想NGSリードのファイルを作成するスクリプトです。④で示すA, C, G, Tの存在比率で、⑤50塩基の長さの、⑥仮想ゲノム(sample32\_ref.fasta)を、まず⑦の部分で作成しています。⑥の中身が、⑧。次に、仮想ゲノム配列をリファレンスとして、⑨20塩基長の部分配列(リード)を、⑩10個分ランダム抽出した、⑪仮想リード(sample32\_ngs.fasta)。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#sa

32. k-mer解析用のランダム配列から生成したFASTA形式ファイル(sample32\_ref.fasta)です。

50塩基の長さのリファレンス配列を生成したのち、20塩基長の部分配列(リード)がsample32\_ref.fastaで、10リードからなる仮想NGSデータがsample32\_ngs.fastaで、10リードからなる仮想NGSデータがsample32\_ngs.fastaで10リードなのでトータル200塩基となり、50塩基からなる元のゲノムになります(4X coverageに相当)。イントロ | NGS | 配列取得 | シミュレーションの生成からと基本的に同じです。

```
out_f1 <- "sample32_ref.fasta"
out_f2 <- "sample32_ngs.fasta"
param_len_ref <- 50
narabi <- c("A", "C", "G", "T")
param_composition <- c(22, 28, 28, 22)
param_len_ngs <- 20
param_num_ngs <- 10
param_desc <- "kkk"
```

#出力ファイル名を指定してout\_f1に格納  
#出力ファイル名を指定してout\_f2に格納  
#リファレンス配列の長さを指定  
#以下の数値指定時にACGTの並びを間違えないようにするために  
#(A, C, G, Tの並びで)各塩基の存在比率を指定  
#リード長を指定  
#リード数を指定  
#FASTA形式ファイルのdescription行に記述する内容

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

#パッケージの読み込み

```
#本番(リファレンス配列生成)
set.seed(1010)
```

#おまじない(同じ乱数になるようにするため)

[トップページへ](#)

# サンプルデータ(例題32)

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#sa

32. k-mer解析用のランダム配列から生成したFASTA形式ファイル(sample32\_ref.fasta)です。

50塩基の長さのリファレンス配列を生成したのち、20塩基長の部分配列の配列)がsample32\_ref.fastaで、10リードからなる仮想NGSデータがsample32\_ngo.fastaで、10塩基で10リードなのでトータル200塩基となり、50塩基からなる元のゲノム配列になります(4X coverageに相当)。イントロ | NGS | 配列取得 | シミュレーションの生成からと基本的に同じです。

```

out_f1 <- "sample32_ref.fasta" #出力ファイル名を指定
out_f2 <- "sample32_ngo.fasta" #出力ファイル名を指定
param_len_ref <- 50 #リファレンス配列の長さ
narabi <- c("A","C","G","T") #以下の数値指定時にACGTの並びで各塩基の割合を指定
param_composition <- c(22, 28, 28, 22) # (A,C,G,Tの並びで)各塩基の割合を指定
param_len_ngo <- 20 #リード長を指定
param_num_ngo <- 10 #リード数を指定
param_desc <- "kkk" #FASTA形式ファイルのdescription
    
```

```

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

```

#パッケージの読み込み
    
```

```

#本番(リファレンス配列生成)
set.seed(1010)
    
```

```

#おまじない(同じ乱数になるようにするため)
    
```



[トップページへ](#)

①サンプルデータの、②例題32は、塩基配列解析の基本テクニックのみを用いて、③仮想ゲノムと仮想NGSリードのファイルを作成するスクリプトです。④で示すA, C, G, Tの存在比率で、⑤50塩基の長さの、⑥仮想ゲノム(sample32\_ref.fasta)を、まず⑦の部分で作成しています。⑥の中身が、⑧。次に、仮想ゲノム配列をリファレンスとして、⑨20塩基長の部分配列(リード)を、⑩10個分ランダム抽出した、⑪仮想リード(sample32\_ngo.fasta)。⑫毎回同じ乱数が生成されるようにしているので、おそらく同じ結果が得られる。ただし、R本体のバージョンアップ後に再度実行すると結果が異なります。おそらくですが、R本体のバージョンが同じ場合のみ有効なのだろうと思います。



# サンプルデータ(例題32)

⑨20塩基長の部分配列(リード)を、⑩10個分ランダム抽出した、⑪仮想リード(sample32\_ngs.fasta)の中身。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#sample\_data

32. k-mer解析用のランダム配列から生成したFASTA形式ファイル(sample32\_ref.fastaとsample32\_ngs.fasta)です。

50塩基の長さのリファレンス配列を生成したものです。塩基の存在比はAが22%, Cが28%配列)がsample32\_ref.fastaで、10リード塩基で10リードなのでトータル200塩基とになります(4X coverageに相当)。イントロの生成からと基本的に同じです。

```
out_f1 <- "sample32_ref.fasta"
out_f2 <- "sample32_ngs.fasta"
param_len_ref <- 50
narabi <- c("A", "C", "G", "T")
param_composition <- c(22, 28, 28, 22)
param_len_ngs <- 20
param_num_ngs <- 10
param_desc <- "kkk"
```

```
#必要なパッケージをロード
library(Biostrings)

#本番(リファレンス配列生成)
set.seed(1010)
```

```
>kkk_24_43
CACCAGGACATGAAGACGCG
>kkk_28_47
AGGACATGAAGACGCGGACG
>kkk_12_31
TTGAACTCACTACACCAGGA
>kkk_4_23
GTTGTCTTTTGAAGTCACTA
>kkk_5_24
TTGTCTTTTGAAGTCACTAC
>kkk_7_26
GTCTTTTGAAGTCACTACAC
>kkk_12_31
TTGAACTCACTACACCAGGA
>kkk_23_42
ACACCAGGACATGAAGACGCG
>kkk_1_20
GAAGTTGTCTTTTGAAGTCA
>kkk_3_22
AGTTGTCTTTTGAAGTCACT
```

10リード分だけランダム抽出したものです。リファレンス配列(仮想ゲノムsample32\_ref.fasta)とランダム抽出した配列の4倍シーケンスしていること(4X coverage)をシミュレーションデータ|ランダムな塩基配列

out\_f1に格納  
out\_f2に格納  
指定  
並びを間違えないようにするために  
の存在比率を指定

description行に記述する内容

[トップページへ](#)

るようにするため)

# サンプルデータ(例題32)

①仮想リード中の、②description部分を見れば、③そのリードが、④仮想ゲノム配列中のどの領域由来かがわかる。

(Rで)塩基配列解析

```
>kkk
GAAGTTGTCTTTTGAAGTCACTACACCAGGACATGAAGACGCGGACGGCA
```

50塩基の長さのリファレンス配列を生成し、塩基の存在比はAが22%、Cが28%、Gが28%、Tが22%のランダムな配列(仮想ゲノム配列)がsample32\_ref.fastaで、10リード分だけランダム抽出したものをsample32\_ngs.fastaで、リード長20塩基で10リードなのでトータル200塩基と生成します(4X coverageに相当)。イントロの生成からと基本的に同じです。

```
out_f1 <- "sample32_ref.fasta"
out_f2 <- "sample32_ngs.fasta"
param_len_ref <- 50
narabi <- c("A","C","G","T")
param_composition <- c(22, 28, 28, 22)
param_len_ngs <- 20
param_num_ngs <- 10
param_desc <- "kkk"
```

```
#必要なパッケージをロード
library(Biostrings)

#本番(リファレンス配列生成)
set.seed(1010)
```

```
>kkk_24_49
CACCAGGACATGAAGACGCGGACGGCA
>kkk_28_47
AGGACATGAAGACGCGGACGGCA
>kkk_12_31
TTGAACTCACTACACCAGGACATGAAGACGCGGACGGCA
>kkk_4_23
GTTGTCTTTTGAAGTCACTACACCAGGACATGAAGACGCGGACGGCA
>kkk_5_24
TTGTCTTTTGAAGTCACTACACCAGGACATGAAGACGCGGACGGCA
>kkk_7_26
GTCTTTTGAAGTCACTACACCAGGACATGAAGACGCGGACGGCA
>kkk_12_31
TTGAACTCACTACACCAGGACATGAAGACGCGGACGGCA
>kkk_23_42
ACACCAGGACATGAAGACGCGGACGGCA
>kkk_1_20
GAAGTTGTCTTTTGAAGTCACTACACCAGGACATGAAGACGCGGACGGCA
>kkk_3_22
AGTTGTCTTTTGAAGTCACTACACCAGGACATGAAGACGCGGACGGCA
```

sample\_data (sample32\_ref.fastaとsample32\_ngs.fasta)で

0リード分だけランダム抽出したものをsample32\_ref.fastaで、10リード分だけランダム抽出したものをsample32\_ngs.fastaで、リード長20塩基で10リードなのでトータル200塩基と生成します(4X coverageに相当)。イントロの生成からと基本的に同じです。

out\_f1に格納指定、out\_f2に格納指定、並びを間違えないようにするためにの存在比率を指定

description行に記述する内容

[トップページへ](#)

るようにするため)

# サンプルデータ(例題32)

①仮想リード中の、②description部分を見れば、③そのリードが、④仮想ゲノム配列中のどの領域由来かがわかる。他の例。

(Rで塩基配列解析

x +

```
>kkk
GAAGTTGTCCTTTGAACTCACTACACCAGGACATGAAGACGCGGACGGCA
```

50塩基の長さのリファレンス配列を生成し、塩基の存在比はAが22%、Cが28%配列)がsample32\_ref.fastaで、10リード塩基で10リードなのでトータル200塩基とになります(4X coverageに相当)。イント

