

バイオインフォマティクス ～Rパッケージの話～

¹東京大学・大学院農学生命科学研究科
アグリバイオインフォマティクス教育研究ユニット
²東京大学・微生物科学イノベーション連携研究機構
門田幸二(かどた こうじ)
kadota@iu.a.u-tokyo.ac.jp
<http://www.iu.a.u-tokyo.ac.jp/~kadota/>

計3回の予定講義内容

- バイオインフォマティクス: 概論とRの基礎(1/25)

バイオインフォマティクスを学ぶ上でのRの位置づけや、基本的な利用法に関する本当に極初級者向けの解説

- バイオインフォマティクス: Rパッケージの話(2/22)

Rを利用する際によく聞くパッケージというものの概念的な話や、どのようにして利用したいパッケージを見つけ出すかなどのお話。

- バイオインフォマティクス: 解析結果の解釈など(3/15)

ガン vs. 正常などの状態の異なるグループ間でのクラスタリングや発現変動解析を行う実例や結果の解釈についての解説。Rを覚える時間がないヒトでもウェブツールを利用して同様の解析ができる話など。

資料はコチラ

①ググって、②私のホームページの、
③講義、のところにPDFがあります(の
でメモなどをとる必要はありません)。

東大 門田



①

門田 幸二のホームページ

②

名前 門田 幸二(かどた こうじ)

所属 [東京大学 大学院農学生命科学研究科](#) [アグリバイオインフォマティクス教育研究ユニット](#)
[東京大学 微生物科学イノベーション連携研究機構](#)

身分 准教授

研究分野 バイオインフォマティクス(トランスクリプトーム解析)



- [研究テーマ](#) (last modified 2018/03/01)
- [原著論文](#) (last modified 2018/03/08)
- [総説・解説記事・翻訳など](#) (last modified 2017/11/13)
- [略歴](#) (last modified 2018/04/09)
- [講義](#) (last modified 2018/04/06)
- [講演・論文など](#) (last modified 2018/05/04) **NEW**
- [外部研究資金](#) (last modified 2018/04/06)
- [その他](#) (last modified 2018/05/17) **NEW**
- [リンク集](#) (last modified 2018/05/11) **NEW**

③

研究テーマ

トランスクリプトーム解析手法の開発。本ユニットでは、様々なトランスクリプトームデータの解析や新規解析手法の開発を通じて、農学生命科学への応用を目指します。「数式を並べ立てた難解な方法を凌駕する"シンプルな方法"の開発」をモットーとしています。これまでの主な研究成果を三つのカテゴリーに分けていますが、いずれも「トランスクリプトーム解析」でひとまとめにできます。また、実験系の方でも気軽に研究成果を利用可能なように「[\(Rで\)マイクロアレイデータ解析](#)」と「[\(Rで\)塩基配列解析](#)」上にも 下記開発手法中の一部について、その利用法を記述しています。

資料はコチラ

講義 NEW

- [東大・院農・アグリバイオ](#) 「バイオスタティスティクス基礎論」 (2006-2008年度、分担)
- [東大・院農・アグリバイオ](#) 「農学生命情報科学実習I」 (2005-2008年度、分担)
- [東大・院農・アグリバイオ](#) 「機能ゲノム学」 (2005-2008年度は分担、2014-2018年度)
- [東大・院農・アグリバイオ](#) 「バイオインフォマティクス基礎実習」 (2004-2008年度、分担)
- [東大・院農・アグリバイオ](#) 「プロテオーム情報学」 (2009年度、分担)
- [東大・院農・アグリバイオ](#) 「バイオインフォマティクスリテラシーII」 (2009年度、分担)
- [東大・院農・アグリバイオ](#) 「ゲノム情報解析基礎」 (2010-2018年度、分担)
- [東大・院農・アグリバイオ](#) 「オーム情報解析」 (2010-2013年度、分担)
- [東大・院農・アグリバイオ](#) 「農学生命情報科学特論I」 (2010-2014年度は分担、2015-2016年度、2018年度)
- [東大・院農・アグリバイオ](#) 「農学生命情報科学特論II」 (2016年度)
- [東大・院農・アグリバイオ](#) 「農学生命情報科学特論III」 (2011, 2013年度、分担)
- 東大・院農 「情報生命工学」 (1コマ; 2003, 2005, 2009年度)
- 東大・農学部 「生物情報工学」 (2コマ; 2005-2007年度)
- 東大・農学部 「生物情報科学」 (1コマ; 2008-2015年度)
- 東大・農学部 「生物情報科学I」 (1コマ; 2016-2017年度)
- 東大・農学部展開科目 「バイオインフォマティクス」 (2016-2017年度、分担)
- [バイオインフォマティクス人材育成講座](#) [スタンダードコース](#) 「[バイオインフォマティクス 次世代シーケンサー編](#)」 (4コマ; 2011年度; 於沖縄工業高等専門学校(沖縄); 2011.10.15)
- [琉球大学・農学部](#) 「[食品機能科学特別講義I](#)」 (3コマ; 2012年度; [H24年度バイオインフォマティクス・スタンダードコースの一環](#); 2012.09.06; 「[講義資料](#)」; 「[課題](#)」)
- [奈良先端科学技術大学院大学\(NAIST\)・バイオサイエンス研究科](#) 「[ゲノム機能解析特論](#)」 (2013年度; [NAIST植物グローバル教育プロジェクト・平成25年度ワークショップの一環](#); 2013.06.06; 「[ゲノム・トランスクリプトームの各種解析をRで行う](#)」)
- [奈良先端科学技術大学院大学\(NAIST\)・バイオサイエンス研究科](#) 「[ゲノム機能解析特論](#)」 (2014年度; [NAIST植物グローバル教育プロジェクト・平成26年度ワークショップの一環](#); 2014.06.12; 「[Rで配列解析の利用法: GC含量計算から発現変動解析まで](#)」)
- [横浜市立大学・大学院医学研究科](#) 「[ゲノム医学](#)」: [第1回](#)(2019.01.25), [第2回](#)(2019.02.22), [第3回](#)(2019.03.15)

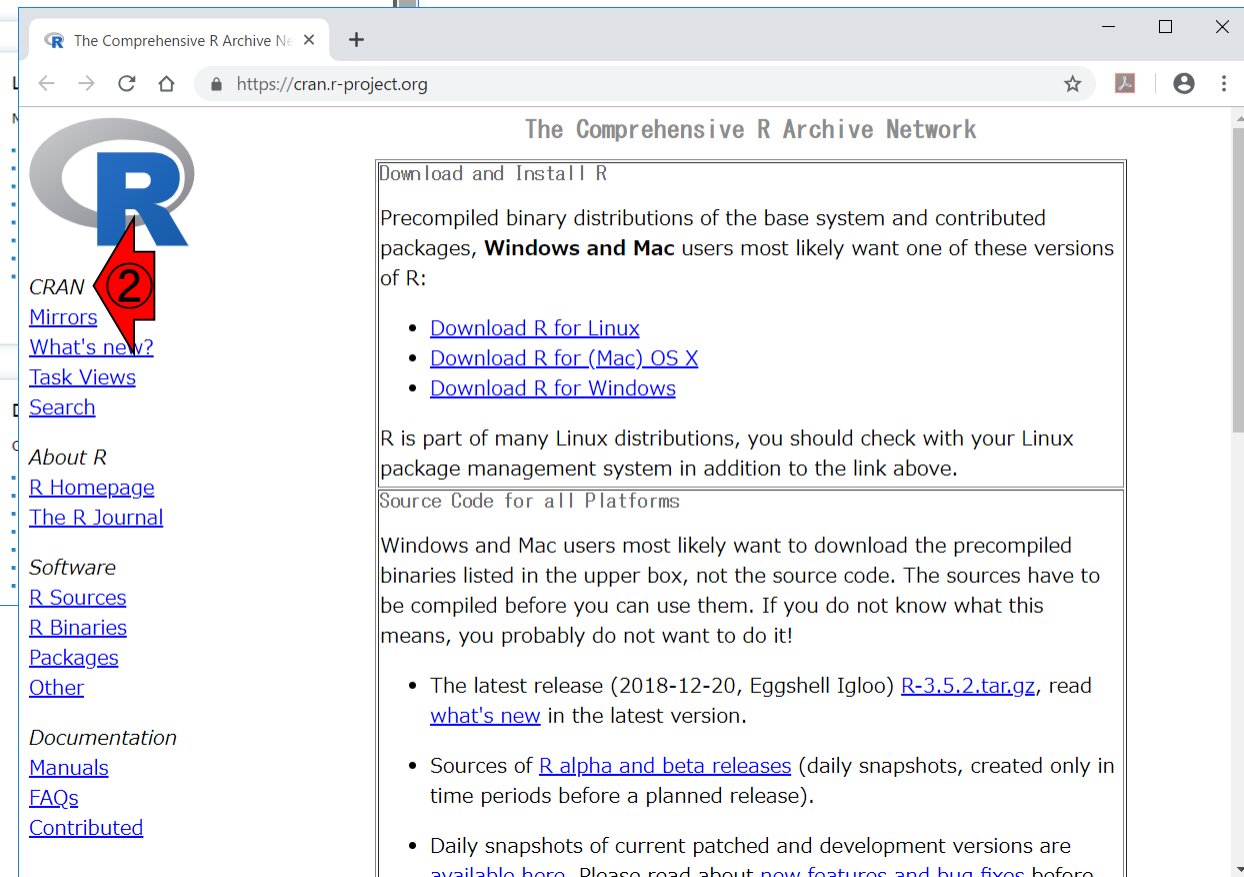
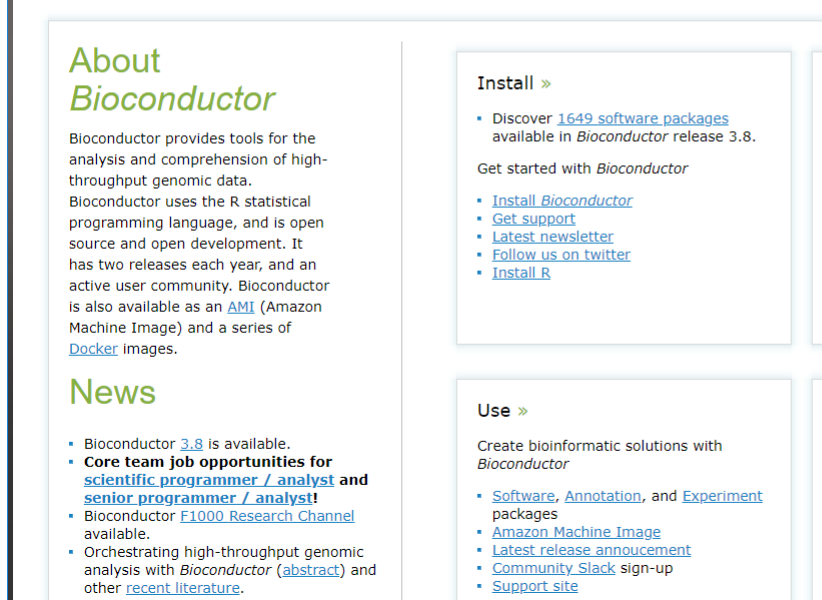
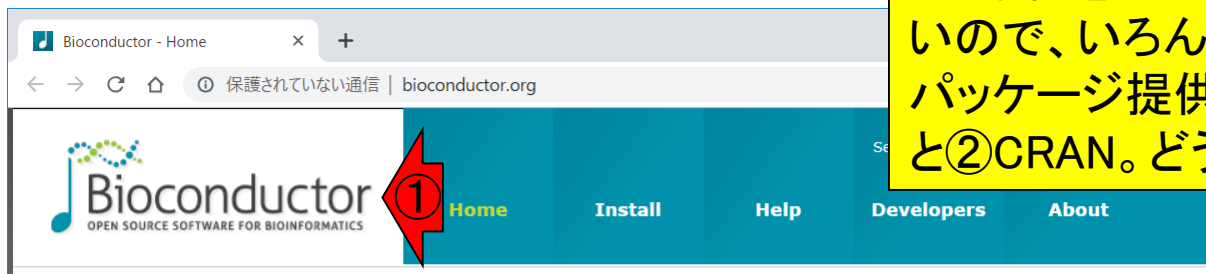
①

Contents

- R本体とRパッケージの関係
- Rパッケージ: BioconductorとCRAN
- Rの基本的な利用法のおさらい: Biostringsを用いた翻訳配列の取得
 - 入力ファイルのダウンロード、Rの起動を作業ディレクトリの変更
 - コピペ実行、コードの解説
- 複数の例題を実行して理解を深める

Rパッケージ

パソコンを購入した直後の状態ではほとんど何もできないので、いろんなソフトウェアをインストールして使います。Rも本体をインストールしただけでは大したことができないので、いろんなパッケージをインストールして使います。パッケージ提供サイトとして有名なのは、①Bioconductorと②CRAN。どういうものが存在するのか概観します。



Bioconductor

①Bioconductorの最新リリースでは、②1,649パッケージが利用可能なようです。②をクリック

The screenshot shows the Bioconductor website home page. The browser address bar shows 'bioconductor.org'. The navigation menu includes 'Home', 'Install', 'Help', 'Developers', and 'About'. A red arrow labeled '1' points to the 'Home' link. The 'Install' section contains a red arrow labeled '2' pointing to the 'Discover 1649 software packages' link. The 'Learn' section lists various resources like 'Courses', 'Support site', and 'Package vignettes'. The 'Use' section lists 'Software, Annotation, and Experiment packages'. The 'Develop' section lists 'Developer resources', 'Use Bioc 'devel'', and 'Build reports'.

Bioconductor - Home

保護されていない通信 | bioconductor.org

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Search:

Home Install Help Developers About

About Bioconductor

Bioconductor provides tools for the analysis and comprehension of high-throughput genomic data. Bioconductor uses the R statistical programming language, and is open source and open development. It has two releases each year, and an active user community. Bioconductor is also available as an [AMI](#) (Amazon Machine Image) and a series of [Docker](#) images.

News

- Bioconductor [3.8](#) is available.
- Core team job opportunities for [scientific programmer / analyst](#) and [senior programmer / analyst](#)!**
- Bioconductor [F1000 Research Channel](#) available.
- Orchestrating high-throughput genomic analysis with *Bioconductor* ([abstract](#)) and other [recent literature](#).

Install »

Discover [1649 software packages](#) available in *Bioconductor* release 3.8.

Get started with *Bioconductor*

- [Install Bioconductor](#)
- [Get support](#)
- [Latest newsletter](#)
- [Follow us on twitter](#)
- [Install R](#)

Learn »

Master *Bioconductor* tools

- [Courses](#)
- [Support site](#)
- [Package vignettes](#)
- [Literature citations](#)
- [Common work flows](#)
- [FAQ](#)
- [Community resources](#)
- [Videos](#)

Use »

Create bioinformatic solutions with *Bioconductor*

- [Software, Annotation, and Experiment packages](#)
- [Amazon Machine Image](#)
- [Latest release announcement](#)
- [Community Slack](#) sign-up
- [Support site](#)

Develop »

Contribute to *Bioconductor*

- [Developer resources](#)
- [Use Bioc 'devel'](#)
- ['Devel' packages](#)
- [Package guidelines](#)
- [New package submission](#)
- [Git source control](#)
- [Build reports](#)

Bioconductor

ページ移動直後は、①1,649パッケージが、②ダウンロード?!ランキング順になっています。③1位はパッケージのインストーラ。上位は基本パッケージが多いです。

Bioconductor - BiocViews

保護されていない通信 | bioconductor.org/packages/release/BiocViews.html#__Software

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Home Install Help Developers About

Home » BiocViews

All Packages

Bioconductor version 3.8 (Release)

Autocomplete biocViews search:

▼ Software (1649)

- ▶ AssayDomain (661)
- ▶ BiologicalQuestion (668)
- ▶ Infrastructure (360)
- ▶ ResearchField (728)
- ▶ StatisticalMethod (572)
- ▶ Technology (1049)
- ▶ WorkflowStep (884)
- ▶ AnnotationData (941)
- ▶ ExperimentData (360)
- ▶ Workflow (23)

Packages found under Software:

Rank based on number of downloads: lower numbers are more frequently downloaded.

Show All entries Search table:

Package	Maintainer	Title	Rank
BiocInstaller	Bioconductor Package Maintainer	Install/Update Bioconductor, CRAN, and github Packages	1
BiocGenerics	Bioconductor Package Maintainer	S4 generic functions for Bioconductor	2
IRanges	Bioconductor Package Maintainer	Infrastructure for manipulating intervals on sequences	3
Biobase	Bioconductor Package Maintainer	Biobase: Base functions for Bioconductor	4
S4Vectors	Bioconductor Package Maintainer	S4 implementation of vector-like and list-like objects	5

Bioconductor

①6-16位を表示。②はゲノムの座標情報を取り扱う系のパッケージです。

Package Name	Maintainer	Description	Number
AnnotationDbi	Bioconductor Package Maintainer	Annotation Database Interface	6
zlibbioc	Bioconductor Package Maintainer	An R packaged zlib-1.2.5	7
GenomicRanges	Bioconductor Package Maintainer	Representation and manipulation of genomic intervals and variables defined along a genome	8
limma	Gordon Smyth	Linear Models for Microarray Data	9
XVector	Hervé Pagès	Representation and manipulation of external sequences	10
BiocParallel	Bioconductor Package Maintainer	Bioconductor facilities for parallel evaluation	11
GenomeInfoDb	Bioconductor Package Maintainer	Utilities for manipulating chromosome and other 'seqname' identifiers	12
Biostrings	H. Pagès	Efficient manipulation of biological strings	13
SummarizedExperiment	Bioconductor Package Maintainer	SummarizedExperiment container	14
DelayedArray	Hervé Pagès	Delayed operations on array-like objects	15
annotate	Bioconductor Package Maintainer	Annotation for microarrays	16

Biostrings

①13位の、②Biostringsパッケージは、③生物配列を効率的に操作する上で有用な関数を多数提供しています。

Package Name	Maintainer	Description	Rank
AnnotationDbi	Bioconductor Package Maintainer	Annotation Database Interface	6
zlibbioc	Bioconductor Package Maintainer	An R packaged zlib-1.2.5	7
GenomicRanges	Bioconductor Package Maintainer	Representation and manipulation of genomic intervals and variables defined along a genome	8
limma	Gordon Smyth	Linear Models for Microarray Data	9
XVector	Hervé Pagès	Representation and manipulation of external sequences	10
BiocParallel	Bioconductor Package Maintainer	Bioconductor facilities for parallel evaluation	11
GenomeInfoDb	Bioconductor Package Maintainer	Utilities for manipulating chromosome and other 'seqname' identifiers	12
Biostrings	H. Pagès	Efficient manipulation of biological strings	13
SummarizedExperiment	Bioconductor Package Maintainer	SummarizedExperiment container	14
DelayedArray	Hervé Pagès	Delayed operations on array-like objects	15
annotate	Bioconductor Package Maintainer	Annotation for microarrays	16
	Bioconductor	genefilter: methods for filtering	

Contents

- R本体とRパッケージの関係
- Rパッケージ: BioconductorとCRAN
- Rの基本的な利用法のおさらい: Biostringsを用いた翻訳配列の取得
 - 入力ファイルのダウンロード、Rの起動を作業ディレクトリの変更
 - コピペ実行、コードの解説
- 複数の例題を実行して理解を深める

おさらい。前回の講義(2019年1月25日)で、デモを行った「翻訳配列の取得」は、①塩基配列を入力として、②アミノ酸配列を出力として得る作業でした。

翻訳配列の取得

①

入力: 塩基配列ファイル (sample1.fasta)

```
sample1.fasta - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V)
ヘルプ(H)
>kadota
AGTGACGGTCTT
```

②

出力: アミノ酸配列ファイル (hoge1.fasta)

```
hoge1.fasta - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V)
ヘルプ(H)
>kadota
SDGL
```


①例題1は、②sample1.fastaというファイルを入力として利用します。赤枠部分のみを拡大表示します。

翻訳配列の取得

(Rで)塩基配列解析

保護されていない | www.iu.a.u-tokyo.ac...

