

講義資料のPDFは、本講義科目のページからダウンロード可能です。スライド11までは自習。スライド12から講義を行います。ディレクトリの変更などの環境構築は自力で行ってください。

ゲノム情報解析基礎：第3回

¹大学院農学生命科学研究科
アグリバイオインフォマティクス教育研究プログラム
²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/>

各講義科目へのアクセス

①教育プログラム、②各講義のページ、③「ゲノム情報解析基礎」の場合。ブラウザは、Google Chromeを推奨。



東京大学大学院農学系研究所
アグリバイオ
Agricultural Bioinformatics

ようこそ!!
アグリバイオ
教育研究ユニット

- + ホーム
- + 本ユニットについて
- + メンバー
- + 教育プログラム
- + 研究フォーラム
- + イベント
- + お問い合わせ
- + リンク

東京大学
THE UNIVERSITY OF TOKYO

教育プログラム

- ▼ プログラム概要
- ▼ 講義について
- ▼ 受講について
- ▼ 各講義のページ
- ▼ スケジュール

プログラム概要

本プログラムで開講する講義科目は()に分けられます。カテゴリーと

カテゴリー	目的
基礎	主にバイオインフォマティクス。生命科学のためのツールを利用した様々なツ
方法論	「基礎」の科目を土台として、手法、質量分析法など)やモデル選択、分子シミュレーション(加)について解説し

お知らせ

- ▶ 2019年度の受講希望者は2019年4月5日までに事務局までお越しください。
- ▶ 2019年度受講生募集要項は [こちら](#) (PDF) です。 **NEW!!**
- ▶ 受講に関する質問はまずこちらをごらんください。
Q & A集(本学の大学院生の方)
Q & A集(本学の大学院生以外の方)
- ▶ 成績証明書の発行を希望される方は申込用紙「Word形式、PDF形式」事務局までご連絡ください。

各講義のページ

(科目名をクリックすると各講義のページに移動します)

先端トピックス

セミナー・討論形式
研究指導

農学生命情報科学特別演習

農学生命情報科学特論 I

農学生命情報科学特論 II

農学生命情報科学特論 III

農学生命情報科学特論 IV

方法論

講義・実習を一体化

生物配列統計学 システム生物学概論 知識情報処理論
オーム情報解析 機能ゲノム学 分子モデリングと分子シミュレーション
フィールドインフォマティクス

基礎

講義・実習を一体化

ゲノム情報解析基礎 構造バイオインフォマティクス基礎
生物配列解析基礎 バイオスタティスティクス基礎論

Contents

- 任意のキーワードを含む行を抽出するコードの解説の続き
 - おさらい、行列要素の抽出、`as.character`、`is.element`
 - 行列要素の抽出、`write.table`
- multi-FASTAファイルからの各種情報抽出
 - 基本情報取得(コンティグ数、配列長、N50、GC含量)
 - 課題1
 - コード内部の説明、`readDNAStringSet`、`width`、`sum`、`length`
 - サブセットの抽出(subsetting)、条件判定
 - 実践的な使い方(指定した長さ以上の配列を取得)
- Rパッケージの話
 - Bioconductor概観、Biostrings、推奨インストール手順、マニュアルなど
 - 課題2と課題3

位置づけと心構え

①(Rで)塩基配列解析の、②基本的な利用法(③Win版 or ④Mac版)の続きとして、Rを用いた行列形式ファイルの取り扱いからスタートしています。ゲノム情報解析と関係性が低いと思われるかもしれませんが、このあたりの基礎が多くの発展的な解析の基本スキルとして重要。

(Rで)塩基配列解析

x +

← → ↻ 🏠 ⓘ 保護されていない通信 | www.iu.a.u-tokyo.ac.jp/~kadota/

(Rで)塩基配列解析 ①

(last modified 2019/03/11, 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) ② についてフリーソフト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③ 会誌など)を切り分け、[ページ](#)④に移行しました。(2018/07/18)

What's new? (過去のお知らせはこちら)