```
out_f1 <- "sample32_ref.fasta"
out_f2 <- "sample32_ngs.fasta"
param_len_ref <- 50
narabi <- c("A","C","G","T")
param_composition <- c(22, 28, 28,
param_len_ngs <- 20
param_num_ngs <- 10
param_desc <- "kkk"
```

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

```
#本番(リファレンス配列生成)
set.seed(1010)
```

```
>kkk_24_43
CACCAGGACATGAAGACGCG
>kkk_28_47
AGGACATGAAGACGCGGACG
>kkk_12_31
TTGAACTCACTACACCAGGA
>kkk_4_23
GTTGTCCTTTTGAAGTCACTA
>kkk_5_24
TTGTCCTTTTGAAGTCACTAC
>kkk_7_26
GTCTTTTGAAGTCACTACAC
>kkk_12_31
TTGAACTCACTACACCAGGA
>kkk_23_42
ACACCAGGACATGAAGACGC
>kkk_1_20
GAAGTTGTCCTTTGAACTCA
>kkk_3_22
AGTTGTCCTTTTGAAGTCACT
```

0リード分だけランダム抽出したもす。リファレンス配列(仮想ゲノムle32\_ngs.fasta)です。リード長20配列の4倍シーケンスしていること

out\_f1に格納
out\_f2に格納
指定
並びを間違えないようにするためにの存在比率を指定

ription行に記述する内容

トップページへ

るようにするため)

# Contents

- Introduction、出現頻度解析(k=2)、出現頻度解析(k=1)
- k=1で実践、multi-FASTAファイル、他の例題を実行
- k=2で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8(k=2)、例題9(k=3)、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11(k=10)、例題12(k=10)、シークエンスエラーを含む場合

# 被覆率 (coverage) 1

おさらい。①仮想ゲノムの配列長は、②50塩基。③仮想リードは、④10個で、⑤リード長は20塩基。従って、リード側の総塩基数は $10 \times 20 = 200$ 塩基。この場合、仮想ゲノムの $200 / 50 = 4$ 倍分塩基配列決定 (sequencing) がなされているので、被覆率 (coverage) は**4X coverage**みたいな表現をします。

(Rで)塩基配列解析

```
>kkk  
GAAGTTGTCTTTTGAACCTACTACACCAGGACATGAAGACGCGGACGGCA
```

す。  
50塩基の長さのリファレンス配列を生成し、塩基の存在比はAが22%, Cが28%配列)がsample32\_ref.fastaで、10リード塩基で10リードなのでトータル200塩基とになります(4X coverageに相当)。イント

```
>kkk_24_43  
CACCAGGACATGAAGACGCG  
>kkk_28_47  
AGGACATGAAGACGCGGACG  
>kkk_12_31  
TTGAACTCACTACACCAGGA  
>kkk_4_23  
GTTGTCTTTTGAACCTACTA  
>kkk_5_24  
TTGTCTTTTGAACCTACTAC  
>kkk_7_26  
GTCTTTTGAACCTACTACAC  
>kkk_12_31  
TTGAACTCACTACACCAGGA  
>kkk_23_42  
ACACCAGGACATGAAGACGC  
>kkk_1_20  
GAAGTTGTCTTTTGAACCTCA  
>kkk_3_22  
AGTTGTCTTTTGAACCTCACT
```

す。リファレンス配列(仮想ゲノム)がsample32\_ref.fasta、リード長20塩基の配列の4倍シーケンスしていること、ランダムな塩基配列の生成からと基本的に同じです。

out\_f1に格納  
out\_f2に格納  
指定  
並びを間違えないようにするために  
の存在比率を指定

ription行に記述する内容

[トップページへ](#)

るようにするため)

```
② out_f1 <- "sample32_ref.fasta"  
out_f2 <- "sample32_ngs.fasta"  
param_len_ref <- 50  
narabi <- c("A", "C", "G", "T")  
⑤ param_composition <- c(22, 28, 28, 20)  
param_len_ngs <- 20  
param_num_ngs <- 10  
④ param_desc <- "kkk"  
  
#必要なパッケージをロード  
library(Biostrings)
```

```
#本番(リファレンス配列生成)  
set.seed(1010)
```

# 被覆率 (coverage) 2

(Rで)塩基配列解析

x +

```
>kkk  
GAAGTTGTCTTTTGAAGTCACTACACCAGGACATGAAGACGCGGACGGCA
```

す。

50塩基の長さのリファレンス配列を生成し、その塩基の存在比はAが22%、Cが28%、Gが28%、Tが22%の配列)がsample32\_ref.fastaで、10リード長20塩基で10リードなのでトータル200塩基とになります(4X coverageに相当)。イントロの生成からと基本的に同じです。

```
>kkk_24_43  
CACCAGGACATGAAGACG  
>kkk_28_47  
AGGACATGAAGACGCGGA  
>kkk_12_31  
TTGAACTCACTACACCAG  
>kkk_4_23  
GTTGTCTTTTGAAGTCA  
>kkk_5_24  
TTGTCTTTTGAAGTCACTAC  
>kkk_7_26  
GTCTTTTGAAGTCACTAC  
>kkk_12_31  
TTGAACTCACTACACCAGGA  
>kkk_23_42  
ACACCAGGACATGAAGACGC  
>kkk_1_20  
GAAGTTGTCTTTTGAAGTCA  
>kkk_3_22  
AGTTGTCTTTTGAAGTCACT
```

```
out_f1 <- "sample32_ref.fasta"  
out_f2 <- "sample32_ngs.fasta"  
param_len_ref <- 50  
narabi <- c("A", "C", "G", "T")  
param_composition <- c(22, 28, 28, 22)  
param_len_ngs <- 20  
param_num_ngs <- 10  
param_desc <- "kkk"
```

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

```
#本番(リファレンス配列生成)  
set.seed(1010)
```

おさらい。①仮想ゲノムの配列長は、②50塩基。③仮想リードは、④10個で、⑤リード長は20塩基。従って、リード側の総塩基数は $10 \times 20 = 200$ 塩基。この場合、仮想ゲノムの $200 / 50 = 4$ 倍分塩基配列決定 (sequencing) がなされているので、被覆率 (coverage) は4X coverageみたいな表現をします。通常は、③NGSリードデータが手元にあり、*de novo genome assembly*によって①元のゲノム配列を再構築するという流れになる。この目的を達成するためには、リード間の「のりしろ」が必要になるので、例えば「うまくつなげるためには、③が50X coverage程度分必要」みたいな議論がなされる。

ription行に記述する内容

[トップページへ](#)

るようにするため)

# 被覆率 (coverage) 3

(Rで)塩基配列解析

x +

```
>kkk  
GAAGTTGTCTTTTGAAGTCACTACACCAGGACATGAAGACGCGGACGGCA
```

す。  
50塩基の長さのリファレンス配列を生成し、塩基の存在比はAが22%、Cが28%、Gが28%、Tが22%の配列)がsample32\_ref.fastaで、10リード長が20塩基で10リードなのでトータル200塩基とになります(4X coverageに相当)。イントロの生成からと基本的に同じです。

```
>kkk_24_43  
CACCAGGACATGAAGACG  
>kkk_28_47  
AGGACATGAAGACGCGGA  
>kkk_12_31  
TTGAACTCACTACACCAG  
>kkk_4_23  
GTTGTCTTTTGAAGTCA  
>kkk_5_24  
TTGTCTTTTGAAGTCACT  
>kkk_7_26  
GTCTTTTGAAGTCACTAC  
>kkk_12_31  
TTGAACTCACTACACCAG  
>kkk_23_42  
ACACCAGGACATGAAGAC  
>kkk_1_20  
GAAGTTGTCTTTTGAAGTCA  
>kkk_3_22  
AGTTGTCTTTTGAAGTCACT
```

```
out_f1 <- "sample32_ref.fasta"  
out_f2 <- "sample32_ngs.fasta"  
param_len_ref <- 50  
narabi <- c("A", "C", "G", "T")  
param_composition <- c(22, 28, 28, 22)  
param_len_ngs <- 20  
param_num_ngs <- 10  
param_desc <- "kkk"
```

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