イントロ | 一般 | 翻訳配列(translate)を取得(基礎) | Biostrings

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。翻訳のための遺伝コード(genetic code)は、Standard Genetic Codeだそうです。もちろん生物種?!によって多少違い(variants)があるようで、"Standard", "SGC0", "Vertebrate Mitochondrial", "SGC1"などいろいろ選べるようです。

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

1. FASTA形式ファイル(sample1.fasta)の場合 :

multi-FASTAではないsingle-FASTA形式ファイルで。

```
in_f <- "sample1.fasta"      #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.fasta"      #出力ファイル名を指定してout_fに格納

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

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

#本番
fasta <- translate(fasta)    #アミノ酸配列に翻訳した結果をfastaに格納
fasta                                #確認してるだけです

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

[トップページへ](#)

①sample1.fasta上で、右クリックで②「...リンク先を保存」。保存先は、③Desktop上にあるhogeフォルダ。

翻訳配列の取得

1. FASTA形式ファイル(sample1.fasta)の場合

multi-FASTAではないsingle-FASTA形式

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

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

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

```
#本番  
fasta <- translate(fasta)  
fasta
```

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

新しいタブで開く(T)
新しいウィンドウで開く(W)
シークレット ウィンドウで開く(G)
名前を付けてリンク先を保存(K)...
リンクのアドレスをコピー(E)
検証(I) Ctrl+Shift+I

ファイル ホーム 共有 表示

C:\Users\kadota\Desktop\hoge

hogeの検索

更新日時

このフォルダーは空です。

0 個の項目

翻訳配列の取得

大抵の場合、デフォルトの保存先は①ダウンロードになっていますが...

1. FASTA形式

multi-FASTA

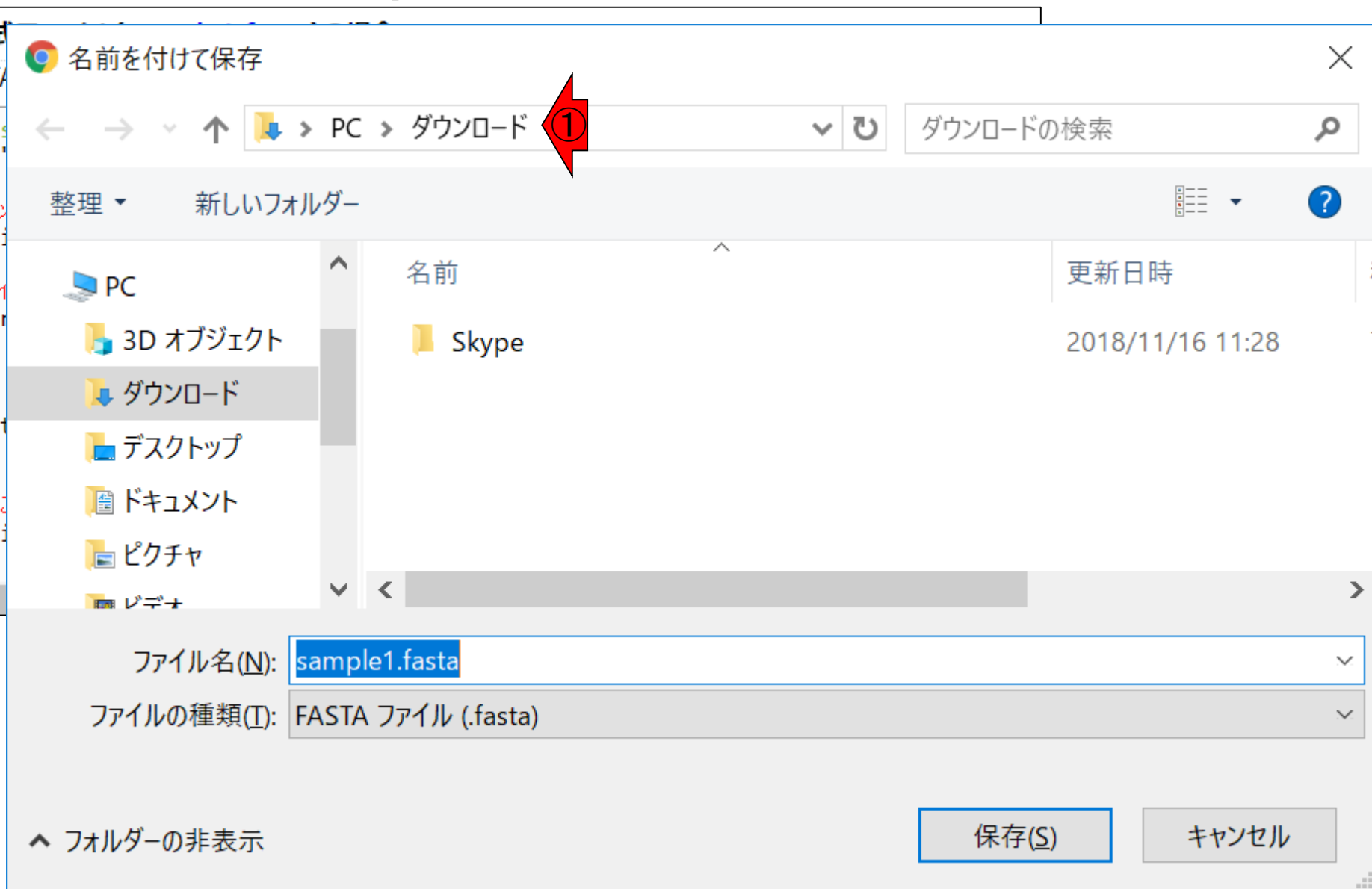
```
in_f <- "  
out_f <- "
```

```
#必要なパ  
library(B
```

```
#入力ファイ  
fasta <- r  
fasta
```

```
#本番  
fasta <- t  
fasta
```

```
#ファイルは  
writeXStr
```



翻訳配列の取得

大抵の場合、デフォルトの保存先は①ダウンロードになっていますが、②デスクトップ上にある、③hogeフォルダです！

1. FASTA形式

multi-FASTA

```
in_f <- "  
out_f <- "
```

```
#必要なパ  
library(B
```

```
#入力ファイ  
fasta <- r  
fasta
```

```
#本番  
fasta <- t  
fasta
```

```
#ファイルは  
writeXStr
```

The screenshot shows a Windows File Explorer window titled "名前を付けて保存" (Save with name). The address bar shows the path "PC > デスクトップ" (PC > Desktop), with a red arrow pointing to the "デスクトップ" folder. The left sidebar shows the "デスクトップ" folder selected, also with a red arrow. The main pane displays a list of folders: "3D オブジェクト", "ダウンロード", "デスクトップ", "ドキュメント", "ピクチャ", and "ビデオ". The "hoge" folder is highlighted, with a red arrow pointing to it. At the bottom, the "ファイル名(N):" field contains "sample1.fasta" and the "ファイルの種類(T):" dropdown is set to "FASTA ファイル (.fasta)". The "保存(S)" (Save) button is highlighted with a blue box.

翻訳配列の取得

1. FASTA形式

multi-FASTA

```
in_f <- "  
out_f <- "
```

```
#必要なパ  
library(B
```

```
#入力ファイ  
fasta <- r  
fasta
```

```
#本番  
fasta <- t  
fasta
```

```
#ファイルは  
writeXStr
```

名前を付けて保存

PC > デスクトップ > hoge

hogeの検索

整理 ▾ 新しいフォルダー

名前 更新日時

PC

3D オブジェクト

ダウンロード

デスクトップ

ドキュメント

ピクチャ

ビデオ

検索条件に一致する項目はありません。

ファイル名(N): sample1.fasta

ファイルの種類(T): FASTA ファイル (.fasta)

保存(S) キャンセル

翻訳配列の取

①ときどきファイルの種類欄がテキストファイルと自動判定され(つまり.txtが付加されて)しまうことがあります。sample1.fasta.txtになるなどしたら、sample1.fastaに戻してから②保存してください。

1. FASTA形式

multi-FASTA

```
in_f <- "
out_f <- "
```

#必要なパッ
library(B

#入力ファイ
fasta <- r
fasta

#本番
fasta <- t
fasta

#ファイルは
writeXStr

名前を付けて保存

PC > デスクトップ > hoge

hogeの検索

整理 ▾ 新しいフォルダー

名前 更新日時

検索条件に一致する項目はありません。

ファイル名(N): sample1.fasta

ファイルの種類(T): FASTA ファイル (.fasta)

保存(S) キャンセル

翻訳配列の取得

①こんな感じに見えていれば無事ダウンロードができています。②×。

「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. FASTA形式ファイル(sample1.fasta)の場合：
multi-FASTAではないsingle-FASTA形式ファイルです。

```
in_f <- "sample1.fasta"      #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.fasta"      #出力ファイル名を指定してout_fに格納

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

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

#本番
fasta <- translate(fasta)    #アミノ酸配列に翻訳した結果をfastaに格納
fasta                                #確認してるだけです

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

sample1.fasta **①** すべて表示 **②** ×

翻訳配列の取得

①hogeフォルダ内に、②sample1.fastaが見えていればOK。②の中身は③のような感じです。

1. FASTA形式ファイル(sample1.fasta)の場合:

multi-FASTAではないsingle-FASTA形式ファイルです。

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

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

```
#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
```

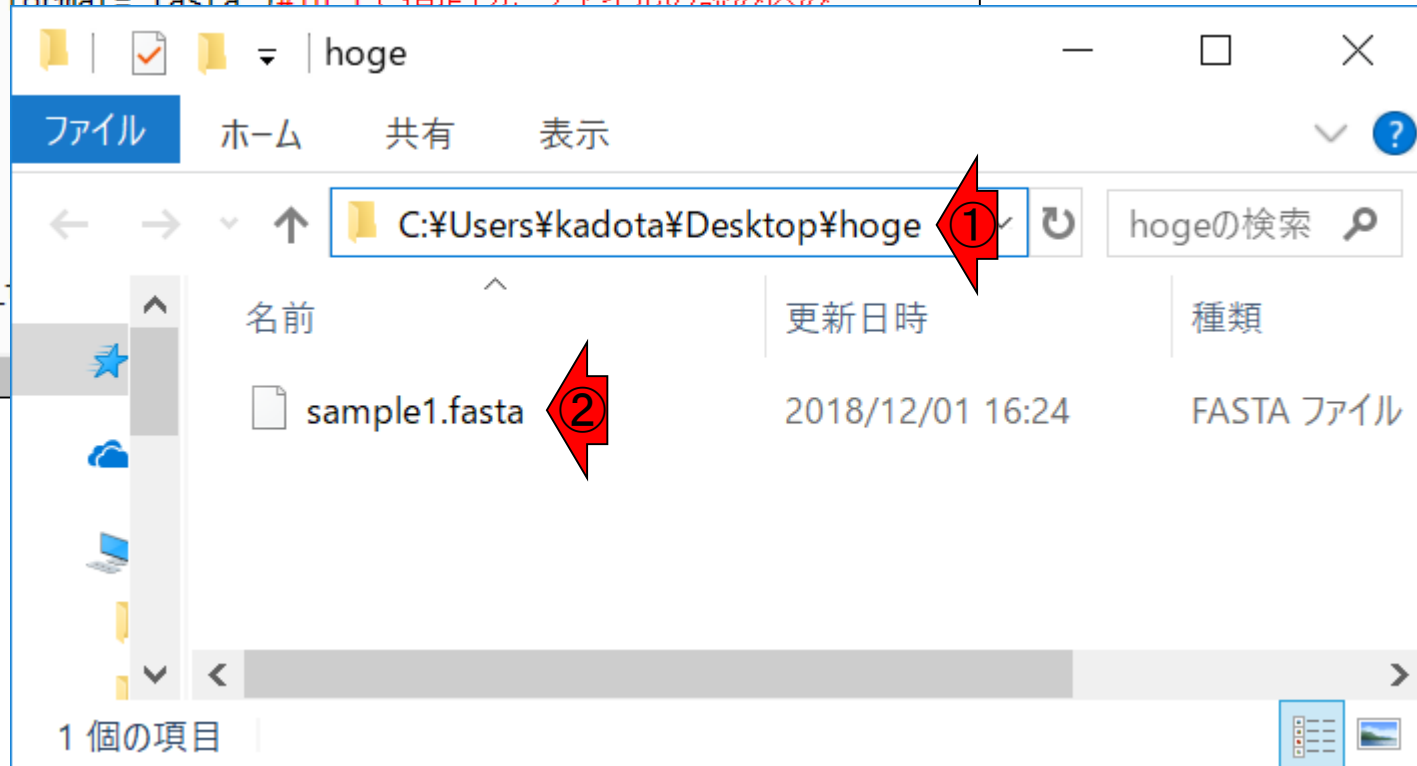
```
#本番
```

```
fasta <- translate(fasta)  
fasta
```

```
#ファイルに保存
```

```
writeXStringSet(fasta, file=out_f)
```

```
>kadota↓  
AGTGACGGTCTT↓
```



翻訳配列の取得

目的をおさらい。①hogeフォルダ内にある、②sample1.fasta中の③塩基配列に対応する翻訳配列(アミノ酸配列)を得るのが目的。

1. FASTA形式ファイル(sample1.fasta)の場合:

multi-FASTAではないsingle-FASTA形式ファイルです。

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

②

#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")#in_fで指定したファイルの読み込み
fasta
```

```
#本番
```

```
fasta <- translate(fasta)
fasta
```

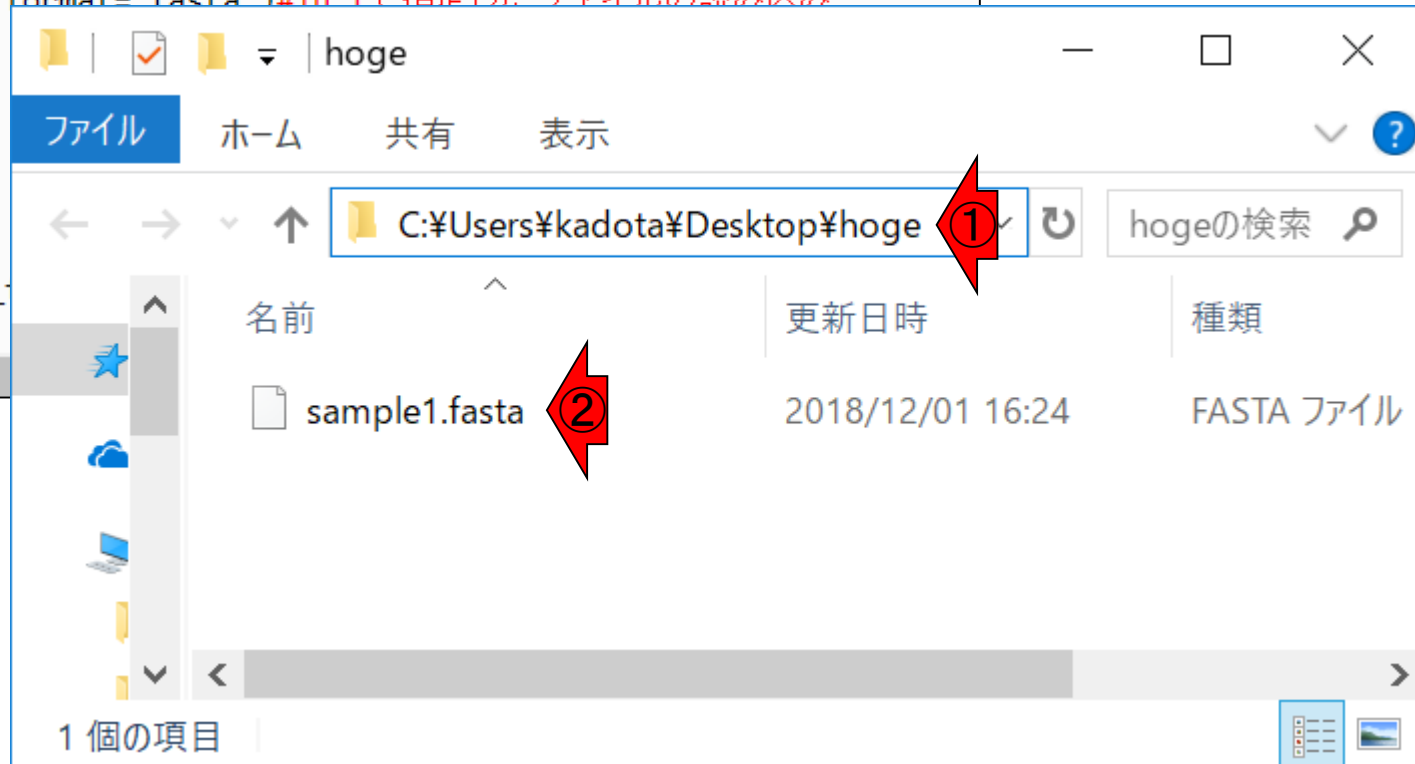
```
#ファイルに保存
```

```
writeXStringSet(fasta, file=out_f)
```

>kadota↓

AGTGACGGTCTT↓

③



目的をおさらい

実際には、プログラム実行結果として、①で指定した名前の翻訳配列を含む出力ファイルが、②hogeフォルダ中に保存されます。

1. FASTA形式ファイル(sample1.fasta)の場合:

multi-FASTAではないsingle-FASTA形式ファイルです。

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



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

```
#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
```

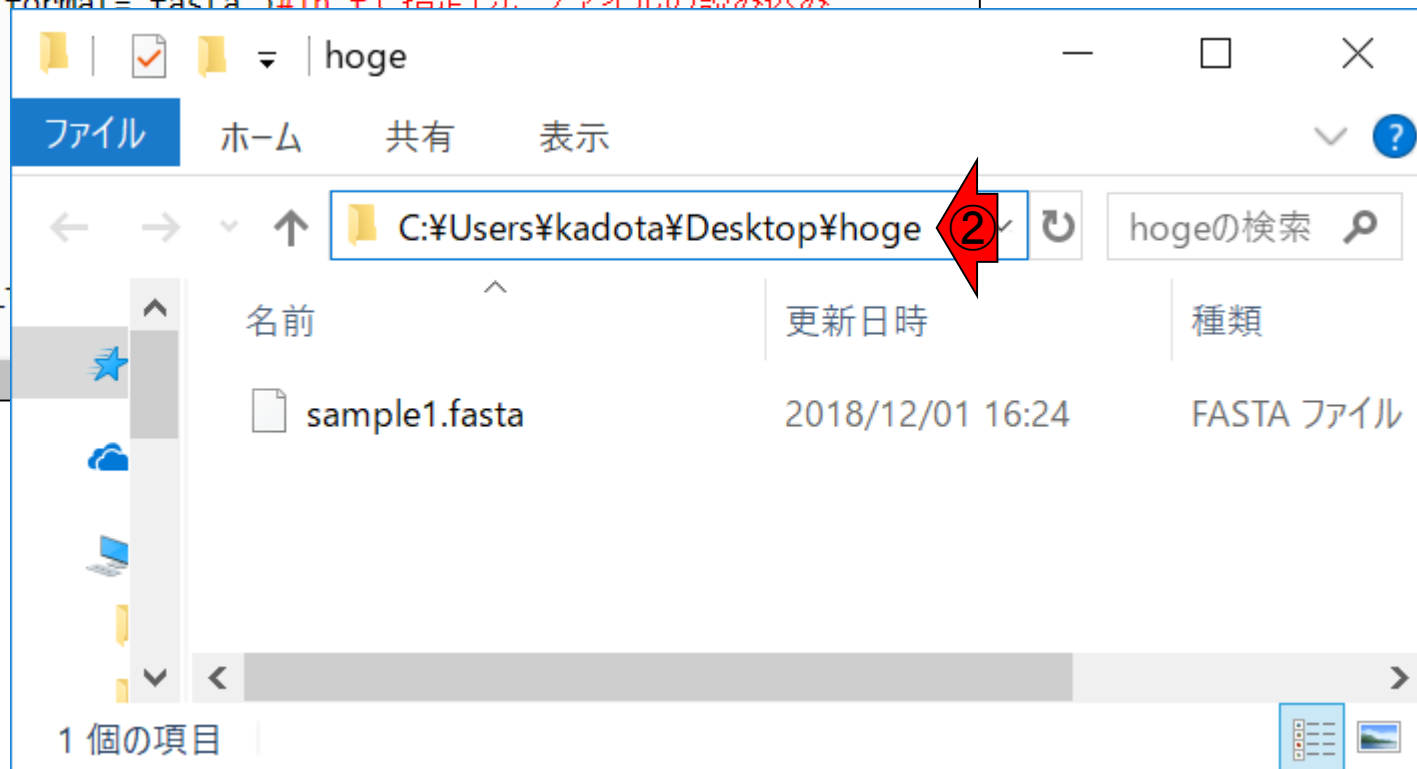
#in_fで指定したファイルの読み込み

```
#本番  
fasta <- translate(fasta)  
fasta
```

```
#ファイルに保存
```

```
writeXStringSet(fasta, file=out_f)
```

>kadota↓
AGTGACGGTCTT↓

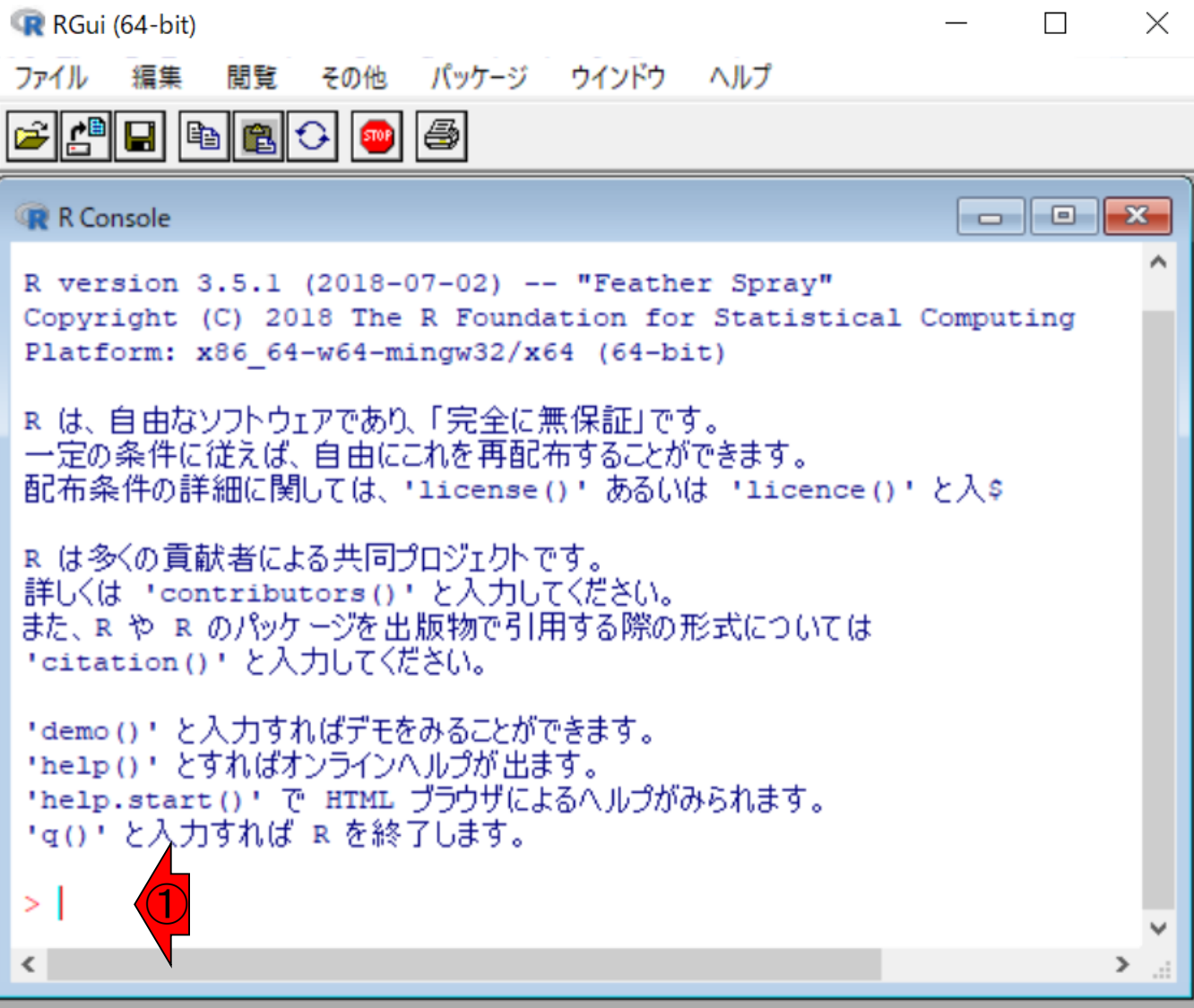


Contents

- R本体とRパッケージの関係
- Rパッケージ: BioconductorとCRAN
- Rの基本的な利用法のおさらい: Biostringsを用いた翻訳配列の取得
 - 入力ファイルのダウンロード、Rの起動を作業ディレクトリの変更
 - コピペ実行、コードの解説
- 複数の例題を実行して理解を深める

①getwd()と打ち込んで、リターンキーを押す。

Rの起動



Rの起動

①このことです。R起動直後のデフォルトの作業ディレクトリは、②ユーザ名kadotaの環境では、「C:/Users/kadota/Documents」です。