```
#本番(リファレンス配列生成)  
set.seed(1010)
```

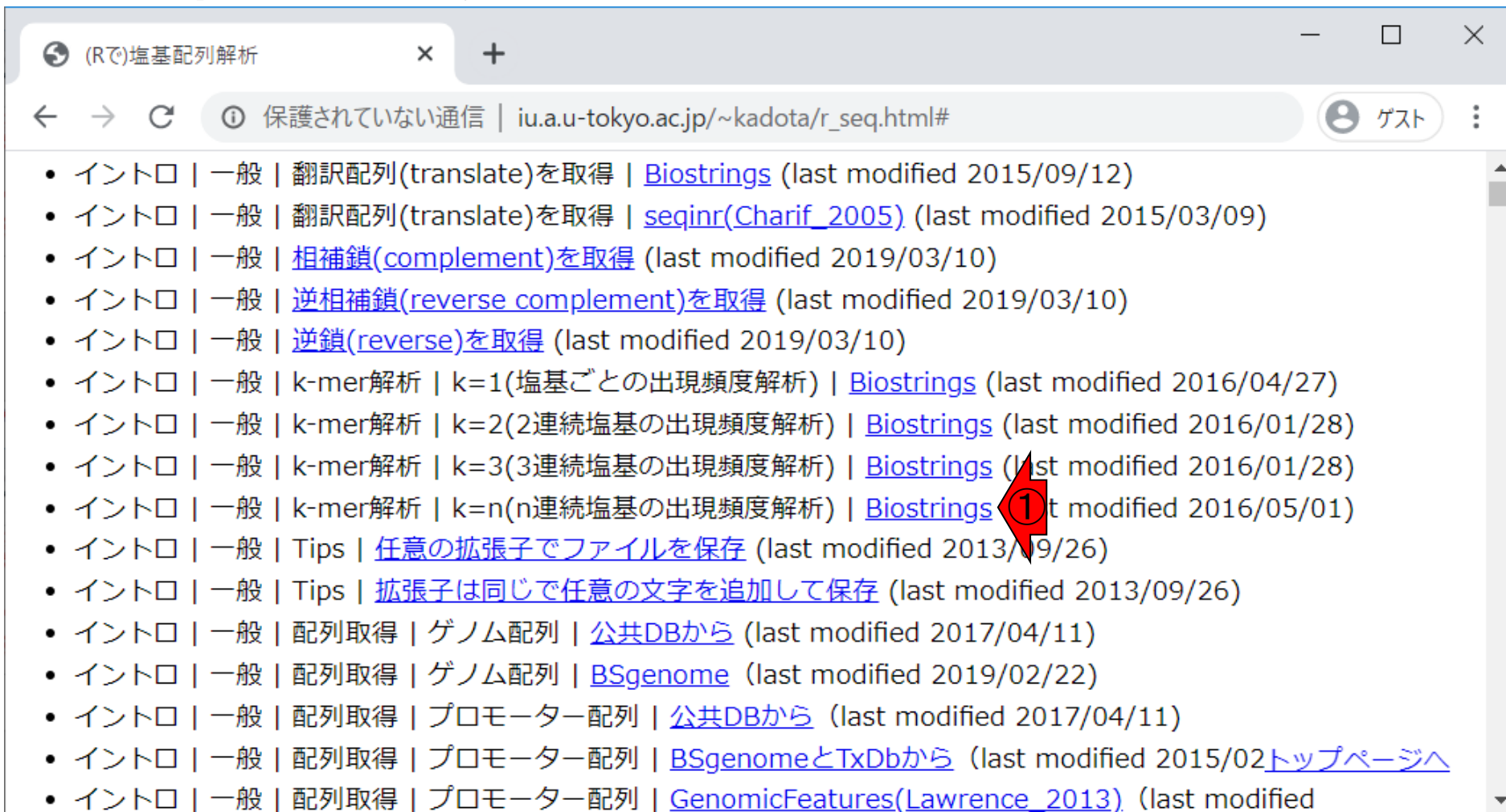
おさらい。①仮想ゲノムの配列長は、②50塩基。③仮想リードは、④10個で、⑤リード長は20塩基。従って、リード側の総塩基数は $10 \times 20 = 200$ 塩基。この場合、仮想ゲノムの $200 / 50 = 4$ 倍分塩基配列決定 (sequencing) がなされているので、被覆率 (coverage) は4X coverageみたいな表現をします。通常は、③NGSリードデータが手元にあり、*de novo genome assembly*によって①元のゲノム配列を再構築するという流れになる。この目的を達成するためには、リード間の「のりしろ」が必要になるので、例えば「うまくつなげるためには、③が50X coverage程度分必要」みたいな議論がなされる。そしてこの議論をするための前提条件として必要なのは再構築したいゲノムサイズ情報。そして③のリード情報を入力として行うk-mer解析によってゲノムサイズの推定が可能だということです。

# Contents

- Introduction、出現頻度解析( $k=2$ )、出現頻度解析( $k=1$ )
- $k=1$ で実践、multi-FASTAファイル、他の例題を実行
- $k=2$ で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8( $k=2$ )、例題9( $k=3$ )、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11( $k=10$ )、例題12( $k=10$ )、シークエンスエラーを含む場合



# 基本的な考え方(例題7)1



(Rで)塩基配列解析

保護されていない通信 | [iu.a.u-tokyo.ac.jp/~kadota/r\\_seq.html#](http://iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#)

- イントロ | 一般 | 翻訳配列(translate)を取得 | [Biostrings](#) (last modified 2015/09/12)
- イントロ | 一般 | 翻訳配列(translate)を取得 | [seqinr\(Charif\\_2005\)](#) (last modified 2015/03/09)
- イントロ | 一般 | [相補鎖\(complement\)](#)を取得 (last modified 2019/03/10)
- イントロ | 一般 | [逆相補鎖\(reverse complement\)](#)を取得 (last modified 2019/03/10)
- イントロ | 一般 | [逆鎖\(reverse\)](#)を取得 (last modified 2019/03/10)
- イントロ | 一般 | k-mer解析 | k=1(塩基ごとの出現頻度解析) | [Biostrings](#) (last modified 2016/04/27)
- イントロ | 一般 | k-mer解析 | k=2(2連続塩基の出現頻度解析) | [Biostrings](#) (last modified 2016/01/28)
- イントロ | 一般 | k-mer解析 | k=3(3連続塩基の出現頻度解析) | [Biostrings](#) (last modified 2016/01/28)
- イントロ | 一般 | k-mer解析 | k=n(n連続塩基の出現頻度解析) | [Biostrings](#) (last modified 2016/05/01)
- 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 2017/04/11)
- イントロ | 一般 | 配列取得 | ゲノム配列 | [BSgenome](#) (last modified 2019/02/22)
- 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 2017/04/11)
- イントロ | 一般 | 配列取得 | プロモーター配列 | [BSgenomeとTxDbから](#) (last modified 2015/02 [トップページへ](#))
- イントロ | 一般 | 配列取得 | プロモーター配列 | [GenomicFeatures\(Lawrence\\_2013\)](#) (last modified

# 基本的な考え方(例題7)2

(Rで)塩基配列解析

← ② [iu.a.u-tokyo.ac.jp/~kadota/r\\_seq.html#intro\\_general\\_kmer\\_n\\_bios...](http://iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#intro_general_kmer_n_bios...)

## 7. サンプルデータの例題32を実行して得られたmulti-FASTAファイル([sample32\\_ngs.fasta](#))の場合:

2連続塩基 (k=2) の出現頻度情報を得るやり方です。4<sup>2</sup> = 16通りのk-merの出現頻度を計算することになります。リード毎に出現頻度を算出しています。

```
in_f <- "sample32_ngs.fasta"      #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt"              #出力ファイル名を指定してout_fに格納
param_kmer <- 2                  #k-merのkの値を指定

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

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

#本番
out <- oligonucleotideFrequency(fasta, width=param_kmer)#k連続塩基の出現頻度情報をoutに格納

#ファイルに保存
tmp <- cbind(names(fasta), out)   #最初の列にID情報、そのあとに出現頻度情報のoutを結合したtmpを作成
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)#tmpの中身を指定したファイル名で保存
```

[トップページ](#)△

# 基本的な考え方(例題7)3

①「n連続塩基」のほうです。項目名に注意。②例題7。③入力ファイルの中身。これの元のゲノム配列は50塩基であり、③の総塩基数は10リード×20塩基 = 200塩基。ゲノムサイズの4倍の被覆率(4X coverage)であることが分かっている。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#int

## 7. サンプルデータの例題32を実行して得られたmulti-FASTAファイル(sample32\_ngs.fasta)の場合:

2連続塩基 (k=2) の出現頻度情報を得るやり方です。リード毎に出現頻度を算出しています。

```

in_f <- "sample32_ngs.fasta"
out_f <- "hoge7.txt"
param_kmer <- 2

#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")

#本番
out <- oligonucleotideFrequency(fasta, k=2)

#ファイルに保存
tmp <- cbind(names(fasta), out)
write.table(tmp, out_f, sep="\t", append=TRUE)

```

```

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

```

fに格納  
fに格納

ファイルの読み込み

の出現頻度情報をoutに格納

出現頻度情報のoutを結合したtmpを  
#tmpの中身を指定したファイル名で保

[トップページ](#)

# 基本的な考え方(例題7)4

①「n連続塩基」のほうです。項目名に注意。②例題7。③入力ファイルの中身。これの元のゲノム配列は50塩基であり、③の総塩基数は10リード×20塩基 = 200塩基。ゲノムサイズの4倍の被覆率(4X coverage)であることが分かっている。④で与えているのはk-mer解析のkの値。③入力ファイルをダウンロードして、コピー実行。

(Rで)塩基配列解析

保護されていない通信 | [iu.a.u-tokyo.ac.jp/~kadota/r\\_seq.html#int](http://iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#int)

## 7. サンプルデータの例題32を実行して得られたmulti-FASTAファイル(samp

2連続塩基 (k=2) の出現頻度情報を得るやり方です。4<sup>2</sup> = 16通りのk-merです。リード毎に出現頻度を算出しています。

```
in_f <- "sample32_ngs.fasta" #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.txt" #出力ファイル名を指定してout_fに格納
param_kmer <- 2 #k-merのkの値を指定

#必要なパッケージをダウンロード
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
out <- oligonucleotideFrequency(fasta, width=param_kmer) #k連続塩基の出現頻度情報をoutに格納