```
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$

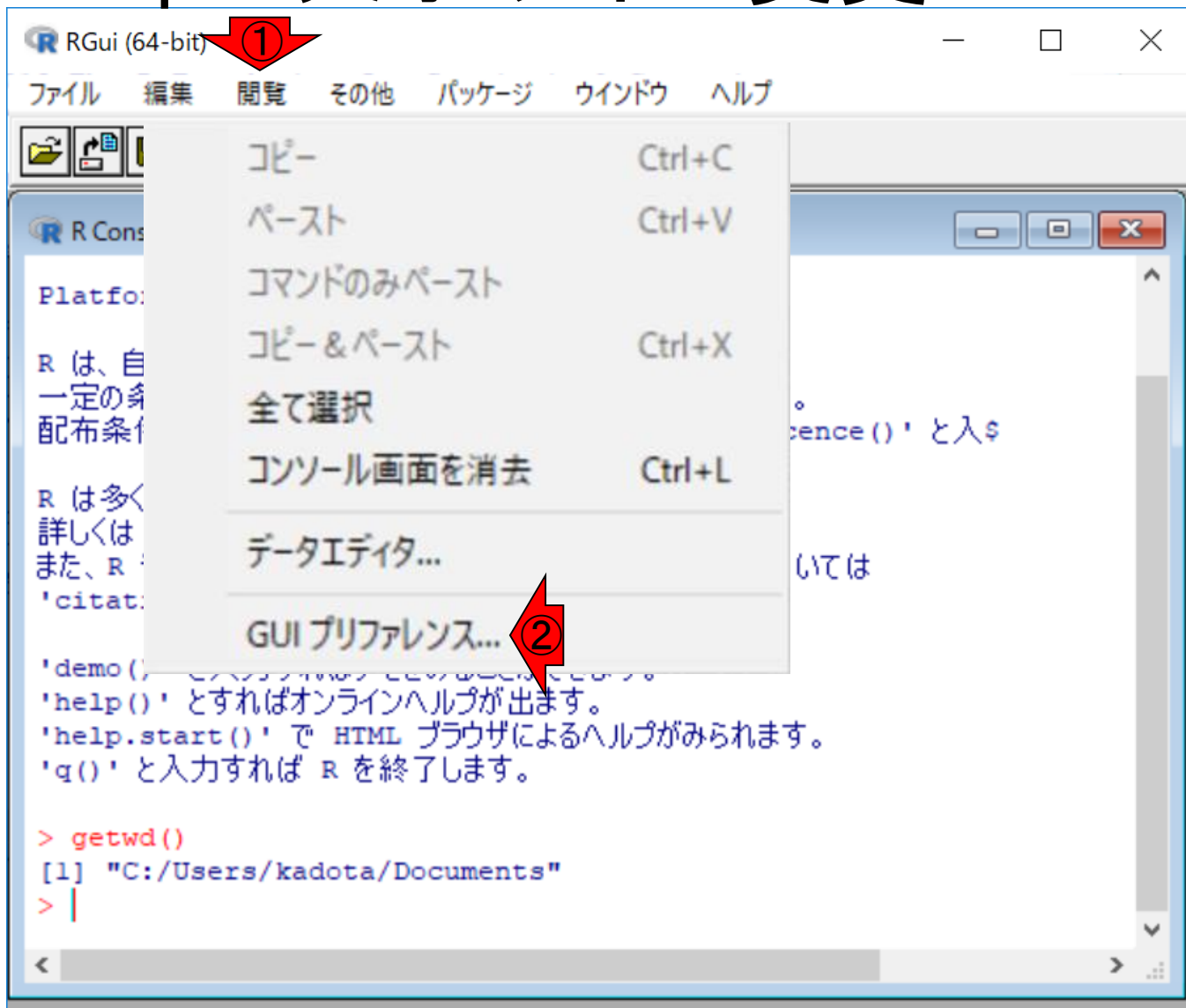
R は多くの貢献者による共同プロジェクトです。
詳しくは 'contributors()' と入力してください。
また、R や R のパッケージを出版物で引用する際の形式については
'citation()' と入力してください。

'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。

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

文字サイズを変更したい場合は、①編集、②GUIプリファレンス

Tips: 文字サイズ変更



Tips: 文字サイズ変更

文字サイズを変更したい場合は、①編集、②GUIプリファレンス、③sizeのところを14とかにしてください。

The image shows the RGui (64-bit) interface. The 'Edit' menu is open, showing options like 'コピー' (Copy), 'ペースト' (Paste), and 'GUIプリファレンス...' (GUI Preferences...). The 'Rgui 設定エディター' (Rgui Settings Editor) dialog box is open, showing various settings. The 'Font' section is highlighted, with the 'size' dropdown set to 10. The 'Console and Pager Colours' section is also visible, with 'background' set to 'normaltext' and 'white' selected in the color list.

① Edit menu

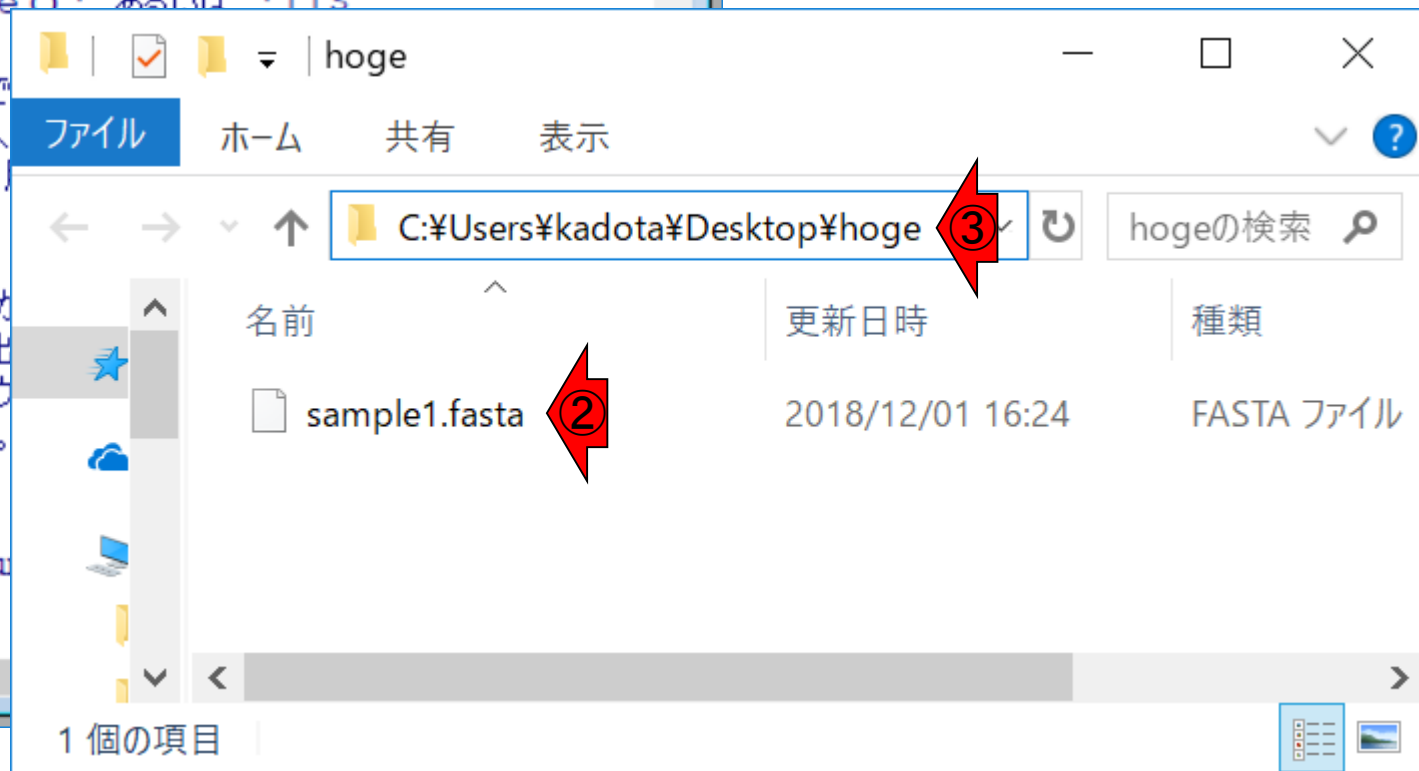
② GUI Preferences

③ Font size dropdown

getwd()

「getwd()」は、現在の作業ディレクトリを表示させるコマンドです。その一方で、②今解析したいファイルは、③デスクトップ上にあるhogeなので、作業ディレクトリをそこに変更する必要があります。

```
RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ
R Console
R は、自由なソフトウェアであり、「完全に無保証」で$
一定の条件に従えば、自由にこれを再配布することがで$
配布条件の詳細に関しては、'license()' または 'lis$
R は多くの貢献者による共同プロジェクトで
詳しくは 'contributors()' と入
また、R や R のパッケージを出版物で引
'citation()' と入力してください。
'demo()' と入力すればデモをみることか
'help()' とすればオンラインヘルプが出
'help.start()' で HTML ブラウ
'q()' と入力すれば R を終了します。
> getwd()
[1] "C:/Users/kadota/Docu
>
```



作業ディレクトリの変更

The screenshot shows the RStudio interface. A red arrow labeled '1' points to the 'ファイル' (File) menu. The 'ファイル' menu is open, and a red arrow labeled '2' points to the 'ディレクトリの変更...' (Change Directory...) option. The background shows a terminal window with the following R code and output:

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

作業ディレクトリの変更

ユーザkadotaの環境ではこのように見えている。デフォルトは①ドキュメントなので、②の位置がハイライトされている。すぐ上の③デスクトップを選択すると…

RGui (64-bit)

ファイル 編集 閲覧 その他

フォルダーの参照

作業ディレクトリの変更
C:\Users\kadota\Documents

- PC
- > 3D オブジェクト
- > ダウンロード
- > デスクトップ
- > **ドキュメント**
- > ピクチャ
- > ビデオ
- > ミュージック
- > Windows (C:)
- > DVD RW ドライブ (D:)
- > SDXC Card (E:)

フォルダー(E): ドキュメント

新しいフォルダーの作成(N) OK キャンセル

R Console

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

R は多くの貢献者による共同
詳しくは 'contributo
また、R や R のパッケージ
'citation()' と入力

'demo()' と入力すれば
'help()' とすればオンラ
'help.start()' で
'q()' と入力すれば R

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

作業ディレクトリの変更

- ①の部分がDesktopに切り替わる。
- ②目的のhogeフォルダを選択

RGui (64-bit)

ファイル 編集 閲覧 その他

R Console

```
R は、自由なソフトウェアであり、  
一定の条件に従えば、自由に  
配布条件の詳細に関しては、  
  
R は多くの貢献者による共同  
詳しくは 'contributor' と入力  
また、R や R のパッケージを  
'citation()' と入力すれば  
  
'demo()' と入力すれば  
'help()' とすればオンライ  
'help.start()' で  
'q()' と入力すれば R  
  
> getwd()  
[1] "C:/Users/kadota/Desktop"  
> |
```

フォルダーの参照

作業ディレクトリの変更
C:\Users\kadota\Desktop ①

- PC
- > 3D オブジェクト
- > ダウンロード
- ▼ デスクトップ ②
 - hoge
- > ドキュメント
- > ピクチャ
- > ビデオ
- > ミュージック
- > Windows (C:)
- > DVD RW ドライブ (D:)

フォルダー(E): デスクトップ

新しいフォルダーの作成(N) OK キャンセル

作業ディレクトリの変更

RGui (64-bit)

ファイル 編集 閲覧 その他

R Console

R は、自由なソフトウェアであり、
一定の条件に従えば、自由に
配布条件の詳細に関しては、
R は多くの貢献者による共同
詳しくは 'contributor'。
また、R や R のパッケージを
'citation()' と入力
'demo()' と入力すれば
'help()' とすればオンラ
'help.start()' で
'q()' と入力すれば R

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

フォルダーの参照

作業ディレクトリの変更
C:\Users\kadota\Desktop\hoge

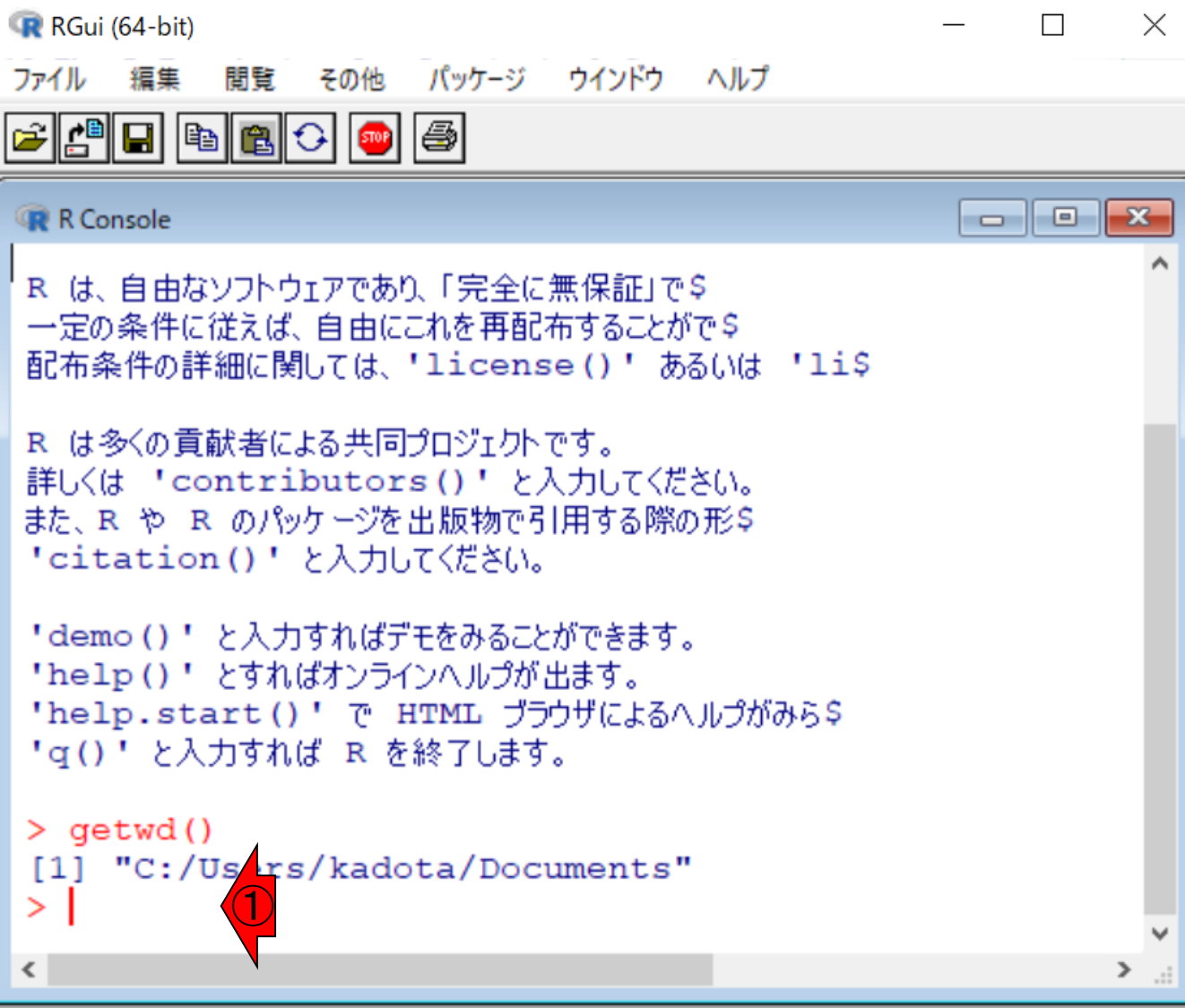
- PC
 - > 3D オブジェクト
 - > ダウンロード
 - ▼ デスクトップ
 - hoge
 - > ドキュメント
 - > ピクチャ
 - > ビデオ
 - > ミュージック
 - > Windows (C:)
 - > DVD RW ドライブ (D:)

フォルダー(E): hoge

新しいフォルダーの作成(N) OK キャンセル

もう一度getwd()

一見すると、何も変わってなさそうですが、①
もう一度`getwd()`を実行すれば、作業ディレクトリが変更されていることが確認できます。



```
RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ
[Icons: File Explorer, Print, Save, Copy, Paste, Refresh, Stop, Print]

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

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$
'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$
'q()' と入力すれば R を終了します。

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

もう一度getwd()

さきほどと同様に`getwd()`とベタ打ちしてもよいが、キーボードの②上矢印キーを一回押すと、直前に打ち込んだコマンド(この場合は`getwd()`)が表示される。これは打ち込んだのと同じ意味なので、そのままリターンキーを押せばよい。いくつか入力したコマンドがあれば、上矢印キーを押していけば見られます。行き過ぎたら下矢印キーを押していけば戻れます

RGui (64-bit)

ファイル 編集 閲覧 その他 パッケージ ウィンドウ



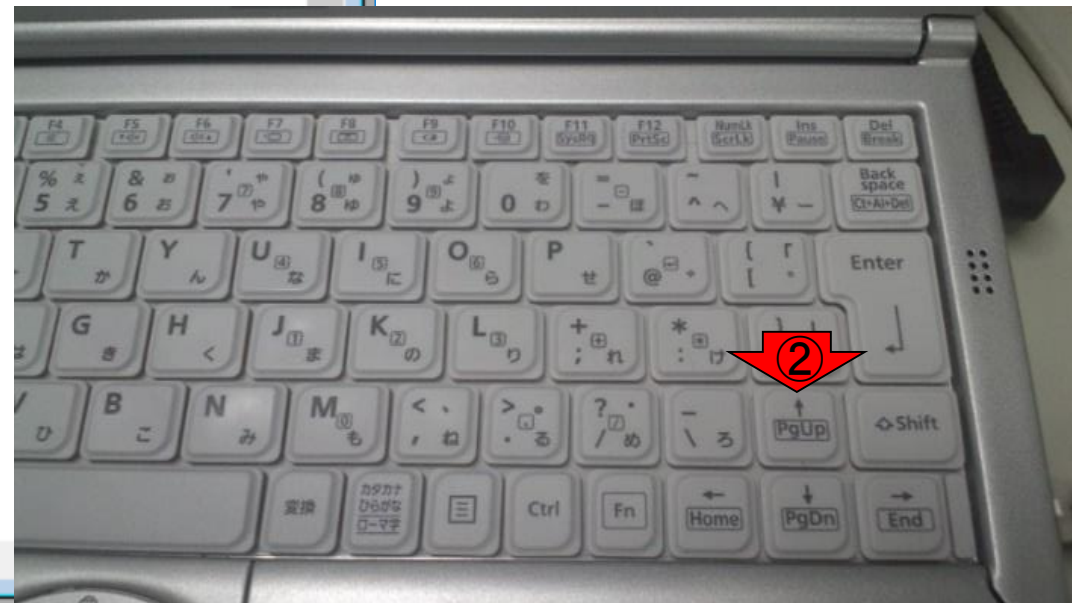
R Console

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

```
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$  
'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$  
'q()' と入力すれば R を終了します。
```

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



確認

①こんな感じで、作業ディレクトリが「.../Desktop/hoge」であればOK。当たり前ですが、解析したいディレクトリ(またはフォルダ)を正しく指定できていなければエラーに遭遇します。また、解析したいファイルが存在しない状態でもエラーが出ます。

RGui (64-bit)

ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ



R Console

配布条件の詳細に関しては、`'license()'` あるいは `'li$`

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\$

`'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\$

`'q()'` と入力すれば R を終了します。

```
> getwd()
```

```
[1] "C:/Users/kadota/Documents"
```

```
> getwd()
```

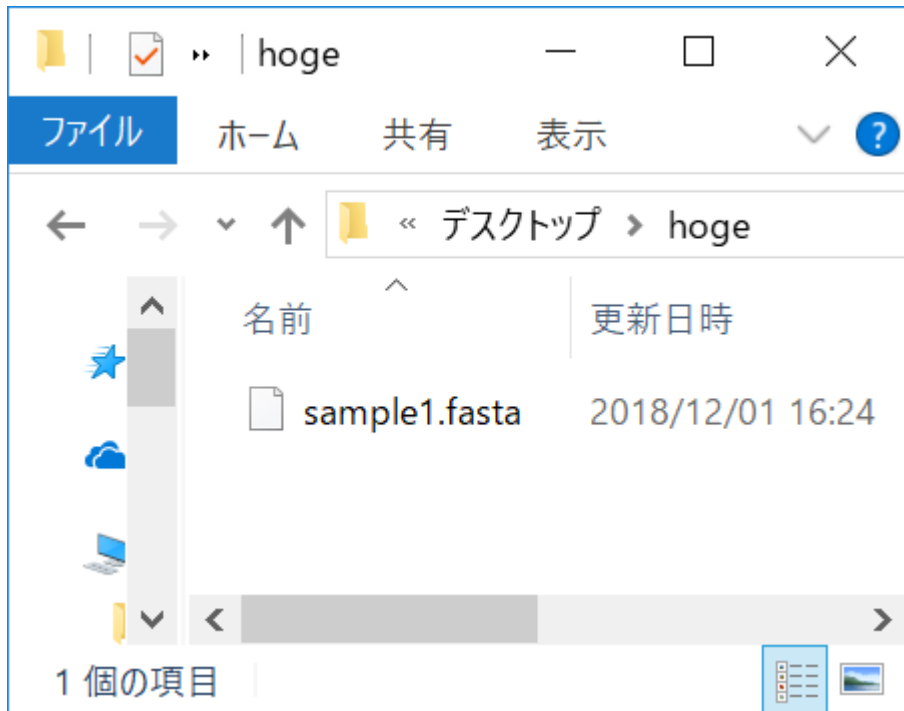
```
[1] "C:/Users/kadota/Desktop/hoge"
```