#ファイルに保存
tmp <- cbind(names(fasta), out) #最初の列にID情報、そのあとに出現頻度情報のoutを結合したtmpを作成
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F) #tmpの中身を指定したファイル名で保存
```

[トップページ](#)△

# 基本的な考え方(例題7)5

実行結果。出力ファイルの中身は、実質的に①outオブジェクトの中身と同じなので…

The screenshot shows the RStudio interface. The console on the left contains the following R code:

```
> library(Biostrings)
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
> out <- oligonucleotideFrequency(fasta, width=param_kmer)#k連続塩基の出現頻度情報をoutに格納
>
> #ファイルに保存
> tmp <- cbind(names(fasta), out) #最初の列にID情報、そのあとに出現頻度情報のoutを結合したtmpを作成
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)#tmpの中身を指定したファイル名で保存
>
> |
```

A red arrow with the number 1 points to the line `tmp <- cbind(names(fasta), out)`. The file explorer on the right shows the directory `C:\Users\kadota\Desktop\hoge` containing files like `hoge1.txt`, `hoge10.png`, `hoge10.txt`, `hoge11.png`, `hoge11.txt`, `hoge12.png`, and `hoge12.txt`.

# 基本的な考え方(例題7)6

実行結果。出力ファイルの中身は、実質的に①outオブジェクトの中身と同じなので、②outと打ち込んで表示。

Environment History Connections

```
tmp <- cbind(names(fasta...
write.table(tmp, out_f, ...
out
```

Files Plots Packages Help Viewer

C: > Users > kadota > Desktop > hoge

	AA	AC	AG	AT	CA	CC	CG	CT	GA	GC	GG	GT	TA	TC	TG	TT
[1,]	1	3	2	1	3	1	2	0	3	1	1	0	0	0	1	0
[2,]	1	3	2	1	1	0	3	0	4	1	2	0	0	0	1	0
[3,]	1	4	1	0	3	1	0	2	2	0	1	0	1	1	1	1
[4,]	1	2	0	0	1	0	0	3	1	0	0	2	1	2	2	4
[5,]	1	3	0	0	1	0	0	3	1	0	0	1	1	2	2	4
[6,]	1	4	0	0	2	0	0	3	1	0	0	1	1	2	1	3
[7,]	1	4	1	0	3	1	0	2	2	0	1	0	1	1	1	1
[8,]	1	4	2	1	3	1	1	0	3	1	1	0	0	0	1	0
[9,]	2	1	1	0	1	0	0	2	2	0	0	2	0	2	2	4
[10,]	1	2	1	0	1	0	0	3	1	0	0	2	0	2	2	4

File Explorer contents:

- ..
- .Rhistory (14.8 KB)
- hoge1.txt (221 B)
- hoge10.png (6.2 KB)
- hoge10.txt (142.1 KB)
- hoge11.png (6.3 KB)
- hoge11.txt (142.1 KB)
- hoge12.png (5.9 KB)
- hoge12.txt (138.8 KB)

# 基本的な考え方(例題7)7

実行結果。出力ファイルの中身は、実質的に①outオブジェクトの中身と同じなので、②outと打ち込んで表示。③全部で10行分ある理由は、入力ファイルが10リードだから。

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins Project: (None)

Console Terminal Jobs

C:/Users/kadota/Desktop/hoge/

```

> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F) #tmpの中身を指定したファイル名で保存
>
> out

```

	AA	AC	AG	AT	CA	CC	CG	CT	GA	GC	GG	GT	TA	TC	TG	TT
[1,]	1	3	2	1	3	1	2	0	3	1	1	0	0	0	1	0
[2,]	1	3	2	1	1	0	3	0	4	1	2	0	0	0	1	0
[3,]	1	4	1	0	3	1	0	2	2	0	1	0	1	1	1	1
[4,]	1	2	0	0	1	0	0	3	1	0	0	2	1	2	2	4
[5,]	1	3	0	0	1	0	0	3	1	0	0	1	1	2	2	4
[6,]	1	4	0	0	2	0	0	3	1	0	0	1	1	2	1	3
[7,]	1	4	1	0	3	1	0	2	2	0	1	0	1	1	1	1
[8,]	1	4	2	1	3	1	1	0	3	1	1	0	0	0	1	0
[9,]	2	1	1	0	1	0	0	2	2	0	0	2	0	2	2	4
[10,]	1	2	1	0	1	0	0	3	1	0	0	2	0	2	2	4

Environment History Connections

```

> kkk_24_43
CACCAGGACATGAAGACGCG
> kkk_28_47
AGGACATGAAGACGCGGACG
> kkk_12_31
TTGAACTCACTACACCAGGA
> kkk_4_23
GTTGTCTTTTGAAGTCACTA
> kkk_5_24
TTGTCTTTTGAAGTCACTAC
> kkk_7_26
GTCTTTTGAAGTCACTACAC
> kkk_12_31
TTGAACTCACTACACCAGGA
> kkk_23_42
ACACCAGGACATGAAGACGC
> kkk_1_20
GAAGTTGTCTTTTGAAGTCA
> kkk_3_22
AGTTGTCTTTTGAAGTCACT

```

# 基本的な考え方(例題7)8

実行結果。出力ファイルの中身は、実質的に①outオブジェクトの中身と同じなので、②outと打ち込んで表示。③全部で10行分ある理由は、入力ファイルが10リードだから。例えば、④3番目のリードに対して行われた2連続塩基の出現頻度解析結果が、⑤3行目の結果に相当。

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Console Terminal Jobs
C:/Users/kadota/Desktop/hoge/
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)#tmpの中身を指定したファイル名で保存
>
> out
```

	AA	AC	AG	AT	CA	CC	CG	CT	GA	GC	GG	GT	TA	TC	TG	TT
[1,]	1	3	2	1	3	1	2	0	3	1	1	0	0	0	1	0
[2,]	1	3	2	1	1	0	3	0	4	1	2	0	0	0	1	0
[3,]	1	4	1	0	3	1	0	2	2	0	1	0	1	1	1	1
[4,]	1	2	0	0	1	0	0	3	1	0	0	2	1	2	2	4
[5,]	1	3	0	0	1	0	0	3	1	0	0	1	1	2	2	4
[6,]	1	4	0	0	2	0	0	3	1	0	0	1	1	2	1	3
[7,]	1	4	1	0	3	1	0	2	2	0	1	0	1	1	1	1
[8,]	1	4	2	1	3	1	1	0	3	1	1	0	0	0	1	0
[9,]	2	1	1	0	1	0	0	2	2	0	0	2	0	2	2	4
[10,]	1	2	1	0	1	0	0	3	1	0	0	2	0	2	2	4

```
> kkk_24_43
CACCAGGACATGAAGACGCG
> kkk_28_47
AGGACATGAAGACGCGGACG
> kkk_12_31
TTGAACTCACTACACCAGGA
> kkk_4_23
GTTGTCTTTTGAAGTCACTA
> kkk_5_24
TTGTCTTTTGAAGTCACTAC
> kkk_7_26
GTCTTTTGAAGTCACTACAC
> kkk_12_31
TTGAACTCACTACACCAGGA
> kkk_23_42
ACACCAGGACATGAAGACGC
> kkk_1_20
GAAGTTGTCTTTTGAAGTCA
> kkk_3_22
AGTTGTCTTTTGAAGTCACT
```



# 基本的な考え方(例題7)9

ゲノムサイズ推定の場合は、リードごとではなく入力ファイル全体で考える。列(columns)ごとの総和(summation)を計算する、①colSums関数を実行。真ん中のSのみ大文字であることに注意。

The screenshot shows the RStudio interface with the following content:

```
> out
      AA AC AG AT CA CC CG CT GA GC GG GT TA TC TG TT
[1,]  1  3  2  1  3  1  2  0  3  1  1  0  0  0  1  0
[2,]  1  3  2  1  1  0  3  0  4  1  2  0  0  0  1  0
[3,]  1  4  1  0  3  1  0  2  2  2  0  1  0  1  1  1
[4,]  1  2  0  0  1  0  0  3  1  0  0  2  1  2  2  4
[5,]  1  3  0  0  1  0  0  3  1  0  0  1  1  2  2  4
[6,]  1  4  0  0  2  0  0  3  1  0  0  1  1  2  1  3
[7,]  1  4  1  0  3  1  0  2  2  2  0  1  0  1  1  1
[8,]  1  4  2  1  3  1  1  0  3  1  1  0  0  0  1  0
[9,]  2  1  1  0  1  0  0  2  2  0  0  2  0  2  2  4
[10,] 1  2  1  0  1  0  0  3  1  0  0  2  0  2  2  4

> colSums(out)
      AA AC AG AT CA CC CG CT GA GC GG GT TA TC TG TT
      11 30 10  3 19  4  6 18 20  3  6  8  5 12 14 21
```

The red arrow points to the `colSums(out)` command in the console.