```
> |
```



①list.files()は、作業ディレクトリの中身を表示するコマンドです。

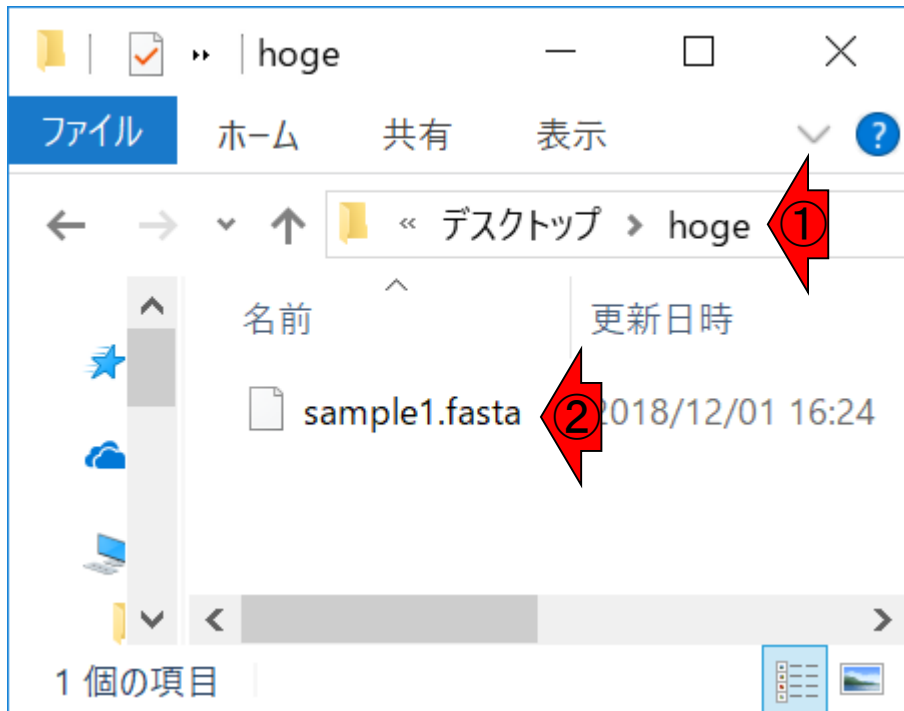
list.files()でフォルダ内を見る



```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
① list.files()
[1] "sample1.fasta"
> |
```

①hogeフォルダの②中身が対応しているのがわかりますね。

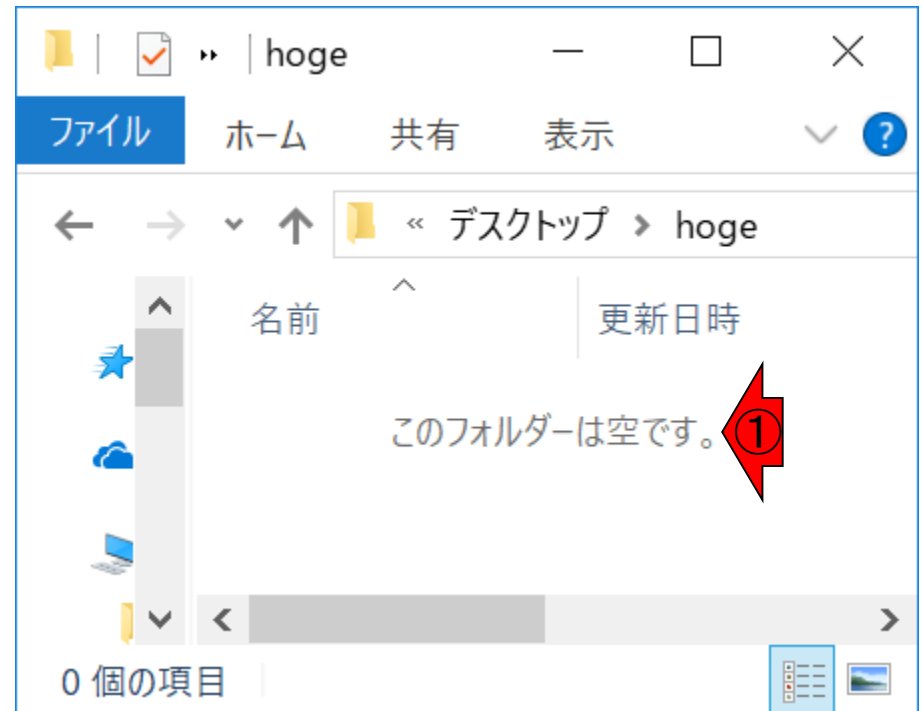
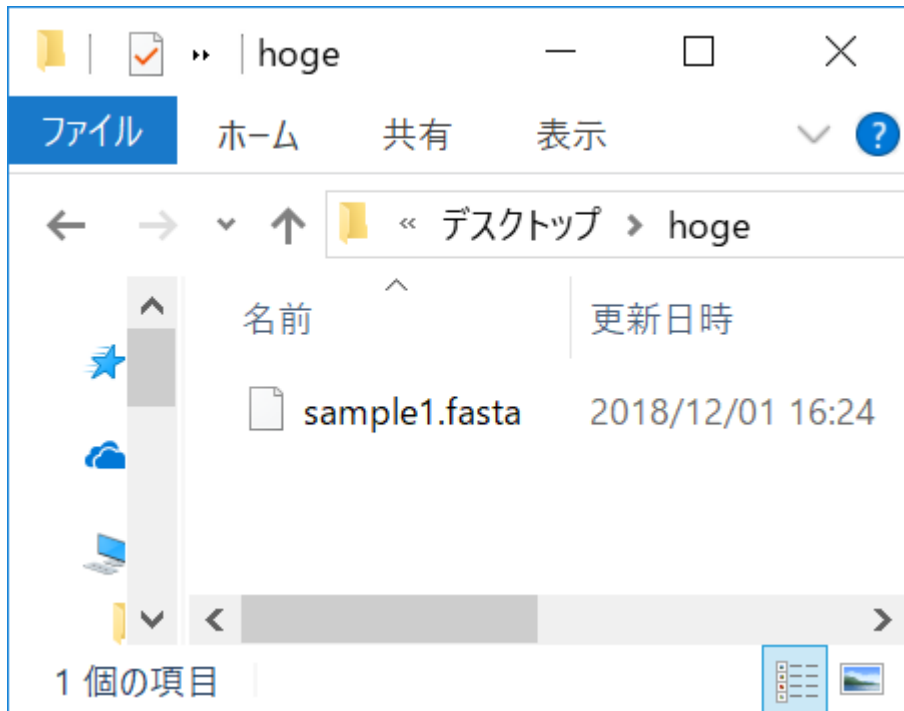
list.files()でフォルダ内を見る



```
R Console  
> getwd()  
[1] "C:/Users/kadota/Desktop/hoge"  
> list.files()  
[1] "sample1.fasta"  
> |
```

list.files()でフォルダ内を見

もし①フォルダの中身が何もないときは、②のように見えるので覚えておきましょう。character(0)は何もないという意味



```
R Console
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files()
[1] "sample1.fasta"
> |
```

```
R Console
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files()
character(0)
> |
```

Contents

- R本体とRパッケージの関係
- Rパッケージ: BioconductorとCRAN
- Rの基本的な利用法のおさらい: Biostringsを用いた翻訳配列の取得
 - 入力ファイルのダウンロード、Rの起動を作業ディレクトリの変更
 - コピペ実行、コードの解説
- 複数の例題を実行して理解を深める

基本はコピー

- ①一連のコマンド群をコピーして
- ②R Console画面上でペースト。

イントロ | 一般 | 翻訳配列(translate)を取得(基礎) | Biostrings

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。翻訳のための遺伝コード(genetic code)は、Standard Genetic Codeだそうです。もちろん生物種?!によって多少違い(variants)があるようで、"Standard", "SGC0", "Vertebrate Mitochondrial", "SGC1"などいろいろ選べるようです。
「ファイル」 - 「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピー。

1. FASTA形式ファイル(sample1.fasta)の場合：

multi-FASTAではないsingle-FASTA形式ファイルです。

```
in f <- "sample1.fasta" #入力ファイル名を指定してin fに格納
out f <- "hoge1.fasta" #出力ファイル名を指定してout fに格納

#必要なパッケージをロード
library(Biostrings)

#入力ファイルの読み込み
fasta <- readDNASTringSet("sample1.fasta")
fasta

#本番
fasta <- translate(fasta) #アミノ酸配列に翻訳
fasta #確認して

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

コピー(C)

①

R Console

Platform: x86_64-w64-mingw32/x64 (64-bit)

R は、自由なソフトウェアであり、「完全な条件に従えば、自由にこれを再配布条件の詳細に関しては、'license()' 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に引用する場合は 'citation()' と入力してください。

'demo()' と入力すればデモをみる。'help()' とすればオンラインヘルプを見ます。'help.start()' で HTML ブラウザによるヘルプを見ます。'q()' と入力すれば R を終了します。

> |

コピー

Ctrl+C

ペースト

Ctrl+V

コマンドのみペースト

コピー & ペースト

Ctrl+X

ウィンドウの消去

Ctrl+L

全て選択

バッファに出力

Ctrl+W

ウィンドウを常にトップに置く

基本はコピペ

イントロ | 一般 | 翻訳配列(translate)を取得

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するコード(genetic code)は、Standard Genetic Codeだそうです。もちろん、"Standard", "SGC0", "Vertebrate Mitochondrial", "SGC1"など、他の遺伝コードも利用可能です。「ファイル」 - 「ディレクトリの変更」で解析したいファイルを置いてください。

1. FASTA形式ファイル(sample1.fasta)の場合:

multi-FASTAではないsingle-FASTA形式ファイルです。

```
in_f <- "sample1.fasta" #入力ファイル名を
out_f <- "hoge1.fasta" #出力ファイル名を

#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

#本番
fasta <- translate(fasta) #アミノ酸配列に翻訳
fasta #確認してるだけで

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

R Console

```
> in_f <- "sample1.fasta"
> out_f <- "hoge1.fasta"
>
> #必要なパッケージをロード
> library(Biostrings)
要求されたパッケージ BiocGenerics をロード中です
要求されたパッケージ parallel をロード中です

次のパッケージを付け加えます: 'BiocGenerics'

The following objects are masked from 'package:parallel':

clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
clusterExport, clusterMap, parApply, parCapply, parLapply,
parLapplyLB, parRapply, parSapply, parSapplyLB

The following object is masked from 'package:stats':

xtabs

The following objects are masked from 'package:base':

anyDuplicated, append, as.data.frame, as.vector, cbind,
colnames, do.call, duplicated, eval, evalq, Filter, Find, get,
intersect, is.unsorted, lapply, Map, mapply, match, mget, order,
paste, pmax, pmax.int, pmin, pmin.int, Position, rank, rbind,
Reduce, rep.int, rownames, sapply, setdiff, sort, table, tapply,
union, unique, unlist

要求されたパッケージ IRanges をロード中です
要求されたパッケージ XVector をロード中です
>
> #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
  A DNAStringSet instance of length 1
    width seq          names
[1]    12 AGTGACGGTCTT kadota
>
> #本番
> fasta <- translate(fasta) #アミノ酸配列に翻訳した結果をfasta$
> fasta #確認してるだけで
  A AAStringSet instance of length 1
    width seq          names
[1]     4 SDGL          kadota
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=50) #fastaの中身を$
> |
```

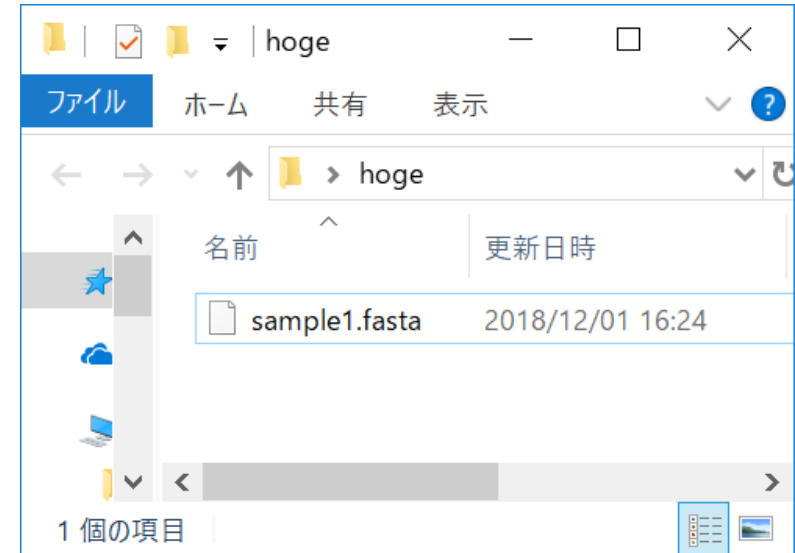
エラーなく実行できた場合の全貌。多少見栄えが異なっても、エラーという文字が見えていなければOK。

実行結果

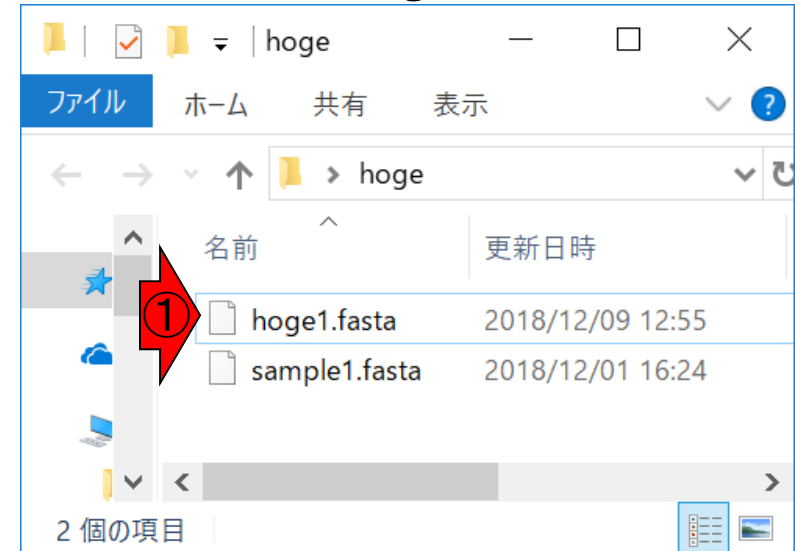
①出力ファイル名として指定したhoge1.fastaが生成されていることが分かります

```
R Console
> fasta <- readDNAStringSet(in_f, format="fasta")$
> fasta                                     #確認し$
A DNAStringSet instance of length 1
  width seq          names $
[1]    12 AGTGACGGTCTT kadota
>
> #本番
> fasta <- translate(fasta)                #アミノ$
> fasta                                     #確認し$
A AAStringSet instance of length 1
  width seq          names $
[1]     4 SDGL        kadota
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fast$
> |
```

実行前のhogeフォルダ



実行後のhogeフォルダ

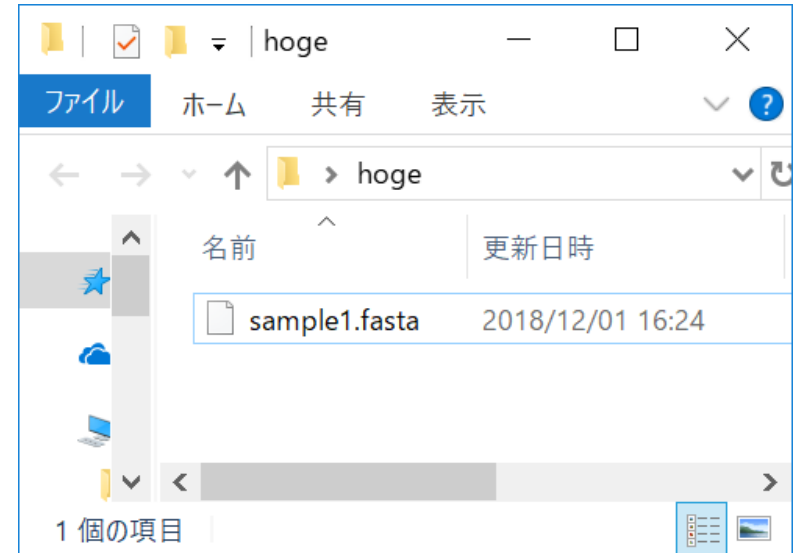


実行結果

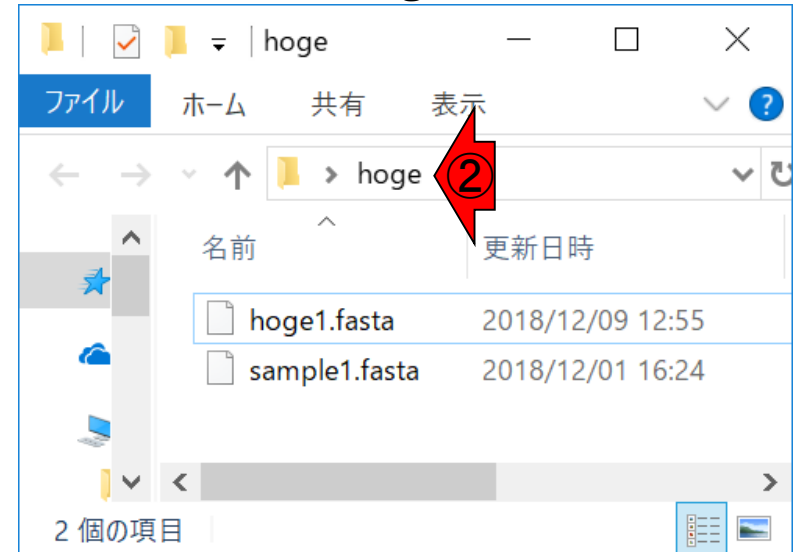
①list.files()で表示される結果と、②「実行後のhogeフォルダの中身は当然同じ

```
R Console
> fasta <- readDNASTringSet(in_f, format="fasta")$
> fasta #確認し$
A DNASTringSet instance of length 1
width seq names
[1] 12 AGTGACGGTCTT kadota
>
> #本番
> fasta <- translate(fasta) #アミノ$
> fasta #確認し$
A AAStringSet instance of length 1
width seq names
[1] 4 SDGL kadota
>
> #ファイルに保存
> writeXString(fasta, file=out_f, format="fast$
> list.files()
[1] "hoge1.fasta" "sample1.fasta"
> |
```

実行前のhogeフォルダ



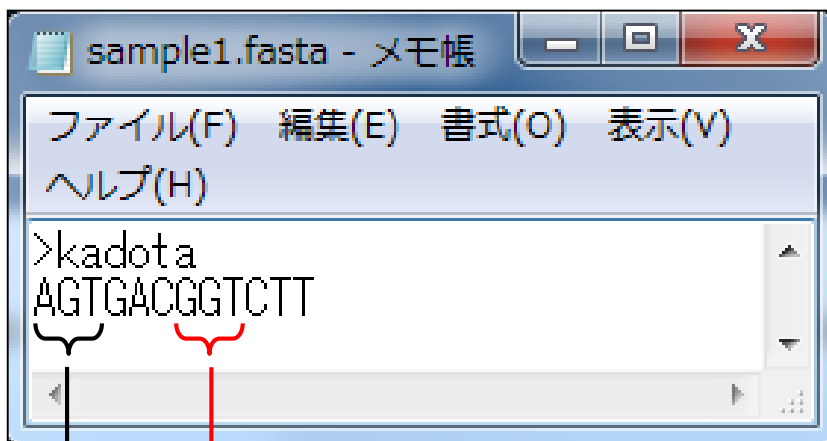
実行後のhogeフォルダ



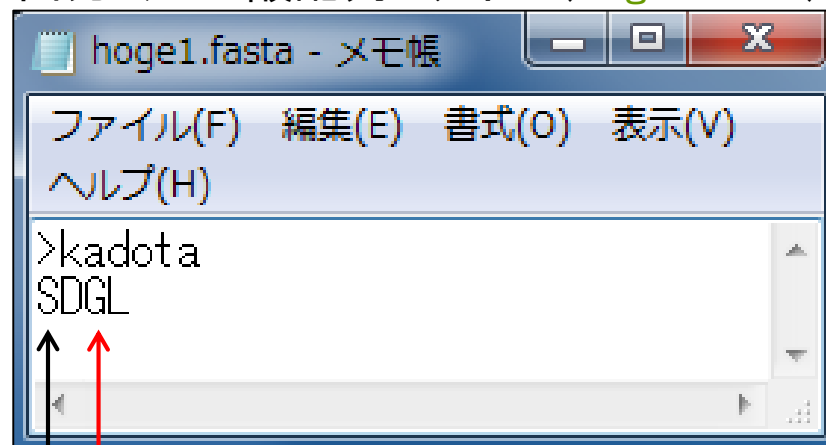
実行結果

入力ファイル中の塩基配列は、3の倍数の12塩基長、ACGTのみからなるので何のエラーもない

入力: 塩基配列ファイル (sample1.fasta)



出力: アミノ酸配列ファイル (hoge1.fasta)



Contents

- R本体とRパッケージの関係
- Rパッケージ: BioconductorとCRAN
- Rの基本的な利用法のおさらい: Biostringsを用いた翻訳配列の取得
 - 入力ファイルのダウンロード、Rの起動を作業ディレクトリの変更
 - コピペ実行、コードの解説
- 複数の例題を実行して理解を深める

コードの解説

イントロ | 一般 | 翻訳配列(translate)を取得(基礎)

一通り入出力の関係が分かったら、中でどのようなことが行われているのかが気になります。それを解説していきます。まず、①の行は、入力ファイルの名前をin_fという名前を取り扱うという宣言です。

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。翻訳のための遺伝コード(genetic code)は、Standard Genetic Codeだそうです。もちろん生物種?!によって多少違い(variants)があるようで、"Standard", "SGC0", "Vertebrate Mitochondrial", "SGC1"などいろいろ選べるようです。

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

1. FASTA形式ファイル(sample1.fasta)の場合：

multi-FASTAではないsingle-FASTA形式ファイルです。