# 基本的な考え方(例題7)10

ゲノムサイズ推定の場合は、リードごとではなく入力ファイル全体で考える。列(columns)ごとの総和(summation)を計算する、①colSums関数を実行。真ん中のSのみ大文字であることに注意。②例えばTTという2連続塩基は、③全部で21回出現したと解釈する。

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

Console Terminal Jobs

```
C:/Users/kadota/Desktop/hoge/
>
> out
  AA AC AG AT CA CC CG CT GA GC GG GT TA TC TG TT
[1,] 1 3 2 1 3 1 2 0 3 1 1 0 0 0 1 0
[2,] 1 3 2 1 1 0 3 0 4 1 2 0 0 0 1 0
[3,] 1 4 1 0 3 1 0 2 2 2 0 1 0 1 1 1
[4,] 1 2 0 0 1 0 0 3 1 0 0 2 1 2 2 4
[5,] 1 3 0 0 1 0 0 3 1 0 0 1 1 2 2 4
[6,] 1 4 0 0 2 0 0 3 1 0 0 1 1 2 1 3
[7,] 1 4 1 0 3 1 0 2 2 0 1 0 1 1 1 1
[8,] 1 4 2 1 3 1 1 0 3 1 1 0 0 0 1 0
[9,] 2 1 1 0 1 0 0 2 2 0 0 2 0 2 2 4
[10,] 1 2 1 0 1 0 0 3 1 0 0 2 0 2 2 4
> colSums(out)
  AA AC AG AT CA CC CG CT GA GC GG GT TA TC TG TT
11 30 10 3 19 4 6 18 20 3 6 8 5 12 14 21
> |
```

① (row 9, col 1) ② (row 10, col 16) ③ (row 10, col 16)

Files Plots Packages Help Viewer

New Folder Delete Rename More

C:\Users\kadota\Desktop\hoge

Name	Size
..	
.Rhistory	14.8 KB
hoge1.txt	221 B
hoge10.png	6.2 KB
hoge10.txt	142.1 KB
hoge11.png	6.3 KB
hoge11.txt	142.1 KB
hoge12.png	5.9 KB
hoge12.txt	138.8 KB

# 基本的な考え方(例題7)11

ゲノムサイズ推定の場合は、リードごとではなく入力ファイル全体で考える。列(columns)ごとの総和(summation)を計算する、①colSums関数を実行。真ん中のSのみ大文字であることに注意。②例えばTTという2連続塩基は、③全部で21回出現したと解釈する。今は例題7の結果に対して、①colSumsを追加で実行した。これを最初からやっているのが例題8。

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Console Terminal Jobs
C:/Users/kadota/Desktop/hoge/
>
> out
      AA AC AG AT CA CC CG CT GA GC GG GT TA TC TG TT
[1,]  1  3  2  1  3  1  2  0  3  1  1  0  0  0  1  0
[2,]  1  3  2  1  1  0  3  0  4  1  2  0  0  0  1  0
[3,]  1  4  1  0  3  1  0  2  2  0  1  0  1  1  1  1
[4,]  1  2  0  0  1  0  0  3  1  0  0  2  1  2  2  4
[5,]  1  3  0  0  1  0  0  3  1  0  0  1  1  2  2  4
[6,]  1  4  0  0  2  0  0  3  1  0  0  1  1  2  1  3
[7,]  1  4  1  0  3  1  0  2  2  0  1  0  1  1  1  1
[8,]  1  4  2  1  3  1  1  0  3  1  1  0  0  0  1  0
[9,]  2  1  1  0  1  0  0  2  2  0  0  2  0  2  2  4
[10,] 1  2  1  0  1  0  0  3  1  0  0  2  0  2  2  4
> colSums(out)
      AA AC AG AT CA CC CG CT GA GC GG GT TA TC TG TT
11 30 10  3 19  4  6 18 20  3  6  8  5 12 14 21
> |
    
```



```

out
colSums(out)
Files Plots Packages Help Viewer
New Folder Delete Rename More
C:\Users\kadota\Desktop\hoge
Name Size
..
.Rhistory 14.8 KB
hoge1.txt 221 B
hoge10.png 6.2 KB
hoge10.txt 142.1 K
hoge11.png 6.3 KB
hoge11.txt 142.1 K
hoge12.png 5.9 KB
hoge12.txt 138.8 K
    
```

# Contents

- Introduction、出現頻度解析(k=2)、出現頻度解析(k=1)
- k=1で実践、multi-FASTAファイル、他の例題を実行
- k=2で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8(k=2)、例題9(k=3)、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11(k=10)、例題12(k=10)、シークエンスエラーを含む場合

# 例題8 (k=2) 1

ゲノムサイズ推定の場合は、リードごとではなく入力ファイル全体で考える。列(columns)ごとの総和(summation)を計算する、①colSums関数を実行。真ん中のSのみ大文字であることに注意。②例えばTTという2連続塩基は、③全部で21回出現したと解釈する。今は例題7の結果に対して、①colSumsを追加で実行した。これを最初から行っているのが、④例題8。⑤colSums関数はここで使っています。コピー実行。

(Rで)塩基配列解析

保護されていない通信 | [iu.a.u-tokyo.ac.jp/~kadota/r\\_seq.html#int](http://iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#int)

## 8. サンプルデータの例題32を実行して得られたmulti-FASTAファイル([sample32.ngs.fasta](#))

2連続塩基 (k=2) の出現頻度情報を得るやり方です。4<sup>2</sup> = 16通りのk-merがあります。全リードを合算した出現頻度を出力するやり方です。

```
in_f <- "sample32.ngs.fasta" #入力ファイル名を指定して読み込み
out_f <- "hoge8.txt" #出力ファイル名を指定して書き出し
param_kmer <- 2 #k-merのkの値を指定

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

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

#本番
hoge <- oligonucleotideFrequency(fasta, width=param_kmer) #k連続塩基の出現頻度情報をhogeに格納
out <- colSums(hoge) #列ごとの総和をoutに格納

#ファイルに保存
write.table(out, out_f, sep="\t", append=F, quote=F, row.names=T, col.names=F) #outの中身を指定して書き出し
```

[トップページ](#)△

# 例題8 (k=2) 2

コピー実行結果。例題7と異なり、例題8では①2連続塩基の出現頻度解析結果をhogeオブジェクトに格納し、②それを入力として、③colSumsを実行した結果を、④outオブジェクトに格納しているので…

```
> 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
>
> ① hoge <- oligonucleotideFrequency(fasta, width=param_kmer) #k連続塩基の出現頻度情報をhogeに格納
> ② out <- colSums(hoge) #列ごとの総和をoutに格納
> ③ write.table(out, out_f, sep="\t", append=F, quote=F, row.names=T, col.names=F)
>
```

The screenshot also shows the Environment pane with the following code snippet:

```
hoge <- oligonucleotideFrequency(fasta, width=param_kmer)
out <- colSums(hoge) #列ごとの総和をoutに格納
write.table(out, out_f, sep="\t", append=F, quote=F, row.names=T, col.names=F)
```

The Files pane shows the directory structure:

Name	Size
..	
.Rhistory	14.8 KB
hoge1.txt	221 B
hoge10.png	6.2 KB
hoge10.txt	142.1 KB
hoge11.png	6.3 KB
hoge11.txt	142.1 KB
hoge12.png	5.9 KB
hoge12.txt	138.8 KB

# 例題8 (k=2) 3

コピー実行結果。例題7と異なり、例題8では①2連続塩基の出現頻度解析結果をhogeオブジェクトに格納し、②それを入力として、③colSumsを実行した結果を、④outオブジェクトに格納しているので、⑤outの中身はこんな感じになります。

The screenshot shows the RStudio interface with the following content:

```
> #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
>
> ①
> hoge <- oligonucleotideFrequency(fasta, width=param_km
er)#k連続塩基の出現頻度情報をhogeに格納
> out <- colSums(hoge) #列ごとの総和をou
tに格納
>
> #ファイルに保存
> write.table(out, out_f, sep="\t", append=F, quote=F, r
ow.names=T, col.names=F)
> out
AA AC AG AT CA CC CG CT GA GC GG GT TA TC TG TT
11 30 10 3 19 4 6 18 20 3 6 8 5 12 14 21
>
```

The Environment pane on the right shows the following objects:

- out
- colSums(hoge)

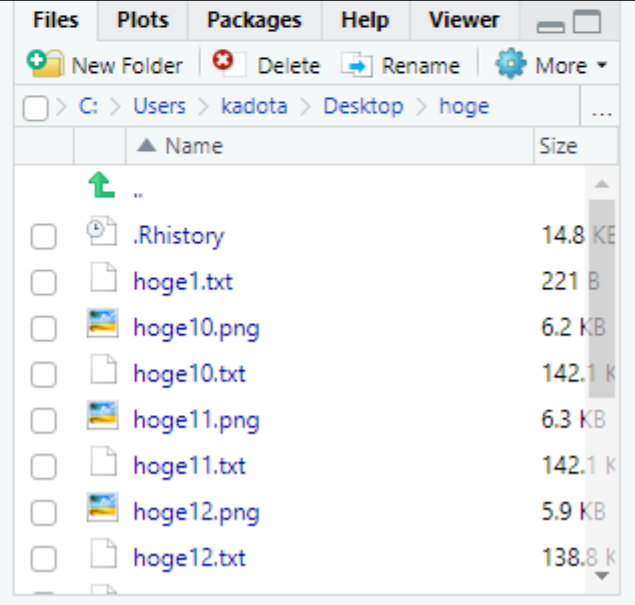
The Files pane shows the directory structure:

```
C:\Users\kadota\Desktop\hoge
..
.Rhistory
hoge1.txt
hoge10.png
hoge10.txt
hoge11.png
hoge11.txt
hoge12.png
hoge12.txt
```

# 例題8 (k=2) 4

コピー実行結果。例題7と異なり、例題8では①2連続塩基の出現頻度解析結果をhogeオブジェクトに格納し、②それを入力として、③colSumsを実行した結果を、④outオブジェクトに格納しているので、⑤outの中身はこんな感じになります。⑥ $4^2 = 16$ 通り全ての2連続塩基(2-mer)が、1回以上出現していることがわかる。ゲノムサイズ推定は、kの値を大きくして、1回以上出現しているk連続塩基(k-mer)の種類数をカウントするのが基本。

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
Console Terminal x Jobs x
C:/Users/kadota/Desktop/hoge/
> #入力ファイルの読み込み
> fasta <- readDNASTringSet(in_f, format="fasta")#in
指定したファイルの読み込み
>
> #本番
> hoge <- oligonucleotideFrequency(fasta, width=param_km
er)#k連続塩基の出現頻度情報をhogeに格納
> out <- colSums(hoge) #列ごとの総和をou
tに格納
>
> #ファイルに保存
> write.table(out, out_f, sep="\t", append=F, quote=F, r
ow.names=T, col.names=F)
> out
AA AC AG AT CA CC CG CT GA GC GG GT TA TC TG TT
11 30 10 3 19 4 6 18 20 3 6 8 5 12 14 21
```





# 例題8 (k=2) 5

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
Console Terminal Jobs
C:/Users/kadota/Desktop/hoge/
> out <- colSums(hoge) #列ごとの総和
tに格納
>
> #ファイルに保存
> write.table(out, out_f, sep="\t", append=F, quote=
ow.names=T, col.names=F)
> out
AA AC AG AT CA CC CG CT GA GC GG GT TA TC TG TT
11 30 10 3 19 4 6 18 20 3 6 8 5 12 14 21
> out > 0
  AA  AC  AG  AT  CA  CC  CG  CT  GA  GC
TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
  GG  GT  TA  TC  TG  TT
TRUE TRUE TRUE TRUE TRUE TRUE
> sum(out > 0)
[1] 16
> |
```

コピー実行結果。例題7と異なり、例題8では①2連続塩基の出現頻度解析結果をhogeオブジェクトに格納し、②それを入力として、③colSumsを実行した結果を、④outオブジェクトに格納しているので、⑤outの中身はこんな感じになります。⑥ $4^2 = 16$ 通り全ての2連続塩基(2-mer)が、1回以上出現していることがわかる。ゲノムサイズ推定は、kの値を大きくして、1回以上出現しているk連続塩基(k-mer)の種類数をカウントするのが基本。この場合はkの値が小さい(= 2)ののでは無意味だが、⑥の結果をベースとして考えると、この入力ファイル(sample32\_ngs.fasta)の推定ゲノムサイズは、⑦16塩基ということになる(ちなみに正解は50塩基)。⑦のsum関数は、TRUEとなった要素数(1回以上出現したk-merの数)をカウントしていることに相当します。



# Contents

- Introduction、出現頻度解析( $k=2$ )、出現頻度解析( $k=1$ )
- $k=1$ で実践、multi-FASTAファイル、他の例題を実行
- $k=2$ で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8( $k=2$ )、例題9( $k=3$ )、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11( $k=10$ )、例題12( $k=10$ )、シーケンスエラーを含む場合

# 例題9 (k=3) 1

(Rで)塩基配列解析

保護されていない通信 | [iu.a.u-tokyo.ac.jp/~kadota/r\\_seq.html#intro\\_general\\_kmer\\_n\\_bios...](http://iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#intro_general_kmer_n_bios...)

## 9. サンプルデータの例題32を実行して得られたmulti-FASTAファイル([sample32\\_ngs.fasta](#))の場合:

3連続塩基 (k=3) の出現頻度情報を得るやり方です。4<sup>3</sup> = 64通りのk-merの出現頻度を計算することになります。全リードを合算した出現頻度を出力するやり方です。

```
in_f <- "sample32_ngs.fasta"      #入力ファイル名を指定してin_fに格納
out_f <- "hoge9.txt"             #出力ファイル名を指定してout_fに格納
param_kmer <- 3                  #k-merのkの値を指定

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

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

#本番
hoge <- oligonucleotideFrequency(fasta, width=param_kmer)#k連続塩基の出現頻度情報をhogeに格納
out <- colSums(hoge)             #列ごとの総和をoutに格納

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

[トップページ](#)△

①例題9。②k = 3になっただけです。コピー実行。実行結果。

# 例題9 (k=3) 2

The screenshot shows the RStudio interface with the following components:

- Console:** Contains R code for reading a FASTA file, calculating oligonucleotide frequencies, and saving the results to a table.
- Environment:** Shows the current workspace with variables like 'hoge' and 'out'.
- Files:** A file explorer showing the directory structure on the desktop.

```
> 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 <- oligonucleotideFrequency(fasta, width=param_kmer)#k連続塩基の出現頻度情報をhogeに格納
> out <- colSums(hoge) #列ごとの総和をoutに格納
>
> #ファイルに保存
> write.table(out, out_f, sep="\t", append=F, quote=F, row.names=T, col.names=F)
>
```

Name	Size
..	
.Rhistory	14.8 KB
hoge1.txt	221 B
hoge10.png	6.2 KB
hoge10.txt	142.1 KB
hoge11.png	6.3 KB
hoge11.txt	142.1 KB
hoge12.png	5.9 KB
hoge12.txt	138.8 KB

# 例題9 (k=3) 3

①例題9。②k = 3になっただけです。コピペ実行。実行結果。③outの中身を表示。4種類の塩基が3連続なので $4^3 = 64$ 種類。

RStudio interface showing the execution of a script to calculate k-mer frequencies for k=3. The console output is as follows:

```

C:/Users/kadota/Desktop/hoge/
> write.table(out, out_1, sep = "\t", append=F, quote=F,
row.names=T, col.names=F)
> out
AAA AAC AAG AAT ACA ACC ACG ACT AGA AGC AGG AGT ATA
  0  7  4  0  7  4  4 13  3  0  5  2  0
ATC ATG ATT CAA CAC CAG CAT CCA CCC CCG CCT CGA CGC
  0  3  0  0 11  4  3  4  0  0  0  0  3
CGG CGT CTA CTC CTG CTT GAA GAC GAG GAT GCA GCC GCG
  1  0  5  7  0  5 11  7  0  0  0  0  2
GCT GGA GGC GGG GGT GTA GTC GTG GTT TAA TAC TAG TAT
  0  6  0  0  0  0  5  0  3  0  4  0  0
TCA TCC TCG TCT TGA TGC TGG TGT TTA TTC TTG TTT
  7  0  0  5 10  0  0  4  0  0 11 10
> length(out)
[1] 64
> sum(out > 0)
[1] 32
>
  
```

The Environment pane shows the variable `out` with the following summary statistics:

```

out
length(out)
sum(out > 0)
  
```

The Files pane shows the directory structure:

```

C: > Users > kadota > Desktop > hoge
Name                               Size
..
.Rhistory                           14.8 KB
hoge1.txt                            221 B
hoge10.png                           6.2 KB
hoge10.txt                           142.1 KB
hoge11.png                           6.3 KB
hoge11.txt                           142.1 KB
hoge12.png                           5.9 KB
hoge12.txt                           138.8 KB
  
```

# 例題9 (k=3) 4

①例題9。②k = 3になっただけです。コピペ実行。実行結果。③outの中身を表示。4種類の塩基が3連続なので $4^3 = 64$ 種類。④ベクトルの要素数を返すlength関数の実行結果も64なので妥当。

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins Project: (None)

Console Terminal x Jobs x

```
C:/Users/kadota/Desktop/hoge/
> write.table(out, out_1, sep = "\t", append=F, quote=F,
row.names=T, col.names=F)
> out
AAA AAC AAG AAT ACA ACC ACG ACT AGA AGC AGG AGT ATA
  0  7  4  0  7  4  4 13  3  0  5  2  0
ATC ATG ATT CAA CAC CAG CAT CCA CCC CCG CCT CGA CGC
  0  3  0  0 11  4  3  4  0  0  0  0  3
CGG CGT CTA CTC CTG CTT GAA GAC GAG GAT GCA GCC GCG
  1  0  5  7  0  5 11  7  0  0  0  0  2
GCT GGA GGC GGG GGT GTA GTC GTG GTT TAA TAC TAG TAT
  0  6  0  0  0  0  5  0  3  0  4  0  0
TCA TCC TCG TCT TGA TGC TGG TGT TTA TTC TTG TTT
  7  0  0  7 10  0  0  4  0  0 11 10
> length(out)
[1] 64
> sum(out > 0)
[1] 32
>
```

Environment History Connections

To Console To Source

```
out
length(out)
sum(out > 0)
```

Files Plots Packages Help Viewer

New Folder Delete Rename More

C: > Users > kadota > Desktop > hoge

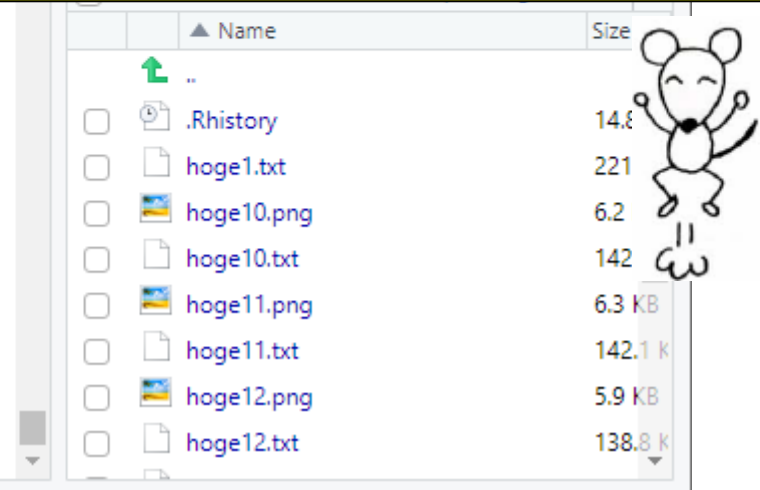
Name	Size
..	
.Rhistory	14.8 KB
hoge1.txt	221 B
hoge10.png	6.2 KB
hoge10.txt	142.1 KB
hoge11.png	6.3 KB
hoge11.txt	142.1 KB
hoge12.png	5.9 KB
hoge12.txt	138.8 KB