```
in_f <- "sample1.fasta" #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.fasta" #出力ファイル名を指定してout_fに格納

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

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

#本番
fasta <- translate(fasta) #アミノ酸配列に翻訳した結果をfastaに格納
fasta #確認してるだけです

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



コードの解説

イントロ | 一般 | 翻訳配列(translate)を取得(基礎)

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するコード(genetic code)は、Standard Genetic Codeだそうです。もちろん「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

1. FASTA形式ファイル(sample1.fasta)の場合:

multi-FASTAではないsingle-FASTA形式ファイルです。

```
in_f <- "sample1.fasta" #入力ファイル名を指定し
out_f <- "hoge1.fasta"  #出力ファイル名を指定し

#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
fasta                                         #確認してるだけです

#本番
fasta <- translate(fasta) #アミノ酸配列に翻訳し
fasta                                         #確認してるだけです

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

一通り入出力の関係が分かったら、中でどのようなことが行われているのかが気になります。それを解説していきます。まず、①の行は、入力ファイルの名前をin_fという名前を取り扱うという宣言です。なので、②R Console画面上で、③in_fと打ち込んでリターンキーを押すと、①と同じ文字列が見られます

```
R Console
> 
> #本番
> fasta <- translate(fasta) #ア$
> fasta #確$
A AStringSet instance of length 1
  width seq      names
[1]    4 SDGL      kadota
> 
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="$
> list.files()
[1] "hoge1.fasta" "sample1.fasta"
> in_f
[1] "sample1.fasta"
> |
```


コードの解説

イントロ | 一般 | 翻訳配列(translate)を取得(基礎)

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するコード(genetic code)は、Standard Genetic Codeだそうです。もちろん「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

1. FASTA形式ファイル(sample1.fasta)の場合：

multi-FASTAではないsingle-FASTA形式ファイルです。

```
in_f <- "sample1.fasta" #入力ファイル名を指定し  
out_f <- "hoge1.fasta" #出力ファイル名を指定し
```

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

```
#入力  
fasta <- readAAStringSet(in_f)
```

```
#本番  
fasta <- translate(fasta)
```

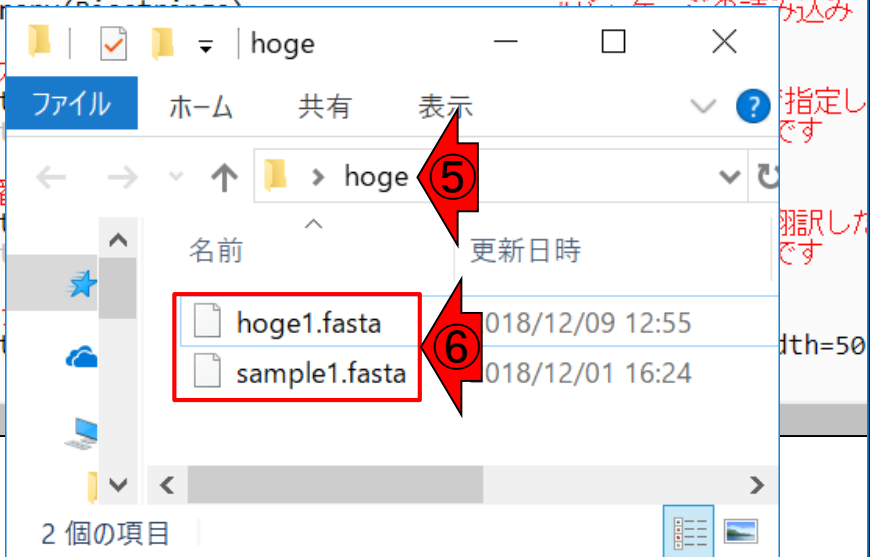
```
#ファイルに保存  
writeXStringSet(fasta, file=out_f, format="$seq $names")
```

```
#ファイルを確認  
list.files()
```

```
in_f  
[1] "sample1.fasta"
```

一通り入出力の関係が分かったら、中でどのようなことが行われているのかが気になります。それを解説していきます。まず、①の行は、入力ファイルの名前をin_fという名前を取り扱うという宣言です。なので、②R Console画面上で、③in_fと打ち込んでリターンキーを押すと、①と同じ文字列が見られます。ちなみに④は、コピー実行後の⑤hogeフォルダの⑥中身を表示しているだけなので意味合いが異なります。

```
>  
> #本番  
> fasta <- translate(fasta) #ア$  
> fasta #確$  
A AAStringSet instance of length 1  
width seq names  
[1] 4 SDGL kadota  
>  
> #ファイルに保存  
> writeXStringSet(fasta, file=out_f, format="$seq $names")  
> list.files() #④  
[1] "hogel.fasta" "sample1.fasta"  
> in_f  
[1] "sample1.fasta"  
> |
```



コードの解説

イントロ | 一般 | 翻訳配列(translate)を取得(基礎)

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するコード(genetic code)は、Standard Genetic Codeだそうです。もちろん「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

1. FASTA形式ファイル(sample1.fasta)の場合:

① FASTAではないsingle-FASTA形式ファイルです。

```
in_f <- "sample1.fasta" #入力ファイル名を指定し
out_f <- "hoge1.fasta" #出力ファイル名を指定し

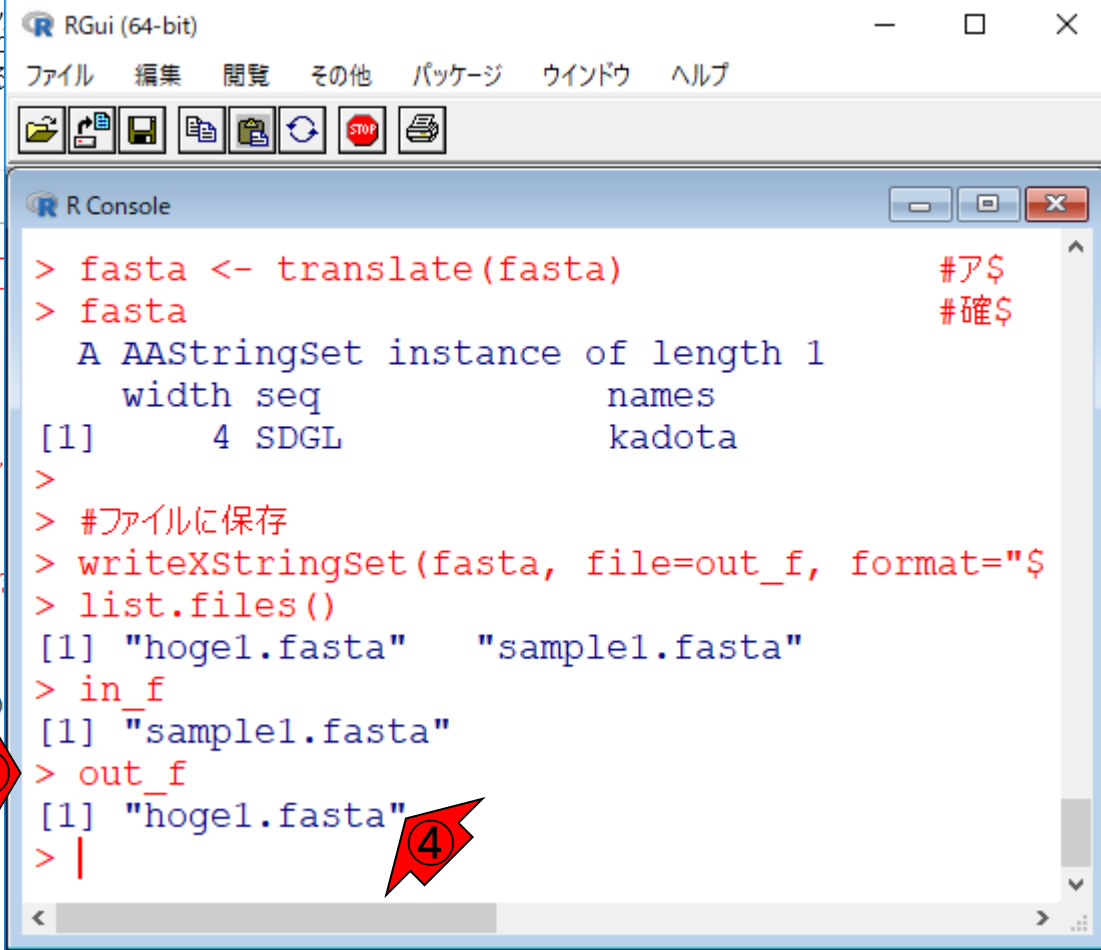
#必要なパッケージをロード
library(Biostrings) #パッケージの読み込み

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

#本番
fasta <- translate(fasta) #アミノ酸配列に翻訳し
fasta #確認してるだけです

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

①in_fと同じノリで、②の行は、出力ファイル名をout_fとして取り扱うという宣言です。③確かにhoge1.fastaが表示されていますね。ちなみに④二重クォーテーション(“”)は、中身が文字列という意味です。



```
RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ

R Console
> fasta <- translate(fasta) #ア$
> fasta #確$
A AAStringSet instance of length 1
width seq names
[1] 4 SDGL kadota
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="$
> list.files()
[1] "hoge1.fasta" "sample1.fasta"
> in_f
[1] "sample1.fasta"
> out_f
[1] "hoge1.fasta"
> |
```

コードの解説

イントロ | 一般 | 翻訳配列(translate)を取得(基礎)

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するコード(genetic code)は、Standard Genetic Codeだそうです。もちろん「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

1. FASTA形式ファイル(sample1.fasta)の場合:

multi-FASTAではないsingle-FASTA形式ファイルです。

```
in_f <- "sample1.fasta"      #入力ファイル名を指定し
out_f <- "hoge1.fasta"       #出力ファイル名を指定し

#必要なパッケージをロー
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
fasta                                     #確認してるだけです

#本番
fasta <- translate(fasta)     #アミノ酸配列に翻訳し
fasta                         #確認してるだけです

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



①は、Biostringsというパッケージを使うよ、という宣言です。「Biostringsパッケージのロード」とか、「パッケージの読み込み」などと表現します。

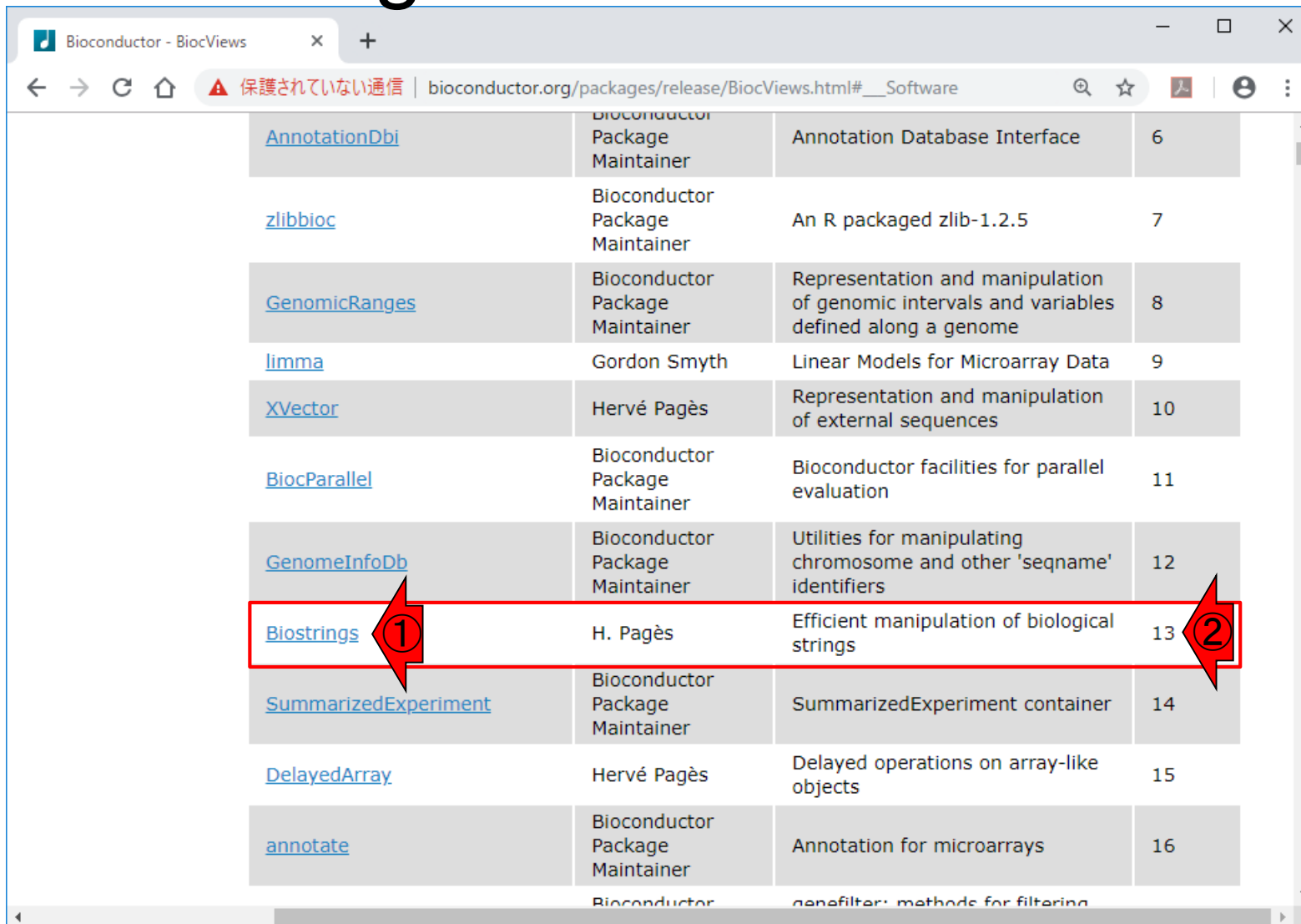
PowerPointを使うためにはソフトを起動しないといけないようなものだと思えばよいです。

```
RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ

R Console
> fasta <- translate(fasta)      #ア$
> fasta                          #確$
A AAStringSet instance of length 1
  width seq          names
[1]     4 SDGL      kadota
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="$
> list.files()
[1] "hoge1.fasta"  "sample1.fasta"
> in_f
[1] "sample1.fasta"
> out_f
[1] "hoge1.fasta"
> |
```

Biostrings

おさらい。①Biostringsは、Bioconductorが提供する②13位のパッケージで、結構有名です。



Package Name	Maintainer	Description	Rank
AnnotationDbi	Bioconductor Package Maintainer	Annotation Database Interface	6
zlibbioc	Bioconductor Package Maintainer	An R packaged zlib-1.2.5	7
GenomicRanges	Bioconductor Package Maintainer	Representation and manipulation of genomic intervals and variables defined along a genome	8
limma	Gordon Smyth	Linear Models for Microarray Data	9
XVector	Hervé Pagès	Representation and manipulation of external sequences	10
BiocParallel	Bioconductor Package Maintainer	Bioconductor facilities for parallel evaluation	11
GenomeInfoDb	Bioconductor Package Maintainer	Utilities for manipulating chromosome and other 'seqname' identifiers	12
Biostrings	H. Pagès	Efficient manipulation of biological strings	13
SummarizedExperiment	Bioconductor Package Maintainer	SummarizedExperiment container	14
DelayedArray	Hervé Pagès	Delayed operations on array-like objects	15
annotate	Bioconductor Package Maintainer	Annotation for microarrays	16
genefilter	Bioconductor Package Maintainer	genefilter: methods for filtering	17

①Biostringsパッケージをロードしたおかげで、②これらの関数が利用可能となるのです。

コードの解説

イントロ | 一般 | 翻訳配列(translate)を取得(基礎) | Biostrings

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。翻訳のための**遺伝コード(genetic code)**は、Standard Genetic Codeだそうです。もちろん生物種?!によって多少違い(variants)があるようで、"Standard", "SGC0", "Vertebrate Mitochondrial", "SGC1"などいろいろ選べるようです。

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

1. FASTA形式ファイル(sample1.fasta)の場合：

multi-FASTAではないsingle-FASTA形式ファイルです。

```
in_f <- "sample1.fasta"      #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.fasta"      #出力ファイル名を指定してout_fに格納

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

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

#本番
fasta <- translate(fasta)    #アミノ酸配列に翻訳した結果をfastaに格納
fasta                       #確認してるだけです

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

コードの解説

イントロ | 一般 | 翻訳配列(translate)を取得

①readDNAStringSetという名前の関数は、②の入力ファイル名に相当するin_fを、③の部分で利用しています。つまりreadDNAStringSetは、readという名前からイメージできるように、DNA塩基配列を読み込むときに用います。

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。翻訳のための遺伝コード(genetic code)は、Standard Genetic Codeだそうです。もちろん生物種?!によって多少違い(variants)があるようで、"Standard", "SGC0", "Vertebrate Mitochondrial", "SGC1"などいろいろ選べるようです。

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

1. FASTA形式ファイル(sample1.fasta)の場合：

multi-FASTAではないsingle-FASTA形式ファイルです。

```
in_f <- "sample1.fasta" #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.fasta" #出力ファイル名を指定してout_fに格納

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

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

#本番
fasta <- translate(fasta) #アミノ酸配列に翻訳した結果をfastaに格納
fasta #確認してるだけです

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

コードの解説

イントロ | 一般 | 翻訳配列(translate)

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列(遺伝子配列)を生成する。遺伝子配列(genetic code)は、Standard Genetic Codeだそうである。Standard, "SGC0", "Vertebrate Mitochondrial"などがある。RStudioの「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。

④は、③の入力ファイルの中身がfasta形式だと明示的に指定していることに相当します。本当はfasta形式がデフォルトなので書かなくてもよいのですが、書いてもよいので書いています。②の中身が⑤であり、これがfasta形式(or FASTA形式)というものです。FASTQ形式というNGS生データファイルも、④の部分をformat="fastq"とすることで読み込むことができます。

1. FASTA形式ファイル(sample1.fasta)の場合：

multi-FASTAではないsingle-FASTA形式ファイルです。

```
in_f <- "sample1.fasta" #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.fasta"  #出力ファイル名を指定してout_fに格納

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

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

#本番
fasta <- translate(fasta) #アミノ酸配列に翻訳した結果をfastaに格納
fasta                       #確認してるだけです

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

```
>kadota↓
AGTGACGGTCTT↓
```

コードの解説

①赤枠内をもう一度コピー実行し、②readDNAStringSet関数を用いて読み込んだ結果を格納したfastaという名前のオブジェクトを表示させます。

イントロ | 一般 | 翻訳配列(translate)を取得(基礎) | Biostrings

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。翻訳のための遺伝コード(genetic code)は、Standard Genetic Codeだそうです。もちろん生物種?!によって多少違い(variants)があるようで、"Standard", "SGC0", "Vertebrate Mitochondrial", "SGC1"などいろいろ選べるようです。「ファイル」 - 「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピー。

1. FASTA形式ファイル(sample1.fasta)の場合：

multi-FASTAではないsingle-FASTA形式ファイルです。

```
in_f <- "sample1.fasta"      #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.fasta"      #出力ファイル名を指定してout_fに格納

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

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

#本番
fasta <- translate(fasta)    #アミノ酸配列に翻訳した結果をfastaに格納
fasta                       #確認してるだけです

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



コードの解説

①赤枠内をもう一度コピー実行し、②readDNAStringSet関数を用いて読み込んだ結果を格納したfastaという名前のオブジェクトを表示させます。表示結果。

イントロ | 一般 | 翻訳配列(translate)を取得(基礎) | Biostrings

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。翻訳のための遺伝コード(genetic code)は、Standard Genetic Codeだそうです。もちろん「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

1. FASTA形式ファイル(sample1.fasta)の場合：

multi-FASTAではないsingle-FASTA形式ファイルです。

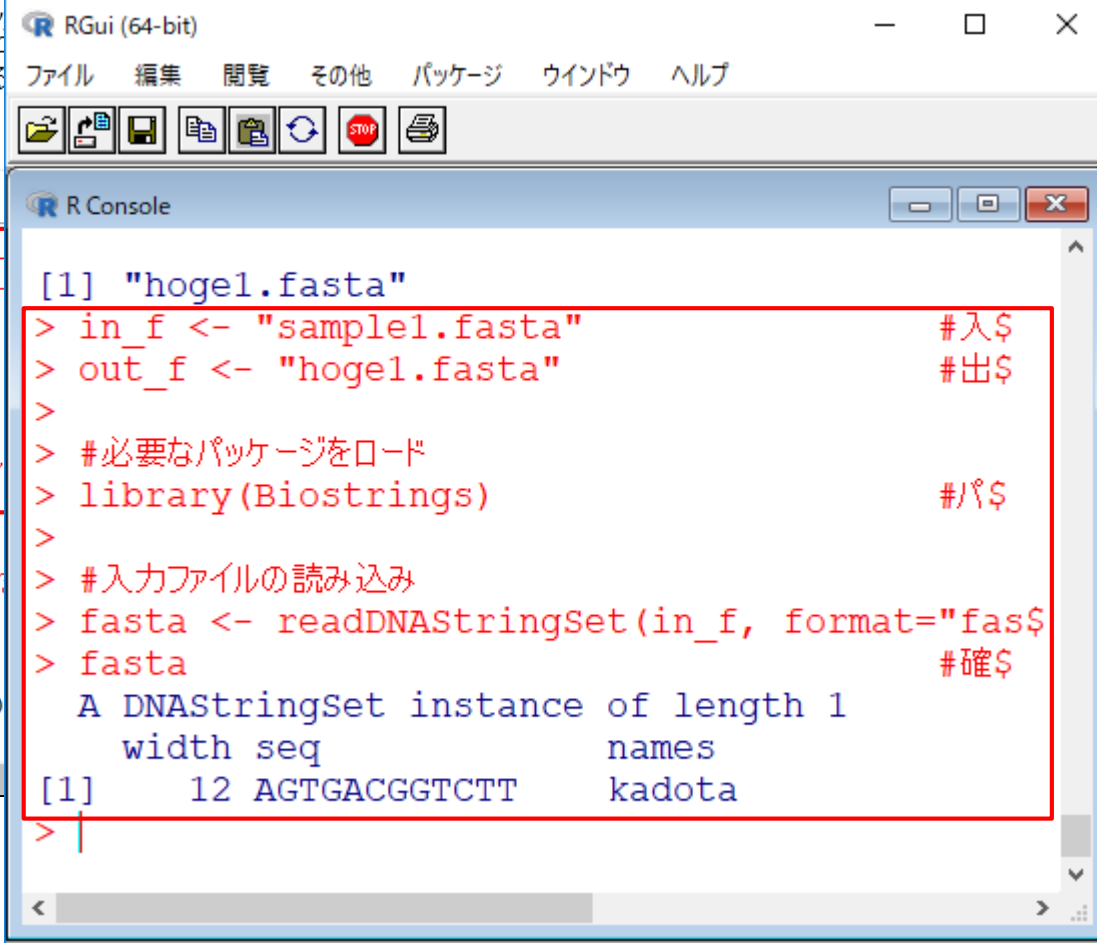
```
in_f <- "sample1.fasta" #入力ファイル名を指定し
out_f <- "hoge1.fasta"  #出力ファイル名を指定し