# 例題9 (k=3) 5

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
Console Terminal Jobs
C:/Users/kadota/Desktop/hoge/
> writeTable(out, out_f, sep="\t", append=T, quote=
ow.names=T, col.names=F)
> out
AAA AAC AAG AAT ACA ACC ACG ACT AGA AGC AGG AGT ATA
  0  7  4  0  7  4  4 13  3  0  5  2  0
ATC ATG ATT CAA CAC CAG CAT CCA CCC CCG CCT CGA CGC
  0  3  0  0 11  4  3  4  0  0  0  0  3
CGG CGT CTA CTC CTG CTT GAA GAC GAG GAT GCA GCC GCG
  1  0  5  7  0  5 11  7  0  0  0  0  2
GCT GGA GGC GGG GGT GTA GTC GTG GTT TAA TAC TAG TAT
  0  6  0  0  0  0  5  0  3  0  4  0  0
TCA TCC TCG TCT TGA TGC TGG TGT TTA TTC TTG TTT
  7  0  0  7 10  0  0  4  0  0 11 10
> length(out)
[1] 64
> sum(out > 0)
[1] 32
>
  
```

①例題9。②k = 3になっただけです。コピペ実行。実行結果。③outの中身を表示。4種類の塩基が3連続なので $4^3 = 64$ 種類。④ベクトルの要素数を返すlength関数の実行結果も64なので妥当。⑤1回以上出現したk-merの数は32。これはk=3でのゲノムサイズ推定値であり、k=2のときの推定値(16塩基)のときよりも正解(50塩基)に近づいていることがわかる。こんな感じで簡単なテクニックさえあれば、シミュレーションデータを自在に作成して自分で検証して納得することができるのがよいですね。



# Contents

- Introduction、出現頻度解析( $k=2$ )、出現頻度解析( $k=1$ )
- $k=1$ で実践、multi-FASTAファイル、他の例題を実行
- $k=2$ で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8( $k=2$ )、例題9( $k=3$ )、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11( $k=10$ )、例題12( $k=10$ )、シーケンスエラーを含む場合



# サンプルデータ(例題33)

①サンプルデータの例題33。より現実の解析に近づけるべく、②1000塩基からなる仮想ゲノム配列を作成。③20塩基の長さで200リードのNGSデータとすることで、先ほどと同じゲノムサイズの4倍のデータ量(4X coverage)としている。行き来が大変なので、コピペ実行したつもりでよい。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#sa

33. k-mer解析用のランダム配列から生成したFASTA形式ファイル(sample33\_ref.fasta)とsample33\_ngs.fastaです。

1000塩基の長さのリファレンス配列を生成したのち、20塩基長の部分配列を抽出したものです。塩基の存在比はAが22%, Cが28%, Gが28%, Tが22%にしています。リファレンス配列(仮想ゲノム配列)がsample33\_ref.fastaで、200リードからなる仮想NGSデータがsample33\_ngs.fastaです。リード長20塩基で200リードなのでトータル4,000塩基となり、1,000塩基からなる元のゲノム配列の4倍シーケンスしていることとなります(4X coverageに相当)。 [イントロ](#) | [NGS](#) | [配列取得](#) | [シミュレーションデータ](#) | [ランダムな塩基配列の生成から](#)と基本的に同じです。

```

out_f1 <- "sample33_ref.fasta"
out_f2 <- "sample33_ngs.fasta"
param_len_ref <- 1000
narabi <- c("A","C","G","T")
param_composition <- c(22, 28, 28, 22)
param_len_ngs <- 20
param_num_ngs <- 200
param_desc <- "kkk"

#必要なパッケージをロード
library(Biostrings)

#本番(リファレンス配列生成)
set.seed(1010)
    
```

#出力ファイル名を指定してout\_f1に格納  
 #出力ファイル名を指定してout\_f2に格納  
 #リファレンス配列の長さを指定  
 #以下の数値指定時にACGTの並びを間違えないようにするために  
 #(A,C,G,Tの並びで)各塩基の存在比率を指定  
 #リード長を指定  
 #リード数を指定  
 #FASTA形式ファイルのdescription行に記述する内容

#パッケージの読み込み

#おまじない(同じ乱数になるようにするため)

[トップページへ](#)

# サンプルデータ(例題33)

①サンプルデータの例題33。より現実の解析に近づけるべく、②1000塩基からなる仮想ゲノム配列を作成。③20塩基の長さで200リードのNGSデータとすることで、先ほどと同じゲノムサイズの4倍のデータ量(4X coverage)としている。行き来が大変なので、コピペ実行したつもりでよい。重要なのは、④のファイル(sample33\_ngs.fasta)を入力としてゲノムサイズ推定を行ったときの正解は1,000塩基だということ。

(Rで)塩基配列解析 × +  
 保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#sa

33. k-mer解析用のランダム配列から生成したFASTA形式ファイル(sample33\_ref.fasta)です。  
 1000塩基の長さのリファレンス配列を生成したのち、20塩基長の部分配列(1000塩基の長さのリファレンス配列を生成したのち、20塩基長の部分配列)がsample33\_ref.fastaで、200リードからなる仮想NGSデータ(長さ20塩基で200リードなのでトータル4,000塩基となり、1,000塩基から生成したリファレンス配列の4倍の長さ)をsample33\_ngs.fastaとして生成しています(4X coverageに相当)。イントロ | NGS | 配列

```

out_f1 <- "sample33_ref.fasta"
out_f2 <- "sample33_ngs.fasta"
param_len_ref <- 1000
narabi <- c("A","C","G","T")
param_composition <- c(22, 28, 28, 22)
param_len_ngs <- 20
param_num_ngs <- 200
param_desc <- "kkk"

#必要なパッケージをロード
library(Biostrings)

#本番(リファレンス配列生成)
set.seed(1010)
    
```

②

④

③

```

#出力ファイル名を指定してout_f1に格納
#出力ファイル名を指定してout_f2に格納
#リファレンス配列の長さを指定
#以下の数値指定時にACGTの並びを間違えないようにする
#(A,C,G,Tの並びで)各塩基の存在比率を指定
#リード長を指定
#リード数を指定
#FASTA形式ファイルのdescription行に記述する内容

#パッケージの読み込み

#おまじない(同じ乱数になるようにするため)
    
```



[トップページへ](#)

# サンプルデータ(例題33)

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
Console Terminal x Jobs x
C:/Users/5/Desktop/hoge/
> 4^3
[1] 64
> 4^4
[1] 256
> 4^5
[1] 1024
> 4^6
[1] 4096
> 4^7
[1] 16384
> 4^8
[1] 65536
> 4^9
[1] 262144
> 4^10
[1] 1048576
>

```



①サンプルデータの例題33。より現実の解析に近づけるべく、②1000塩基からなる仮想ゲノム配列を作成。③20塩基の長さで200リードのNGSデータとすることで、先ほどと同じゲノムサイズの4倍のデータ量(4X coverage)としている。行き来が大変なので、コピー実行したつもりでよい。重要なのは、④のファイル(sample33\_ngs.fasta)を入力としてゲノムサイズ推定を行ったときの正解は1,000塩基だということ。それゆえ、⑤k=3で1回以上出現したk-merの数を調べたとしても、最大値でも $4^3 = 64$ にしかならないので、1,000塩基のゲノムサイズ推定を行うのは原理的に不可能という事実を正しく理解しておかねばならない。1,000塩基のゲノムサイズ推定目的の場合は、最低でも⑥k=5は必要だが、現実にはシーケンスエラーなども含むのもっと大きなk値を選択します。



# Contents

- Introduction、出現頻度解析( $k=2$ )、出現頻度解析( $k=1$ )
- $k=1$ で実践、multi-FASTAファイル、他の例題を実行
- $k=2$ で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8( $k=2$ )、例題9( $k=3$ )、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11( $k=10$ )、例題12( $k=10$ )、シークエンスエラーを含む場合

# 例題11 (k=10) 1

①の、②例題11。③k=10で、④正解ゲノムサイズ1,000塩基だということが分かっているファイル(sample33\_ngs.fasta)をダウンロードしておいて、コピー実行。

(Rで)塩基配列解析

保護されていない通信 | [iu.a.u-tokyo.ac.jp/~kadota/r\\_seq.html#intro\\_q](http://iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#intro_q) | kmer\_n\_bios...

## 11. サンプルデータの例題33を実行して得られたmulti-FASTAファイル(sample33\_ngs.fasta)の場合:

10連続塩基 (k=10) の出現頻度情報を得るやり方です。4<sup>10</sup> = 1,048,576通りのk-merの出現頻度を計算することになります。全リードを合算した出現頻度を出力するやり方です。

```
in_f <- "sample33_ngs.fasta"           #入力ファイル名を指定してin_fに格納
out_f <- "hoge11.txt"                  #出力ファイル名を指定してout_fに格納
param_kmer <- 10                       #k-merのkの値を指定

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

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

#本番
hoge <- oligonucleotideFrequency(fasta, width=param_kmer)#k連続塩基の出現頻度情報をhogeに格納
out <- colSums(hoge)                   #列ごとの総和をoutに格納

#ファイルに保存
write.table(out, out_f, sep="\t", append=F, quote=F, row.names=T, col.names=F)#outの中身を指定し、
length(out)                            #4^param_kmerの値を表示
sum(out > 0)                             #1回以上出現したk-merの種類数を表示
```

# 例題11 (k=10) 2

①の、②例題11。③k=10で、④正解ゲノムサイズ1,000塩基だということが分かっているファイル(sample33\_ngs.fasta)をダウンロードしておいて、コピー実行。実行結果。⑤推定ゲノムサイズは904。真の値に近いといえは近いが、若干小さめともいえる。この主な理由は、入力ファイルのデータ量が少なめだから(4X coverage)。データ量を10X coverageに増やしたものが例題12。

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
Console Terminal Jobs
R 4.2.0 · C:/Users/kadota/Desktop/hogo/
>
> #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 <- oligonucleotideFrequency(fasta, width=param_kmer)
#k連続塩基の出現頻度情報をhogeに格納
> out <- colSums(hoge) #列ごとの総和をoutに格納
>
> #ファイルに保存
> write.table(out, out_f, sep="\t", append=F, quote=F, row.names=T, col.names=F)#outの中身を指定したファイル名で保存
> length(out) #4^param_kmerの値を表示
[1] 1048576
> sum(out > 0) #1回以上出現したk-merの種類数を表示
[1] 904
>
> |
```

out_f1	"sample32_ngs.fasta"
out_f2	"sample32_ngs.fasta"
param_...	chr [1:5] "A" "C" ...
param_...	num [1:4] 22 28 28...
param_...	"kkk"
param_...	10

Name	Size	M
..		
seq_20.fasta	29 B	M
hoge4.fa	299 B	M
hoge2.txt	104 B	M
hoge1.txt	221 B	M



# Contents

- Introduction、出現頻度解析( $k=2$ )、出現頻度解析( $k=1$ )
- $k=1$ で実践、multi-FASTAファイル、他の例題を実行
- $k=2$ で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8( $k=2$ )、例題9( $k=3$ )、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11( $k=10$ )、例題12( $k=10$ )、シークエンスエラーを含む場合

# 例題12 (k=10) 1

①の、②例題12。③k=10で、④正解ゲノムサイズ1,000塩基から20塩基長のリードを500個分生成した、総塩基数10,000のファイル(sample34\_ngs.fasta)をダウンロードして、コピー実行。

(Rで)塩基配列解析

保護されていない通信 | iu.a.u-tokyo.ac.jp/~kadota/r\_seq.html#intro... | kmer\_n\_bios...

## 12. サンプルデータの例題34を実行して得られたmulti-FASTAファイル(sample34\_ngs.fasta)の場合:

10連続塩基 (k=10) の出現頻度情報を得るやり方です。4<sup>10</sup> = 1,048,576通りのk-merの出現頻度を計算することになります。全リードを合算した出現頻度を出力するやり方です。

```

in_f <- "sample34_ngs.fasta"      #入力ファイル名を指定してin_fに格納
out_f <- "hoge12.txt"            #出力ファイル名を指定してout_fに格納
param_kmer <- 10                 #k-merのkの値を指定

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

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

#本番
hoge <- oligonucleotideFrequency(fasta, width=param_kmer)#k連続塩基の出現頻度情報をhogeに格納
out <- colSums(hoge)              #列ごとの総和をoutに格納

#ファイルに保存
write.table(out, out_f, sep="\t", append=F, quote=F, row.names=T, col.names=F)#outの中身を指定して出力
length(out)                       #4^param_kmerの値を表示
sum(out > 0)                       #1回以上出現したk-merの種類数を表示

```



# 例題12 (k=10) 2

①の、②例題13。③k=10で、④正解ゲノムサイズ1,000塩基から20塩基長のリードを500個分生成した、総塩基数10,000のファイル(sample34\_ngs.fasta)をダウンロードして、コピペ実行。実行結果。⑤推定ゲノムサイズは985。ほぼ正解ですね。シークエンスエラーのないデータなら、10X程度のcoverageであれば十分だということがわかります。

out_f	"hoge12.txt"
out_f1	"sample32_ref.fast..."
out_f2	"sample32_ngs.fast..."
param_...	chr [1:5] "A" "C" ...
param_...	num [1:4] 22 28 28...
param_...	"kkk"
param_...	10

Name	Size	M
..		
seq_20.fasta	29 B	M
hoge4.fa	299 B	M
hoge2.txt	104 B	M
hoge1.txt	221 B	M

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
Console Terminal Jobs
R 4.2.0 C:/Users/kadota/Desktop/hogo/
>
> #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 <- oligonucleotideFrequency(fasta, width=param_kmer)
#k連続塩基の出現頻度情報をhogeに格納
> out <- colSums(hoge) #列ごとの総和をoutに
格納
>
> #ファイルに保存
> write.table(out, out_f, sep="\t", append=F, quote=F, row.
names=T, col.names=F)#outの中身を指定したファイル名で保存
> length(out) #4^param_kmerの値を
表示
[1] 1048576
> sum(out > 0) #1回以上出現したk-me
rの種類数を表示
[1] 985
>
> |
  
```



# 例題12 (k=10) 3

①の、②例題13。③k=10で、④正解ゲノムサイズ1,000塩基から20塩基長のリードを500個分生成した、総塩基数10,000のファイル(sample34\_ngs.fasta)をダウンロードして、コピペ実行。実行結果。⑤推定ゲノムサイズは988。ほぼ正解ですね。シーケンスエラーのないデータなら、10X程度のcoverageであれば十分だということがわかります。ちなみに最初のほうのスライドでも解説しているが、「長さ1,000の塩基配列を10-merで分割すると、 $(1,000 - 10 + 1)$ 個のk-merを生成可能」でした。従って、この場合の「1回以上出現したk-merの数」をカウントするやり方で得られるパーフェクトの値は991だという点は念頭においておくとよい。

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
Console Terminal Jobs
R 4.2.0 · C:/Users/kadota/Desktop/hogo/
>
> #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 <- oligonucleotideFrequency(fasta, width=param_kmer)
#k連続塩基の出現頻度情報をhogeに格納
> out <- colSums(hoge) #列ごとの総和をoutに
格納
>
> #ファイルに保存
> write.table(out, out_f, sep="\t", append=F, quote=F, row.
names=T, col.names=F)#outの中身を指定したファイル名で保存
> length(out) #4^param_kmerの値を
表示
[1] 1048576
> sum(out > 0) #1回以上出現したk-me
rの種類数を表示
[1] 985
>
> |

```



Name	Size	M
..		
seq_20.fasta	29 B	N
hoge4.fa	299 B	N
hoge2.txt	104 B	N
hoge1.txt	221 B	N

# Contents

- Introduction、出現頻度解析( $k=2$ )、出現頻度解析( $k=1$ )
- $k=1$ で実践、multi-FASTAファイル、他の例題を実行
- $k=2$ で実践、関数マニュアル、例題2を実行、例題7を実行
- 確率の話、作図(例題10)、作図(例題11)、作図(例題12)
- 塩基配列解析の基礎
  - GC含量、ランダム配列を生成、部分配列の切り出し
- ゲノムサイズ推定
  - サンプルデータ(例題32)、被覆率(coverage)、基本的な考え方(例題7)
  - 例題8( $k=2$ )、例題9( $k=3$ )、1,000塩基の仮想ゲノム(サンプルデータの例題33)
  - 例題11( $k=10$ )、例題12( $k=10$ )、シークエンスエラーを含む場合

# シーケンスエラー1

## 参考図書

- 藤博幸・編、よくわかるバイオインフォマティクス入門、講談社、2018  
坊農秀雅 著、Dr.Bonoの生命科学データ解析、MEDSi、2017  
坊農秀雅 著、RNA-Seqデータ解析 WETラボのための鉄板レシピ、羊土社、2019

## 講義日程 (2020年度)

1. 2020年05月12日(谷澤)
2. 2020年05月19日(門田)  
講義資料PDF(最終更新: 2020.03.16)  
(Rで)塩基配列解析  
seq\_20.fasta  
NGSハンズオン講習会の2016年7月20日の講義資料(スライド114-)  
課題1: kadai1.fasta  
課題3: kadai3.fasta  
課題4: kadai4\_2mer.txt, kadai4.R  
課題5: kadai5.fasta
3. 2020年05月26日(門田)
4. 2020年06月02日(門田)



実際には、シーケンスエラーを含むデータからゲノムサイズ推定が行われます。①で基本戦略についての解説がなされています。

# シーケンスエラー2

## ゲノム解析、塩基配列解析

門田 幸二

- NGS解析手段、ウェブツール(DDBJ Pipeline)との連携
- k-mer解析 (k個の連続塩基に基づく各種解析) の基礎と応用
- 塩基ごとの出現頻度解析(k=1)、2連続塩基の出現頻度解析(k=2)
- 塩基配列解析を行うための基本スキルの復習や作図
- de novoアセンブリ時のエラー補正やゲノムサイズ推定の基本的な考え方

› 講義動画  › スライド(7.3MB)  

› 予習項目  › 講義詳細  › 解析データ(ZIP:2.2MB) 

実際には、シーケンスエラーを含むデータからゲノムサイズ推定が行われます。①で基本戦略についての解説がなされています。②のスライド114以降。