#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
fasta                                           #確認してるだけです

#本番
fasta <- translate(fasta) #アミノ酸配列に翻訳し
fasta                       #確認してるだけです

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



```
RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ

R Console
[1] "hoge1.fasta"
> in_f <- "sample1.fasta" #入$
> out_f <- "hoge1.fasta"  #出$
>
> #必要なパッケージをロード
> library(Biostrings)     #パ$
>
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fas$
> fasta                   #確$
A DNAStringSet instance of length 1
width seq names
[1] 12 AGTGACGGTCTT kadota
> |
```

- ①fastaというオブジェクト(←「もの」という意味)の中身は、
- ②こんな感じ。

コードの解説

イントロ | 一般 | 翻訳配列(translate)を取得(基礎) | Biostrings

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。翻訳のための遺伝コード(genetic code)は、Standard Genetic Codeだそうです。もちろん「Standard」、「SGC0」、「Vertebrate Mitochondrial」、「SGC1」など「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

1. FASTA形式ファイル(sample1.fasta)の場合：

multi-FASTAではないsingle-FASTA形式ファイルです。

```
in_f <- "sample1.fasta"      #入力ファイル名を指定し
out_f <- "hoge1.fasta"       #出力ファイル名を指定し

#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
fasta                               #確認してるだけです

#本番
fasta <- translate(fasta)     #アミノ酸配列に翻訳し
fasta                               #確認してるだけです

#ファイルに保存
writeXStringSet(fasta, file=out_f, format="fasta", width=100)
```

```
RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ

R Console

[1] "hoge1.fasta"
> in_f <- "sample1.fasta"      #入$
> out_f <- "hoge1.fasta"      #出$
>
> #必要なパッケージをロード
> library(Biostrings)        #パ$
>
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fas$
> fasta                       #確$

A DNAStringSet instance of length 1
width seq          names
[1] 12 AGTGACGGTCTT kadota
```

コードの解説

①fastaというオブジェクト(←「もの」という意味)の中身は、
②こんな感じ。③入力ファイルの、④中身とよく対応しているのがわかりますね。

イントロ | 一般 | 翻訳配列(translate)を取得(基礎) | Biostrings

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。翻訳のための遺伝コード(genetic code)は、Standard Genetic Codeだそうです。もちろん「Standard」、「SGC0」、「Vertebrate Mitochondrial」、「SGC1」など「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

1. FASTA形式ファイル(sample1.fasta)の場合:

multi-FASTAではないsingle-FASTA形式ファイルです。

```
in_f <- "sample1.fasta" #入力ファイル名を指定し
out_f <- "hoge1.fasta"  #出力ファイル名を指定し

#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
fasta                                           #確認してるだけです

#本番
fasta <- translate(fasta) #アミノ酸配列に翻訳し
fasta                                           #確認してるだけです

#ファイルに保存
writeXStringSet(fasta, file=out_f, format="fasta", width=12)
```

```
RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ

R Console
> kadota
AGTGACGGTCTT

[1] "hoge1.fasta"
> in_f <- "sample1.fasta"
> out_f <- "hoge1.fasta"
>
> #必要なパッケージをロード
> library(Biostrings)
>
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fasta")
> fasta
A DNAStringSet instance of length 1
width seq names
[1] 12 AGTGACGGTCTT kadota
> |
```

コードの解説

イントロ | 一般 | 翻訳配列(translate)を取得

①namesという列の情報が、②FASTA形式ファイル中の③description行部分(行頭が>で、その後いろいろな書かれている部分;ここではkadotaに相当)の情報に相当します。

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。翻訳のための遺伝

コード(genetic code)は、Standard Genetic Codeだそうです。もちろん、"Standard", "SGC0", "Vertebrate Mitochondrial", "SGC1"など「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

1. FASTA形式ファイル(sample1.fasta)の場合:

multi-FASTAではないsingle-FASTA形式ファイルです。

```
in_f <- "sample1.fasta" #入力ファイル名を指定し
out_f <- "hoge1.fasta"  #出力ファイル名を指定し

#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
fasta #確認してるだけです

#本番
fasta <- translate(fasta) #アミノ酸配列に翻訳し
fasta #確認してるだけです

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

```
RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ

R Console
[1] "hoge1.fasta"
> in_f <- "sample1.fasta"
> out_f <- "hoge1.fasta"
>
> #必要なパッケージをロード
> library(Biostrings) #パ$
>
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fas$
> fasta #確$
A DNAStringSet instance of length 1
width seq names
[1] 12 AGTGACGGTCTT kadota
> |
```

コードの解説

イントロ | 一般 | 翻訳配列(translate)を取得(基礎) | Biostrings

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。翻訳のための遺伝コード(genetic code)は、Standard Genetic Codeだそうです。もちろん「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

1. FASTA形式ファイル(sample1.fasta)の場合：

multi-FASTAではないsingle-FASTA形式ファイルです。

```

in_f <- "sample1.fasta"      #入力ファイル名を指定し
out_f <- "hoge1.fasta"      #出力ファイル名を指定し

#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
fasta                                #確認してるだけです

#本番
fasta <- translate(fasta)    #アミノ酸配列に翻訳し
fasta                                #確認してるだけです

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

```

RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ

R Console
>kadota↓
[1] "hoge1.fasta"
> in_f <- "sample
> out_f <- "hoge1.fasta"
>
> #必要なパッケージをロード
> library(Biostrings)
>
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fas$
> fasta
A DNASTringSet instance of length 1
width seq names
[1] 12 AGTGACGGTCTT kadota
> |
    
```

コードの解説

①seqという列の情報が、②塩基配列部分に相当します。
③widthという列の数値情報が、②の塩基配列の長さに相当します。

イントロ | 一般 | 翻訳配列(translate)を取得(基礎) | Biostrings

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。翻訳のための遺伝コード(genetic code)は、Standard Genetic Codeだそうです。もちろん「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

1. FASTA形式ファイル(sample1.fasta)の場合:

multi-FASTAではないsingle-FASTA形式ファイルです。

```
in_f <- "sample1.fasta" #入力ファイル名を指定し
out_f <- "hoge1.fasta"  #出力ファイル名を指定し

#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
fasta                                           #確認してるだけです

#本番
fasta <- translate(fasta) #アミノ酸配列に翻訳し
fasta                                           #確認してるだけです

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

```
RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ

R Console
>kadota↓
[1] "hoge1.fasta"
> in_f <- "sample
> out_f <- "hoge1.fasta
>
> #必要なパッケージをロード
> library(Biostrings)
>
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fas$
> fasta
A DNAStringSet instance of length 1
width seq names
[1] 12 AGTGACGGTCTT kadota
> |
```

①長ったらしい関数名ですが、意味が分かれると納得できます、という話。

readDNAStringSet

イントロ | 一般 | 翻訳配列(translate)を取得(基礎) | Biostrings

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。翻訳のための遺伝コード(genetic code)は、Standard Genetic Codeだそうです。もちろん「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

1. FASTA形式ファイル(sample1.fasta)の場合:

multi-FASTAではないsingle-FASTA形式ファイルです。

```
in_f <- "sample1.fasta"      #入力ファイル名を指定し
out_f <- "hoge1.fasta"       #出力ファイル名を指定し

#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
fasta                                       #確認してるだけです

#本番
fasta <- translate(fasta)      #アミノ酸配列に翻訳し
fasta                             #確認してるだけです

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

```
RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ

R Console

[1] "hoge1.fasta"
> in_f <- "sample1.fasta"      #入$
> out_f <- "hoge1.fasta"      #出$
>
> #必要なパッケージをロード
> library(Biostrings)        #パ$
>
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fas$
> fasta                       #確$
A DNAStringSet instance of length 1
  width seq          names
[1]    12 AGTGACGGTCTT kadota
> |
```

①長ったらしい関数名ですが、意味が分かると納得できません、という話。まず、②DNASTringは塩基配列という意味で、それを③読み込む(read)関数だということ。

readDNASTringSet

イントロ | 一般 | 翻訳配列(translate)を取得(基礎) | Biostrings

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。翻訳のための遺伝コード(genetic code)は、Standard Genetic Codeだそうです。もちろん、"Standard", "SGC0", "Vertebrate Mitochondrial", "SGC1"など「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

1. FASTA形式ファイル(sample1.fasta)の場合:

multi-FASTAではないsingle-FASTA形式ファイルです。

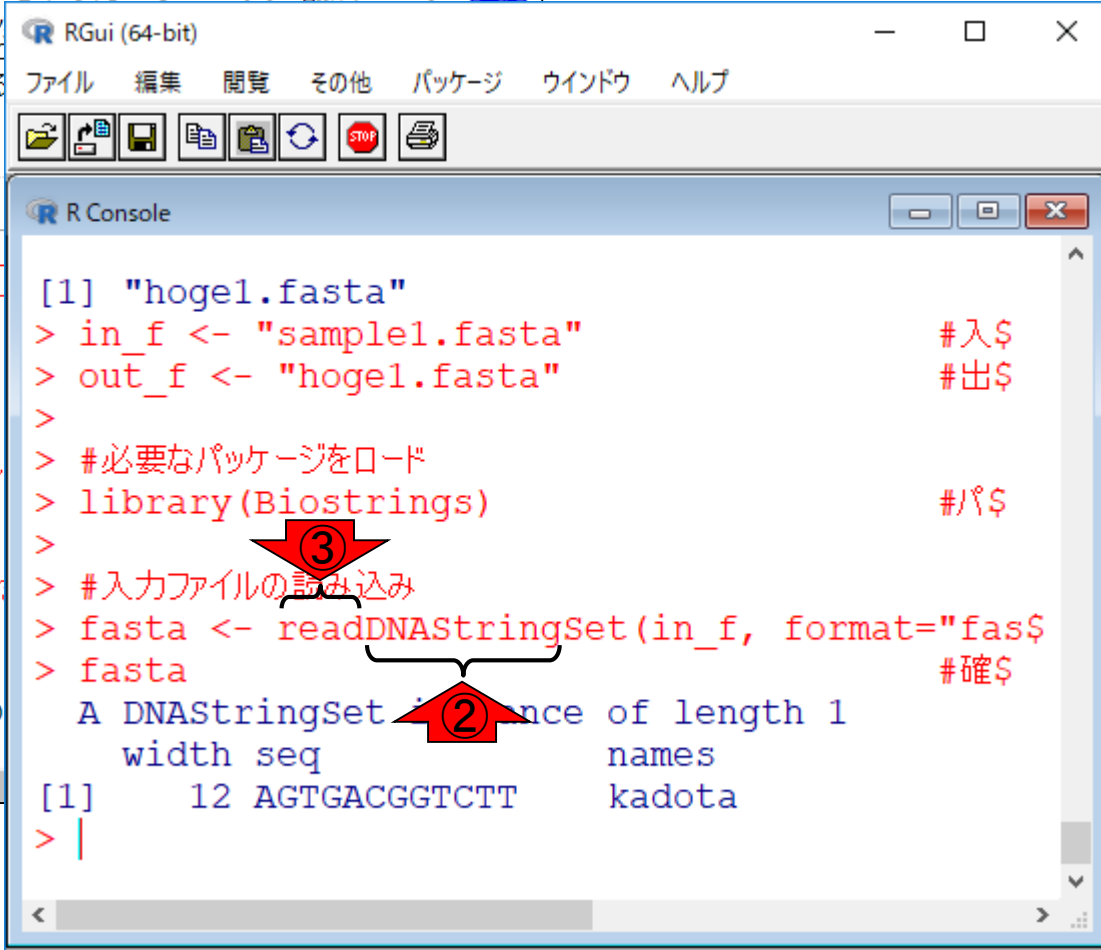
```
in_f <- "sample1.fasta"      #入力ファイル名を指定し
out_f <- "hoge1.fasta"       #出力ファイル名を指定し

#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
fasta                                       #確認してるだけです

#本番
fasta <- translate(fasta)      #アミノ酸配列に翻訳し
fasta                           #確認してるだけです

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



```
RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ

R Console
[1] "hoge1.fasta"
> in_f <- "sample1.fasta"      #入$
> out_f <- "hoge1.fasta"     #出$
>
> #必要なパッケージをロード
> library(Biostrings)        #パ$
>
> #入力ファイルの読み込み
> fasta <- readDNASTringSet(in_f, format="fas$
> fasta                       #確$
A DNASTringSet of length 1
  width seq      names
[1]    12 AGTGACGGTCTT kadota
> |
```


readDNAStringSet

イントロ | 一般 | 翻訳配列(translate)を取得

①長ったらしい関数名ですが、意味が分かると納得できません、という話。まず、②DNASTringは塩基配列という意味で、それを③読み込む(read)関数だということ。④Setは、集合という意味。DNASTringSetで、塩基配列集合だと解釈すればよい。ヒトだって22本の染色体(塩基配列の集

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するコード(genetic code)は、Standard Genetic Codeだそうです。もちろん、"Standard", "SGC0", "Vertebrate Mitochondrial", "SGC0"など、他のコードも利用可能です。「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてある

1. FASTA形式ファイル(sample1.fasta)の場合:

multi-FASTAではないsingle-FASTA形式ファイルです。

```
in_f <- "sample1.fasta" #入力ファイル名を指定し
out_f <- "hoge1.fasta"  #出力ファイル名を指定し

#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
fasta                                           #確認してるだけです

#本番
fasta <- translate(fasta) #アミノ酸配列に翻訳し
fasta                                           #確認してるだけです

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

```
RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ

R Console
[1] "hoge1.fasta"
> in_f <- "sample1.fasta" #入$
> out_f <- "hoge1.fasta" #出$
>
> #必要なパッケージをロード
> library(Biostrings) #パ$
>
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fas$
> fasta #確$
A DNASTringSet of length 1
width seq names
[1] 12 AGTGACGGTCTT kadota
> |
```

この①readDNASTringSet関数を用いて読み込んだ
②fastaオブジェクトは…

DNAStringSet形式

イントロ | 一般 | 翻訳配列(translate)を取得(基礎) | Biostrings

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。翻訳のための遺伝コード(genetic code)は、Standard Genetic Codeだそうです。もちろん「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

1. FASTA形式ファイル(sample1.fasta)の場合:

multi-FASTAではないsingle-FASTA形式ファイルです。

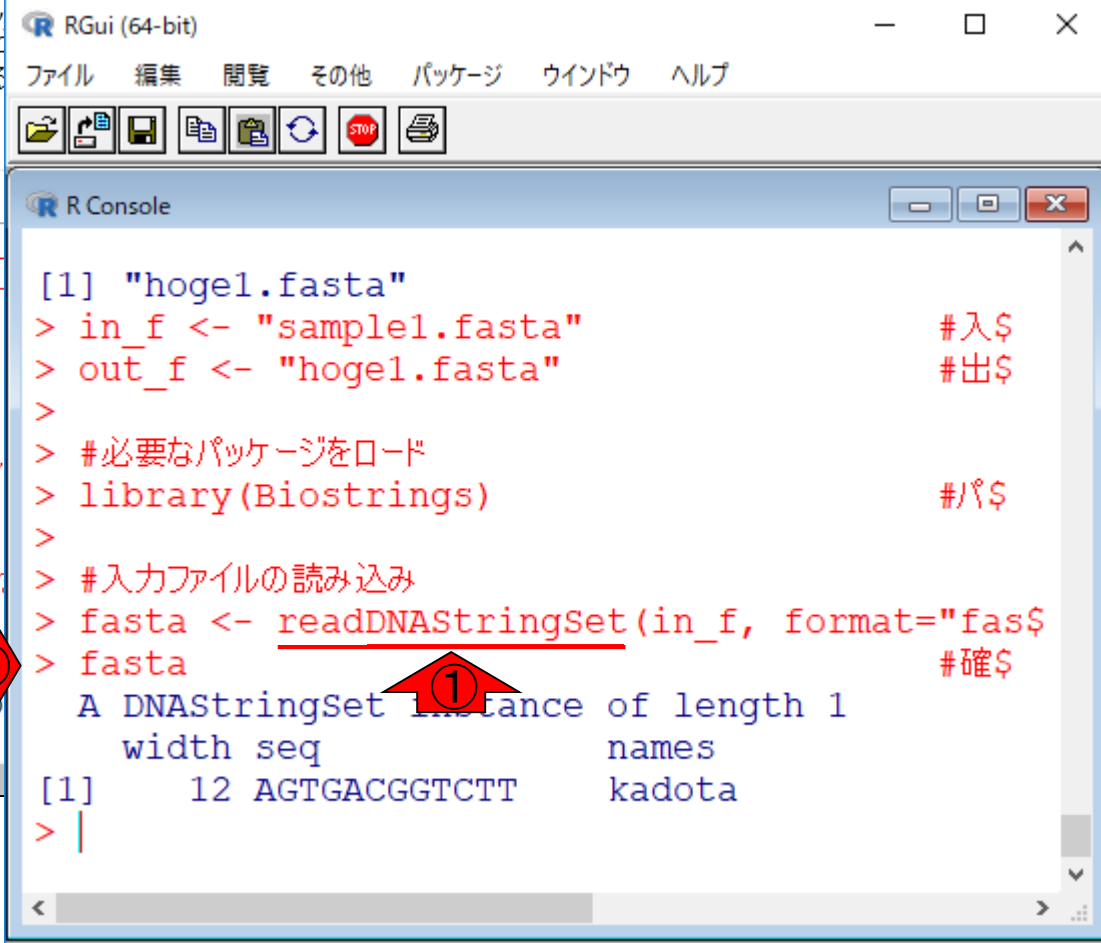
```
in_f <- "sample1.fasta"      #入力ファイル名を指定し
out_f <- "hoge1.fasta"      #出力ファイル名を指定し

#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
fasta                                     #確認してるだけです

#本番
fasta <- translate(fasta)   #アミノ酸配列に翻訳し
fasta                       #確認してるだけです

#ファイルに保存
writeXStringSet(fasta, file=out_f, format="fasta", width=100)
```



```
RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ

R Console

[1] "hoge1.fasta"
> in_f <- "sample1.fasta"      #入$
> out_f <- "hoge1.fasta"      #出$
>
> #必要なパッケージをロード
> library(Biostrings)        #パ$
>
> #入力ファイルの読み込み
> fasta <- readDNASTringSet(in_f, format="fas$
> fasta                       #確$
A DNASTringSet instance of length 1
width seq names
[1] 12 AGTGACGGTCTT kadota
> |
```

DNAStringSet形式

この①readDNAStringSet関数を用いて読み込んだ
②fastaオブジェクトは、③DNAStringSetという形式で
格納されています。実用上は、深く考える必要はなく、

イントロ | 一般 | 翻訳配列(translate)を取得(基礎) | Biostrings

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。翻訳のための遺伝コード(genetic code)は、Standard Genetic Codeだそうです。もちろん、"Standard", "SGC0", "Vertebrate Mitochondrial", "SGC1"など「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

1. FASTA形式ファイル(sample1.fasta)の場合:

multi-FASTAではないsingle-FASTA形式ファイルです。

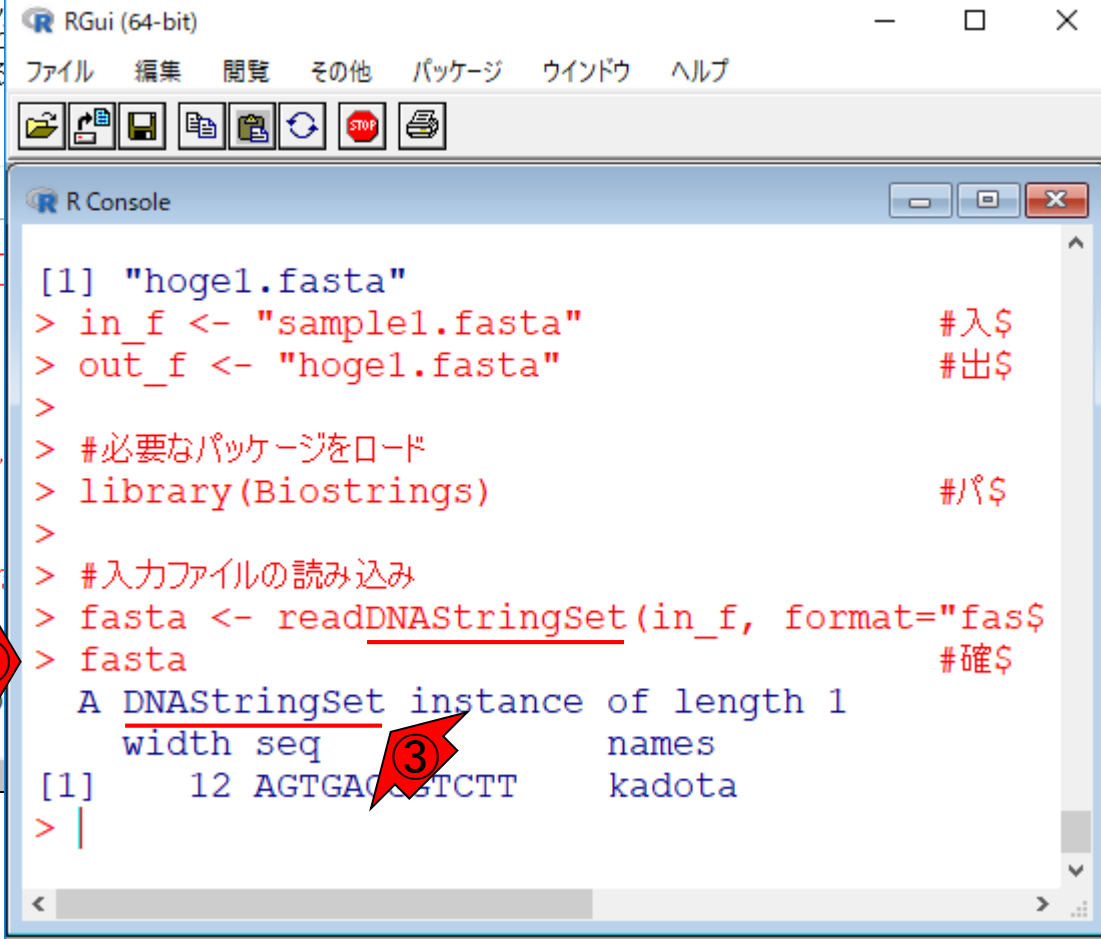
```
in_f <- "sample1.fasta"      #入力ファイル名を指定し
out_f <- "hoge1.fasta"       #出力ファイル名を指定し

#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
fasta                                     #確認してるだけです

#本番
fasta <- translate(fasta)    #アミノ酸配列に翻訳し
fasta                       #確認してるだけです

#ファイルに保存
writeXStringSet(fasta, file=out_f, format="fasta", width=100)
```



```
RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ

R Console

[1] "hoge1.fasta"
> in_f <- "sample1.fasta"      #入$
> out_f <- "hoge1.fasta"     #出$
>
> #必要なパッケージをロード
> library(Biostrings)       #パ$
>
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fas$
> fasta                       #確$
A DNAStringSet instance of length 1
  width seq          names
[1]    12 AGTGACCTCTT kadota
> |
```

DNAStringSet形式

イントロ | 一般 | 翻訳配列(translate)を取得(基礎)

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するコード(genetic code)は、Standard Genetic Codeだそうです。もちろん「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

1. FASTA形式ファイル(sample1.fasta)の場合:

multi-FASTAではないsingle-FASTA形式ファイルです。

```
in_f <- "sample1.fasta"      #入力ファイル名を指定し
out_f <- "hoge1.fasta"       #出力ファイル名を指定し

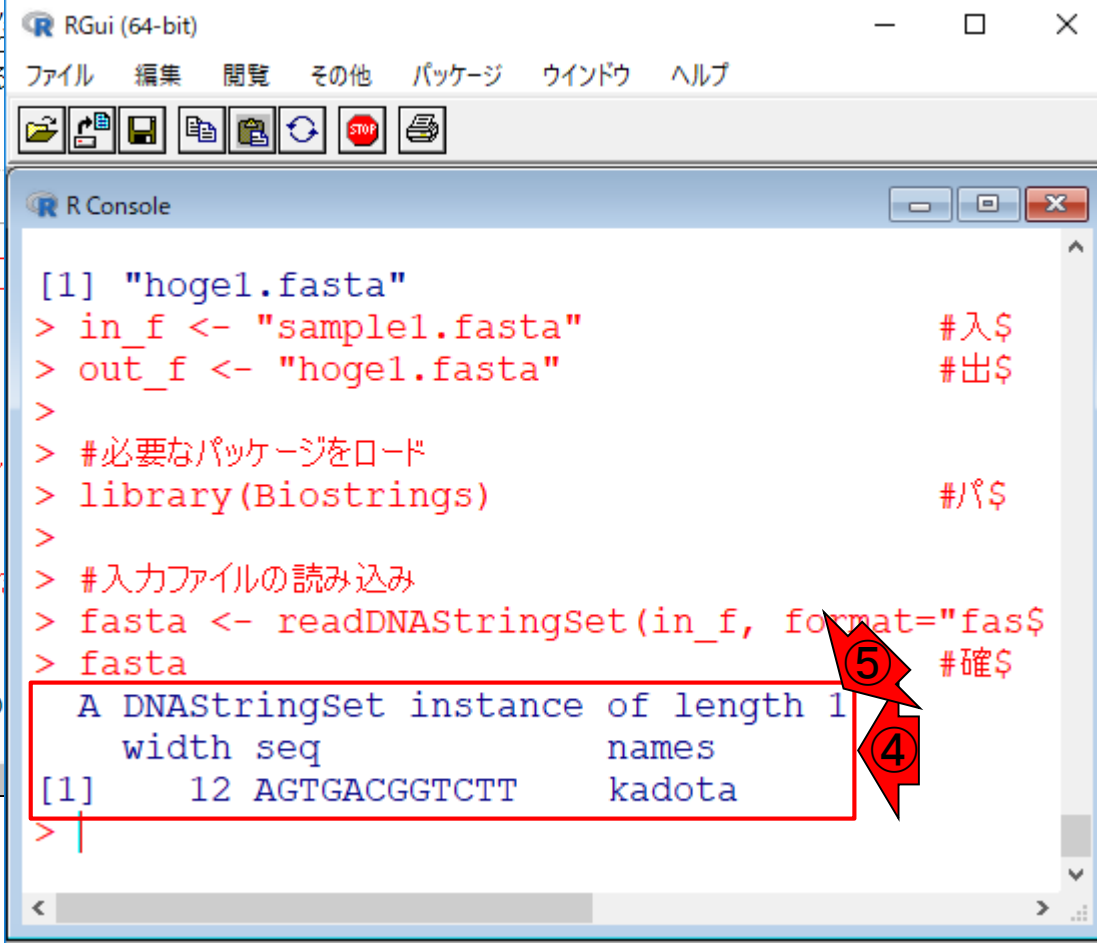
#必要なパッケージをロード
library(Biostrings)         #パッケージの読み込み

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

#本番
fasta <- translate(fasta)    #アミノ酸配列に翻訳し
fasta                       #確認してるだけです

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

この①readDNAStringSet関数を用いて読み込んだ②fastaオブジェクトは、③DNAStringSetという形式で格納されています。実用上は、深く考える必要はなく、④こんな感じの見え方なら、「DNAStringSet形式」だと判断できればそれで十分です。ちなみに⑤は配列



```
RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ

R Console
[1] "hoge1.fasta"
> in_f <- "sample1.fasta"      #入$
> out_f <- "hoge1.fasta"      #出$
>
> #必要なパッケージをロード
> library(Biostrings)        #パ$
>
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fas$
> fasta                       #確$

A DNAStringSet instance of length 1
width seq          names
[1]      12 AGTGACGGTCTT  kadota
> |
```

translate関数

イントロ | 一般 | 翻訳配列(translate)を取得(基礎) | Biostrings

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。翻訳のための**遺伝コード(genetic code)**は、Standard Genetic Codeだそうです。もちろん生物種?!によって多少違い(variants)があるようで、"Standard", "SGC0", "Vertebrate Mitochondrial", "SGC1"などいろいろ選べるようです。

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

1. FASTA形式ファイル(sample1.fasta)の場合：

multi-FASTAではないsingle-FASTA形式ファイルです。

```

in_f <- "sample1.fasta"      #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.fasta"      #出力ファイル名を指定してout_fに格納

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

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

#本番
fasta <- translate(fasta)    #アミノ酸配列に翻訳した結果をfastaに格納
fasta                                     #確認してるだけです

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

```



①

次の①translate関数は、②さきほどのDNAStringSet形式のfastaオブジェクトを、③入力としています。

translate関数

イントロ | 一般 | 翻訳配列(translate)を取得(基礎) | Biostrings

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。翻訳のための遺伝コード(genetic code)は、Standard Genetic Codeだそうです。もちろん「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

1. FASTA形式ファイル(sample1.fasta)の場合:

multi-FASTAではないsingle-FASTA形式ファイルです。

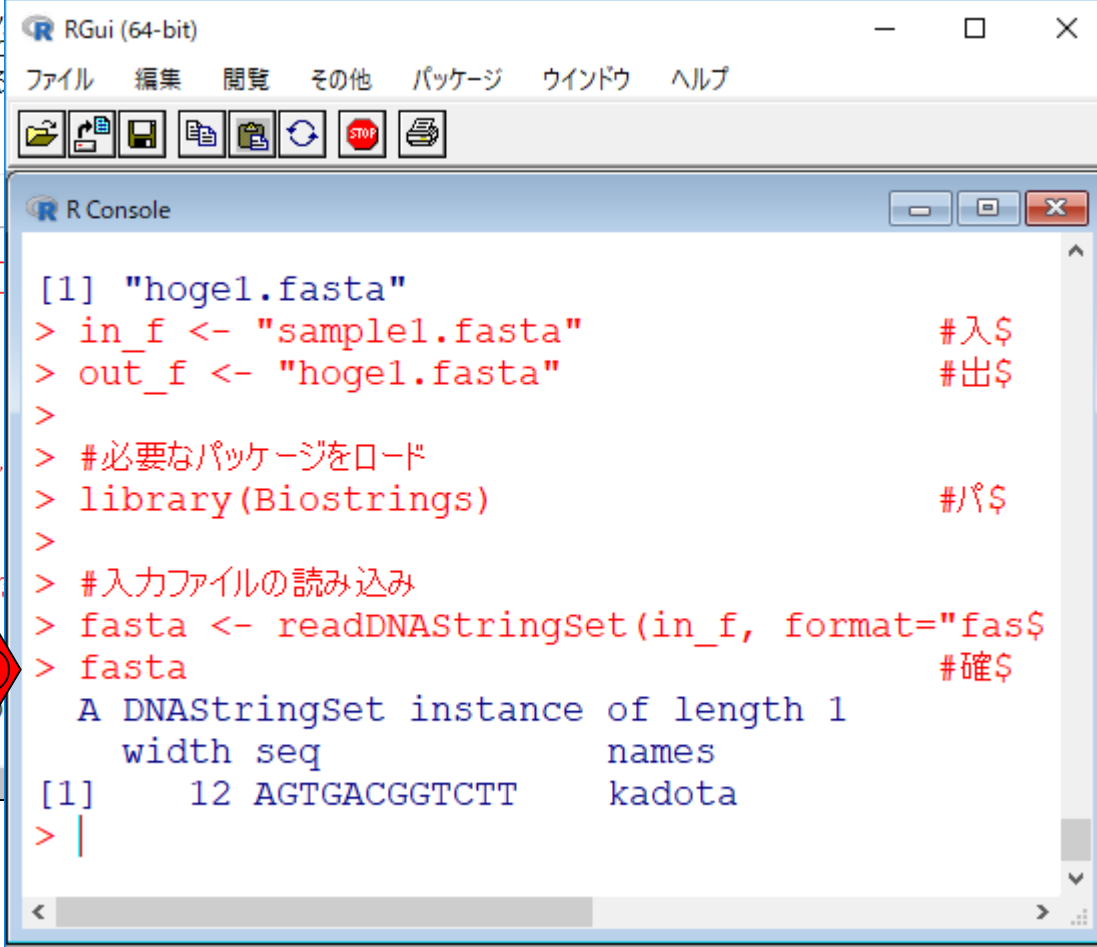
```
in_f <- "sample1.fasta"      #入力ファイル名を指定し
out_f <- "hoge1.fasta"       #出力ファイル名を指定し

#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
fasta                                     #確認してるだけです

#本番 ②
fasta <- translate(fasta)      #アミノ酸配列に翻訳し
fasta                               #確認してるだけです

#ファイルに保存 ①
writeXStringSet(fasta, file=out_f, format="fasta", width=100) ②
```



```
RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ

R Console

[1] "hoge1.fasta"
> in_f <- "sample1.fasta"      #入$
> out_f <- "hoge1.fasta"      #出$
>
> #必要なパッケージをロード
> library(Biostrings)        #パ$
>
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fas$
> fasta                        #確$
A DNAStringSet instance of length 1
  width seq          names
[1]    12 AGTGACGGTCTT kadota
> |
```

translate関数

イントロ | 一般 | 翻訳配列(translate)を取得(基礎)

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するコード(genetic code)は、Standard Genetic Codeだそうです。もちろん「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

1. FASTA形式ファイル(sample1.fasta)の場合:

multi-FASTAではないsingle-FASTA形式ファイルです。

```
in_f <- "sample1.fasta"      #入力ファイル名を指定し
out_f <- "hoge1.fasta"      #出力ファイル名を指定し

#必要なパッケージをロード
library(Biostrings)        #パッケージの読み込み②

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

#準備④
fasta <- translate(fasta)     #アミノ酸配列に翻訳した
fasta                           #確認してるだけです

#ファイルに保存①
writeXStringSet(fasta, file=out_f, format="fasta", width=50)
```

次の①translate関数は、②さきほどのDNAStringSet形式のfastaオブジェクトを、③入力としています。赤枠内コピー実行。④出力情報も同じfastaというオブジェクト名にしています。別の名前(例:hoge)とかでもいいのですが、特に以前のDNAStringSet形式の情報を残しておく必要もないのでそうしています。

ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ



R Console

```
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fas$
> fasta                                     #確$
A DNAStringSet instance of length 1
  width seq          names
[1]    12 AGTGACGGTCTT kadota
>
> #準備④
> fasta <- translate(fasta)                #ア$
> fasta                                     #確$
A AAStringSet instance of length 1
  width seq          names
[1]     4 SDGL        kadota
> |
```

translate関数

イントロ | 一般 | 翻訳配列(translate)を取得(基礎) | Biostrings

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。翻訳のための遺伝コード(genetic code)は、Standard Genetic Codeだそうです。もちろん「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

1. FASTA形式ファイル(sample1.fasta)の場合：

multi-FASTAではないsingle-FASTA形式ファイルです。

```

in_f <- "sample1.fasta"      #入力ファイル名を指定し
out_f <- "hoge1.fasta"      #出力ファイル名を指定し

#必要なパッケージをロード
library(Biostrings)        #パッケージの読み込み②

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

#本番②
fasta <- translate(fasta)    #アミノ酸配列に翻訳し
fasta                       #確認してるだけです

#FASTA形式で保存④①
writeXStringSet(fasta, file=out_f, format="fasta", width=1)
    
```

```

RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ

R Console
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fas$
> fasta                                     #確$
A DNAStringSet instance of length 1
  width seq          names
[1]    12 AGTGACGGTCTT kadota
>
> #本番
> fasta <- translate(fasta)                #ア$
> fasta                                     #確$
A AAStringSet instance of length 1
  width seq          names
[1]     4 SDGL        kadota
> |
    
```


translate関数

①translate関数実行②前と④後のfastaオブジェクト。違いは赤枠部分。DNA配列を入力として、アミノ酸配列(Amino Acids)を出力としているので妥当ですね。

イントロ | 一般 | 翻訳配列(translate)を取得(基礎) | Biostrings

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。翻訳のための遺伝コード(genetic code)は、Standard Genetic Codeだそうです。もちろん「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

1. FASTA形式ファイル(sample1.fasta)の場合:

multi-FASTAではないsingle-FASTA形式ファイルです。

```
in_f <- "sample1.fasta" #入力ファイル名を指定し
out_f <- "hoge1.fasta" #出力ファイル名を指定し

#必要なパッケージをロード
library(Biostrings) #パッケージの読み込み②

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

#本番②
fasta <- translate(fasta) #アミノ酸配列に翻訳し
fasta #確認してるだけです

#FASTA形式で保存④①
writeXStringSet(fasta, file=out_f, format="fasta", width=100) #確認してるだけです
```

```
RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ

R Console
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fasta")
> fasta #確認$
A DNAStringSet instance of length 1
  width seq          names
[1]    12 AGTGACGGTCTT kadota
>
> #本番
> fasta <- translate(fasta) #ア$
> fasta #確認$
A AAStringSet instance of length 1
  width seq          names
[1]     4 SDGL          kadota
> |
```

translate関数

イントロ | 一般 | 翻訳配列(translate)を取得(基礎)

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するコード(genetic code)は、Standard Genetic Codeだそうです。もちろん「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

1. FASTA形式ファイル(sample1.fasta)の場合:

multi-FASTAではないsingle-FASTA形式ファイルです。

```
in_f <- "sample1.fasta" #入力ファイル名を指定し
out_f <- "hoge1.fasta" #出力ファイル名を指定し

#必要なパッケージをロード
library(Biostrings) #パッケージの読み込み②

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

#本番②
fasta <- translate(fasta) #アミノ酸配列に翻訳し
fasta #確認してるだけです

#FASTA形式に保存④①
writeXStringSet(fasta, file=out_f, format="fasta", width=100) #確認してるだけです
```

①translate関数実行②前と④後のfastaオブジェクト。違いは赤枠部分。DNA配列を入力として、アミノ酸配列(Amino Acids)を出力としているので妥当ですね。このことから、Biostringsパッケージが提供するtranslateという関数は、入力がDNAStringSet形式で、出力がAAStringSet形式なのだとすることを学ぶ。

ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ



R Console

```
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fasta")
> fasta #確認$
A DNAStringSet instance of length 1
  width seq          names
[1]    12 AGTGACGGTCTT kadota
>
> #本番
> fasta <- translate(fasta) #ア$
> fasta #確認$
A AAStringSet instance of length 1
  width seq          names
[1]     4 SDGL kadota
> |
```

writeXStringSet

イントロ | 一般 | 翻訳配列(translate)を取得(基礎)

最後の①writeXStringSet関数は、DNA塩基配列のDNAStringSet形式のオブジェクトや、アミノ酸(Amino Acids)配列のAAStringSet形式のオブジェクトを、②formatオプションで指定して形式(デフォルトはfasta)で、③fileオプションで指定したファイル名で出力してくれます。

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するコード(genetic code)は、Standard Genetic Codeだそうです。もちろん生

1. FASTA形式ファイル(sample1.fasta)の場合：

multi-FASTAではないsingle-FASTA形式ファイルです。

```
in_f <- "sample1.fasta"      #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.fasta"       #出力ファイル名を指定してout_fに格納

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

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

#本番
fasta <- translate(fasta)    #アミノ酸配列に翻訳した結果をfastaに格納
fasta                         #確認してるだけです

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

①

③

②

writeXStringSet

イントロ | 一般 | 翻訳配列(translate)を取得(基礎)

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するコード(genetic code)は、Standard Genetic Codeだそうです。もちろん生きているように、"Standard", "SGC0", "Vertebrate Mitochondrial", "SGC1"など「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

最後の①writeXStringSet関数は、DNA塩基配列のDNAStringSet形式のオブジェクトや、アミノ酸(Amino Acids)配列のAAStringSet形式のオブジェクトを、②formatオプションで指定して形式(デフォルトはfasta)で、③fileオプションで指定したファイル名で出力してくれます。④DNA or AAどちらでもよいので、writeXStringSetという名前なのだとして理解すればよい。

1. FASTA形式ファイル(sample1.fasta)の場合 :

multi-FASTAではないsingle-FASTA形式ファイルです。

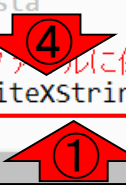
```
in_f <- "sample1.fasta"      #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.fasta"      #出力ファイル名を指定してout_fに格納

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

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

#本番
fasta <- translate(fasta)   #アミノ酸配列に翻訳した結果をfastaに格納
fasta                                #確認してるだけです

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



writeXStringSet

イントロ | 一般 | 翻訳配列(translate)を取得(基礎)

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するコード(genetic code)は、Standard Genetic Codeだそうです。もちろん生るようで、"Standard", "SGC0", "Vertebrate Mitochondrial", "SGC1"など「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

1. FASTA形式ファイル(sample1.fasta)の場合：

multi-FASTAではないsingle-FASTA形式ファイルです。

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

#必要なパッケージをロード
library(Biostrings)

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

#本番
fasta <- translate(fasta)

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

#入力ファイル名を指定し
#出力ファイル名を指定し

#パッケージの読み込み

#in_fで指定したファイルの読み込み
#確認してるだけです

#アミノ酸配列に翻訳した結果をfastaに格納
#確認してるだけです

#fastaの中身を指定したファイル名で保

最後の①writeXStringSet関数は、DNA塩基配列のDNAStringSet形式のオブジェクトや、アミノ酸(Amino Acids)配列のAAStringSet形式のオブジェクトを、②formatオプションで指定して形式(デフォルトはfasta)で、③fileオプションで指定したファイル名で出力してくれます。④DNA or AAどちらでもよいので、writeXStringSetという名前なのだと理解すればよい。⑤の出力ファイル名情報を格納したout_fは、①writeXStringSet関数の③fileオプション部分で利用されています。

Contents

- R本体とRパッケージの関係
- Rパッケージ: BioconductorとCRAN
- Rの基本的な利用法のおさらい: Biostringsを用いた翻訳配列の取得
 - 入力ファイルのダウンロード、Rの起動を作業ディレクトリの変更
 - コピペ実行、コードの解説
- 複数の例題を実行して理解を深める

複数の例題

イントロ | 一般 | 翻訳配列(translate)を取得(基礎) | Biostrings

Biostringsパッケージを用いて塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。翻訳のための遺伝コード(genetic code)は、Standard Genetic Codeだそうです。もちろん生物種?!によって多少違い(variants)があるようで、"Standard", "SGC0", "Vertebrate Mitochondrial", "SGC1"などいろいろ選べるようです。
「ファイル」 - 「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピー。

1. FASTA形式ファイル(sample1.fasta)の場合：

multi-FASTAではないsingle-FASTA形式ファイルです。

```
in_f <- "sample1.fasta"      #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.fasta"      #出力ファイル名を指定してout_fに格納

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

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

#本番
fasta <- translate(fasta)    #アミノ酸配列に翻訳した結果をfastaに格納
fasta                         #確認してるだけです

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



2. (multi-)FASTA形式ファイル(sample4.fasta)の場合：

配列中にACGT以外のもが存在するためエラーが出る例です。4番目の配列(つまりgene_4)の17番目のポジションがNなので妥当です。

```
in_f <- "sample4.fasta"      #入力ファイル名を指定してin_fに格納
out_f <- "hoge2.fasta"      #出力ファイル名を指定してout_fに格納
```


例題2

①の項目の場合も、②例題2が提供されています。
②例題2は、③入力ファイルが、④のような中身となっています。計5個の塩基配列ですので、DNAStr**ingSet**の意味が実感できます。

2. (multi-)FASTA形式ファイル(sample4.fasta)の場合:

配列中にACGT以外のものが存在するためエラーが出る例です。4番目の配列(つまりgene_4)の17番目のポジションがNなので妥当です。

```
in_f <- "sample4.fasta" #入力ファイル名を指定してin_fに格納
out_f <- "hoge2.fasta"  #出力ファイル名を指定してout_fに格納

#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
fasta <- translate(fasta) #アミノ酸配列に翻訳した結果をfastaに格納
fasta                       #確認してるだけです

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

```
>gene_1↓
CGACAGCTCCTCGGCATCCGA↓
>gene_2↓
GTCTGCCTCAAGCGCCCCAAGTGGGTT↓
>gene_3↓
TGTAGGAGAAGGGCGTAATCT↓
>gene_4↓
CGTGCTGATTCCACACNGCAGTAAACGCGG↓
>gene_5↓
CGTGCTGATTCCACACAGCAGTAAACGCGG↓
```

例題2

②

①の項目の場合も、②例題2が提供されています。
②例題2は、③入力ファイルが、④のような中身となっています。計5個の塩基配列ですので、DNAStrngSetの意味が実感できます。赤枠の結果。

2. (multi-)FASTA形式ファイル(sample4.fasta)の場合:

配列中にACGT以外のものが存在するためエラーが出る例です。4番目の配列(つまりgene_4)の17番目のポジションがNなので妥当です。

```
in_f <- "sample4.fasta" #入力ファイル名を指
out_f <- "hoge2.fasta"  #出力ファイル名を指

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

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

#本番
fasta <- translate(fasta) #アミノ酸配列に翻訳
fasta #確認してるだけです

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

③

The screenshot shows the RGui interface with the R Console window open. The console displays the output of the R code from the previous block. The output shows a DNAStrngSet instance of length 5 with the following details:

	width	seq	names
[1]	21	CGACAG...ATCCGA	gene_1
[2]	27	GTCTGC...TGGGTT	gene_2
[3]	21	TGTAGG...TAATCT	gene_3
[4]	30	CGTGCT...ACGCGG	gene_4
[5]	30	CGTGCT...ACGCGG	gene_5

例題2

①入力ファイル読み込み後のfastaオブジェクトの中身。配列長が一定以上の長さになると、②R Console画面上で1配列1行分で表示しきれなくなるため、③「…」で表現される。④5配列なので、5になります。

2. (multi-)FASTA形式ファイル(sample4.fasta)の場合:

配列中にACGT以外のものが存在するためエラーが出る例です。4番目の配列(つまりgene_4)の17番目のポジションがNなので妥当です。

```
in_f <- "sample4.fasta" #入力ファイル名を指  
out_f <- "hoge2.fasta" #出力ファイル名を指  
  
#必要なパッケージをロード  
library(Biostrings) #パッケージの読み込  
  
#入力ファイルの読み込み  
fasta <- readDNAStringSet(in_f, format="fasta") #in_fで指定  
fasta #確認してるだけです  
  
#本番  
fasta <- translate(fasta) #アミノ酸配列に翻訳  
fasta #確認してるだけです  
  
#ファイルに保存  
writeXStringSet(fasta, file=out_f, format="fasta", width=60)
```

The screenshot shows the RGui interface with the R Console window open. The console displays the output of the R code, showing a DNAStringSet instance of length 5. The output is a table with columns 'width', 'seq', and 'names'. The 'seq' column contains truncated sequences due to the width constraint, indicated by ellipses. Red arrows point to specific parts of the console output: arrow 1 points to the 'library(Biostrings)' command, arrow 2 points to the 'readDNAStringSet' command, arrow 3 points to the truncated sequence 'CGACAG...ATCCGA' in the first row, and arrow 4 points to the 'names' column showing 'gene_1' through 'gene_5'.

```
RGui (64-bit)  
ファイル 編集 閲覧 その他 パツ  
R Console  
> #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="fas$  
> fasta #確認$  
  
A DNAStringSet instance of length 5  
width seq names  
[1] 21 CGACAG...ATCCGA gene_1  
[2] 27 GTCTGC...TGGGTT gene_2  
[3] 21 TGTAGG...TAATCT gene_3  
[4] 30 CGTGCT...ACGCGG gene_4  
[5] 30 CGTGCT...ACGCGG gene_5  
> |
```

Tips (小ネタ)

①や②のようにすれば、任意の配列を抽出することができます(例えばヒトのChr19とChr21の配列のみ抽出するなど)。他にも一定の長さ以上のものを抽出することなどもできます。

2. (multi-)FASTA形式ファイル(sample4.fasta)の場合:

配列中にACGT以外のものが存在するためエラーが出る例です。4番目の配列(つまりgene_4)の17番目のポジションがNなので妥当です。

```
in_f <- "sample4.fasta" #入力ファイル名を指  
out_f <- "hoge2.fasta" #出力ファイル名を指  
  
#必要なパッケージをロード  
library(Biostrings) #パッケージの読み込  
  
#入力ファイルの読み込み  
fasta <- readDNAStringSet(in_f, format="fasta") #in_fで指定  
fasta #確認してるだけです  
  
#本番  
fasta <- translate(fasta) #アミノ酸配列に翻訳  
fasta #確認してるだけです  
  
#ファイルに保存  
writeXStringSet(fasta, file=out_f, format="fasta", width=
```

```
RGui (64-bit)  
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ  
[2] 27 GTCTGC...TGGGTT gene_2  
[3] 21 TGTAGG...TAATCT gene_3  
[4] 30 CGTGCT...ACGCGG gene_4  
[5] 30 CGTGCT...ACGCGG gene_5  
> fasta[4]  
A DNAStringSet instance of length 1  
width seq names  
[1] 30 CGTGCT...ACGCGG gene_4  
> fasta[2:3]  
A DNAStringSet instance of length 2  
width seq names  
[1] 27 GTCTGC...TGGGTT gene_2  
[2] 21 TGTAGG...TAATCT gene_3  
> |
```

Contents

- R本体とRパッケージの関係
- Rパッケージ: BioconductorとCRAN
- Rの基本的な利用法のおさらい: Biostringsを用いた翻訳配列の取得
 - 入力ファイルのダウンロード、Rの起動を作業ディレクトリの変更
 - コピペ実行、コードの解説
- 複数の例題を実行して理解を深める
- 「(Rで)塩基配列解析」の興味ある項目の例題をこなす

Biostrings

(Rで)塩基配列解析 ①

(last modified 2019/02/10, since 2010)

このウェブページのR関連部分は、[インストール](#) | についての推奨手順 (Windows2018.11.15版と Macintosh2018.11.27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](#)(Windows2018.12.23版と Macintosh2019.01.15版)で自習してください。2018年7月に[\(Rで\)塩基配列解析の一部](#) (講習会・書籍・学会誌など) を切り分けて[サブページ](#)に移行

What's new? (過去)

「生命科学データ

- [イントロ](#) | 一般 | [指定した範囲の配列を取得](#) (last modified 2015/04/06)
- [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\(基礎\)](#) | [Biostrings](#) (last modified 2015/09/12)
- [イントロ](#) | 一般 | [翻訳配列\(translate\)を取得\(応用\)](#) | [seqinr\(Charif_2005\)](#) (last modified 2015/03/09)
- [イントロ](#) | 一般 | [相補鎖\(complement\)を取得](#) (last modified 2013/06/14)
- [イントロ](#) | 一般 | [逆相補鎖\(reverse complement\)を取得](#) (last modified 2013/06/14)
- [イントロ](#) | 一般 | [逆鎖\(reverse\)を取得](#) (last modified 2013/06/14)
- [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)

①(Rで)塩基配列解析において、Biostringsが提供する関数を利用した項目としては、例えば②などもあります。もちろんBiostringsパッケージが提供する全ての関数について紹介しているわけではありません。

②

Biostrings

Biostringsが他にどのような関数を提供しているかは、①のリンク先(単純にBiostringsでググっても辿れます。)でも見られます。

The screenshot shows a web browser window with the URL `bioconductor.org/packages/release/BiocViews.html#___Software`. The page displays a list of Bioconductor packages. The 'Biostrings' package is highlighted with a red box, and a red arrow with the number 1 points to its name.

Package Name	Maintainer	Description	Rank
AnnotationDbi	Bioconductor Package Maintainer	Annotation Database Interface	6
zlibbioc	Bioconductor Package Maintainer	An R packaged zlib-1.2.5	7
GenomicRanges	Bioconductor Package Maintainer	Representation and manipulation of genomic intervals and variables defined along a genome	8
limma	Gordon Smyth	Linear Models for Microarray Data	9
XVector	Hervé Pagès	Representation and manipulation of external sequences	10
BiocParallel	Bioconductor Package Maintainer	Bioconductor facilities for parallel evaluation	11
GenomeInfoDb	Bioconductor Package Maintainer	Utilities for manipulating chromosome and other 'seqname' identifiers	12
Biostrings	H. Pagès	Efficient manipulation of biological strings	13
SummarizedExperiment	Bioconductor Package Maintainer	SummarizedExperiment container	14
DelayedArray	Hervé Pagès	Delayed operations on array-like objects	15
annotate	Bioconductor Package Maintainer	Annotation for microarrays	16
	Bioconductor	genefilter: methods for filtering	

ここがパッケージ提供サイト①Bioconductor内の、
②Biostringsのページ。③少しページ下部に移動。

Biostrings

The screenshot shows a web browser window displaying the Bioconductor Biostrings package page. The browser's address bar shows the URL: `bioconductor.org/packages/release/bioc/html/Biostrings.html`. The page features a teal navigation bar with the Bioconductor logo and menu items: Home, Install, Help, Developers, and About. A search bar is located in the top right. The main content area includes a breadcrumb trail: Home » Bioconductor 3.8 » Software Packages » Biostrings. The title 'Biostrings' is prominently displayed. Below the title, there are several status boxes: 'platforms all', 'rank 13 / 1649', 'posts 14 / 0.8 / 1 / 2', 'in Bioc > 14 years', 'build warnings', and 'updated < 3 months'. The DOI is listed as `10.18129/B9.bioc.Biostrings`. The page title is 'Efficient manipulation of biological strings'. The description states: 'Bioconductor version: Release (3.8). Memory efficient string containers, string matching algorithms, and other utilities, for fast manipulation of large biological sequences or sets of sequences.' The author is listed as H. Pagès, P. Aboyoun, R. Gentleman, and S. DebRoy. The citation is: 'Pagès H, Aboyoun P, Gentleman R, DebRoy S (2019). Biostrings: Efficient manipulation of biological strings. R package version 2.50.2.' On the right side, there are sections for 'Documentation' and 'Support' with various links.

① Home

② Biostrings

③

Biostrings

①Documentationのあたりまでくると、②赤枠内にPDFファイルのリンク先が見られます。例えば③Biostrings Quick OverviewのPDFをクリックすると…

Bioconductor - Biostrings

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

Documentation

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

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

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

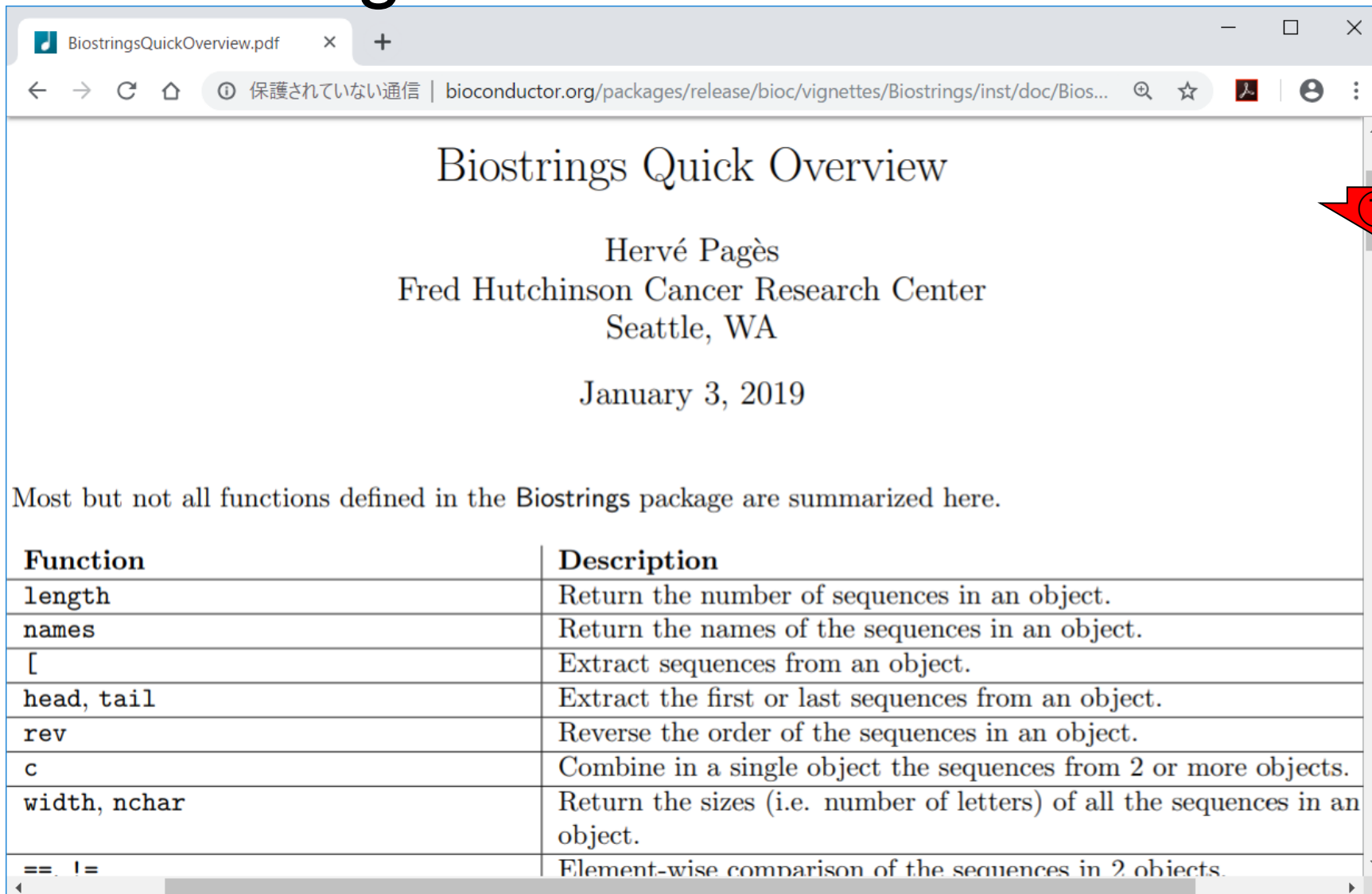
Details

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

Version 2.50.2

Biostrings

こんな感じになります。①ページ下部に移動しながら、記述内容を見ていくと…



Biostrings Quick Overview

Hervé Pagès
Fred Hutchinson Cancer Research Center
Seattle, WA

January 3, 2019

Most but not all functions defined in the Biostrings package are summarized here.

Function	Description
length	Return the number of sequences in an object.
names	Return the names of the sequences in an object.
[Extract sequences from an object.
head, tail	Extract the first or last sequences from an object.
rev	Reverse the order of the sequences in an object.
c	Combine in a single object the sequences from 2 or more objects.
width, nchar	Return the sizes (i.e. number of letters) of all the sequences in an object.
==, !=	Element-wise comparison of the sequences in 2 objects.

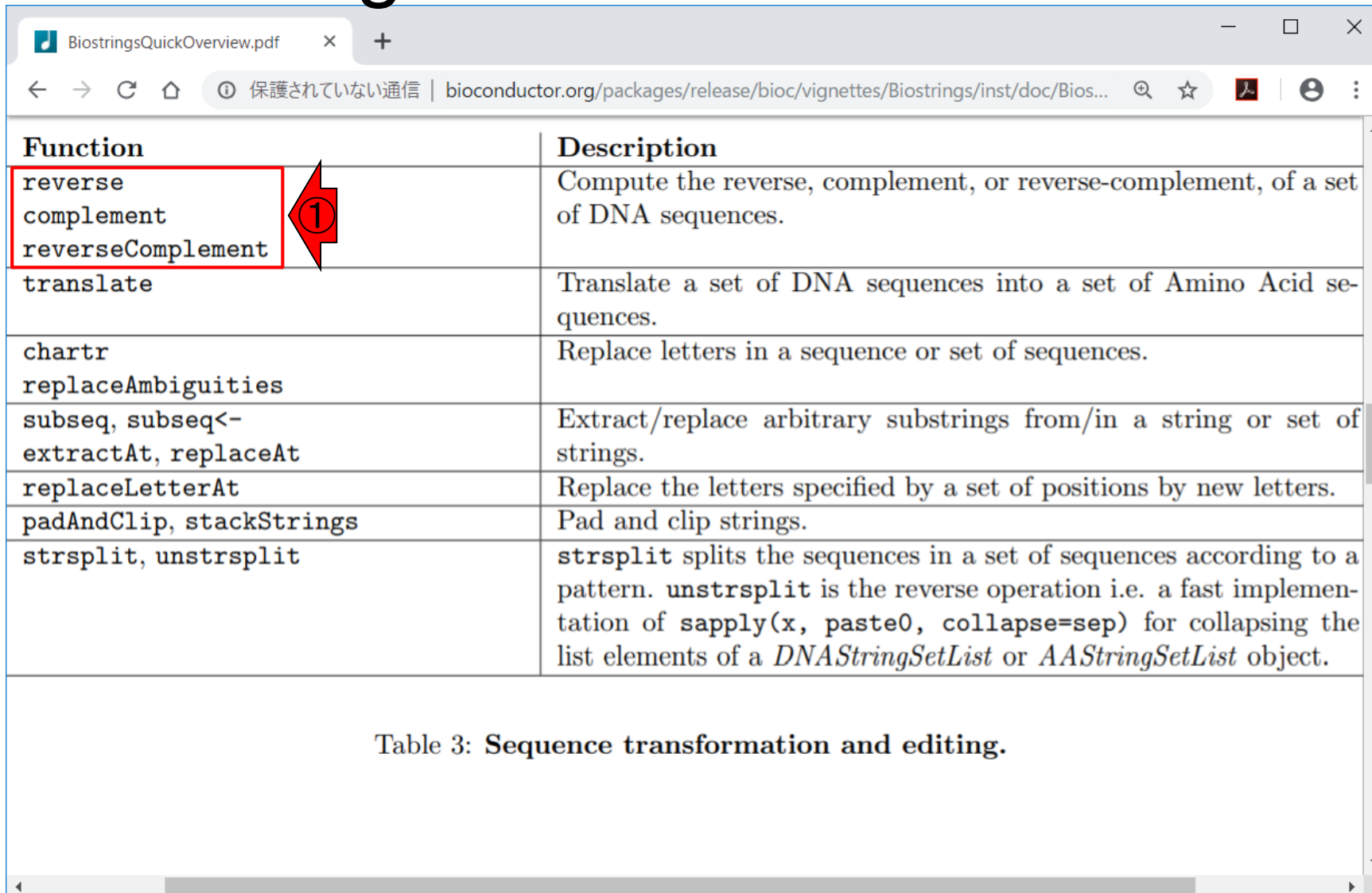
Biostrings

①Table 3で、さきほど利用した②translate関数を見つけた。③それがどういう役割を果たすものかに関する記述。こんな感じで挙動が分かっているものを眺めて、少しずつ使える関数を増やしていくとよい。

Function	Description
reverse complement reverseComplement	Compute the reverse, complement, or reverse-complement, of a set of DNA sequences.
translate	Translate a set of DNA sequences into a set of Amino Acid sequences.
chartr replaceAmbiguities	Replace letters in a sequence or set of sequences.
subseq, subseq<- extractAt, replaceAt	Extract/replace arbitrary substrings from/in a string or set of strings.
replaceLetterAt	Replace the letters specified by a set of positions by new letters.
padAndClip, stackStrings	Pad and clip strings.
strsplit, unstrsplit	strsplit splits the sequences in a set of sequences according to a pattern. unstrsplit is the reverse operation i.e. a fast implementation of <code>sapply(x, paste0, collapse=sep)</code> for collapsing the list elements of a <i>DNASetList</i> or <i>AASetList</i> object.

① Table 3: Sequence transformation and editing.

Biostrings



Function	Description
reverse complement reverseComplement	Compute the reverse, complement, or reverse-complement, of a set of DNA sequences.
translate	Translate a set of DNA sequences into a set of Amino Acid sequences.
chartr replaceAmbiguities	Replace letters in a sequence or set of sequences.
subseq, subseq<- extractAt, replaceAt	Extract/replace arbitrary substrings from/in a string or set of strings.
replaceLetterAt	Replace the letters specified by a set of positions by new letters.
padAndClip, stackStrings	Pad and clip strings.
strsplit, unstrsplit	strsplit splits the sequences in a set of sequences according to a pattern. unstrsplit is the reverse operation i.e. a fast implementation of <code>sapply(x, paste0, collapse=sep)</code> for collapsing the list elements of a <i>DNASetList</i> or <i>AASetList</i> object.

Table 3: Sequence transformation and editing.

Biostrings

(Rで)塩基配列解析 ②

(last modified 2019/02/10, since 2010)

このウェブページのR関連部分は、[インストール](#) | についての推奨手順 (Windows2018.11.15版と Macintosh2018.11.27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](#)(Windows2018.12.23版と Macintosh2019.01.15版)で自習してください。2018年7月に[\(Rで\)塩基配列解析の一部](#) (講習会・書籍・学会誌など) を切り分けて[サブページ](#)に移行

What's new? (過去)

「生命科学データ

- [イントロ](#) | 一般 | [指定した範囲の配列を取得](#) (last modified 2015/04/06)
- [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\(基礎\)](#) | [Biostrings](#) (last modified 2015/09/12)
- [イントロ](#) | 一般 | [翻訳配列\(translate\)を取得\(応用\)](#) | [seqinr\(Charif_2005\)](#) (last modified 2015/03/09)
- [イントロ](#) | 一般 | [相補鎖\(complement\)を取得](#) (last modified 2013/06/14)
- [イントロ](#) | 一般 | [逆相補鎖\(reverse complement\)を取得](#) (last modified 2013/06/14)
- [イントロ](#) | 一般 | [逆鎖\(reverse\)を取得](#) (last modified 2013/06/14)
- [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)

①ここでも示しています。②のウェブページだけでも相当な項目数がありますので、まずはBioconductorのパッケージリストなどを眺めずに、②の様々な興味ある項目の例題をやって慣れていくとよいと思います。