- 2019年度も[アグリバイオインフォマティクス教育研究プログラム](#)を実施します。例年東大以外の企業の方、研究員、大学院生が2割程度受講しております。受講ガイダンスは、2019年4月5日(Fri.)17:15より東大農学部2号館2階化学第一講義室で開催します。(2019/03/11) **NEW**
- 細かいところの修正はここに明記してなくても随時行っています。(2019/03/11) **NEW**
- 「インストール | Rパッケージ | [必要最小限プラスアルファ](#)」を更新しました。(2019/03/08) **NEW**
- 「[生命科学データ解析を支える情報技術](#) (監修: 坊農秀雅)」が出版されています。最先端のネタを含むかなり広範な内容を含んでいますので、一通り目次を眺めてみるとよいと思います。Bioconda, Homebrew, Docker, GitHub, EC2, AWSなど聞いたことがある有用そうなものの全体像がわかるというメリットがあると思います。(2019/02/06)

- [はじめに](#) (last modified 2018/08/04)

[トップページへ](#)

- [過去のお知らせ](#) (last modified 2019/03/02) **NEW**

目的: タブ区切りテキストファイル(annotation.txt)中の第1列目に対して、リストファイル(genelist1.txt)中の文字列と一致する行を抜き出して、hoge1.txtというファイル名で出力したい。

全体像

入力: アノテーションファイル(annotation.txt)

	A	B	C	D
1	gene1	hoge01	plasma_mem	nuclear
2	gene2	hoge02	hohinu	membrane
3	gene3	hoge03	agribio	endoplasmic
4	gene4	hoge04	genesis	endoplasmic
5	gene5	hoge05	kamo	membrane
6	gene6	hoge06	netteba	humei
7	gene7	hoge07	tebasaki	nuclear
8	gene8	hoge08	biiru	nuclear
9	gene9	hoge09	nihonshu	nuclear
10	gene10	hoge10	agene1	membrane
11	gene11	hoge11	iyaaaa	endoplasmic

出力: hoge1.txt

	A	B	C	D
1	gene1	hoge01	plasma_mem	nuclear
2	gene7	hoge07	tebasaki	nuclear
3	gene9	hoge09	nihonshu	nuclear

入力: リストファイル(genelist1.txt)

	A
1	gene1
2	gene7
3	gene9

任意のキーワード

目的: タブ区切りテキストファイル(annotation.txt)中の第1列目に対して、リストファイル(genelist1.txt)中の文字列と一致する行を抜き出して、hoge1.txtというファイル名で出力したい。

①の②例題1を実行し、③赤枠の解説を前回行いました。

- インストール | Rパッケージ | [個別\(2018年11月以前\)](#) (last modified 2015/04/03)
- [基本的な利用法](#) (last modified 2015/04/03)
- [サンプルデータ](#) (last modified 2018/06/09)
- 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\(基礎\)](#) ① (last modified 2016/04/20)
- 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 2015/02/19)
- イントロ | 一般 | [指定した範囲の配列を取得](#) (last modified 2015/02/19)
- イントロ | 一般 | [指定したID\(染色体やdescription\)で検索](#) (last modified 2015/02/19)
- イントロ | 一般 | [翻訳配列\(translate\)を取得\(基礎\)](#) ② (last modified 2015/02/19)
- イントロ | 一般 | [翻訳配列\(translate\)を取得\(応用\)](#) (last modified 2015/02/19)

イントロ | 一般 | 任意のキーワードを含む行を抽出(基礎)

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

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

```
in_f1 <- "annotation.txt" #入力ファイル名を指定してin_f1に格納(アノテーションファイル)
in_f2 <- "genelist1.txt" #入力ファイル名を指定してin_f2に格納(リストファイル)
out_f <- "hoge1.txt" #出力ファイル名を指定してout_fに格納
param <- 1 #アノテーションファイル中の検索したい列番号を指定
```

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

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

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


作業ディレクトリは①「デスクトップ - hoge」。hogeフォルダ中に②annotation.txtと③genelist1.txtが存在するという前提。

任意のキーワード

イントロ | 一般 | 任意のキーワードを含む行を抽出(基礎)

例えばタブ区切りテキストファイルが手元があり、この中からリストファイルを示します。Linux (UNIX)のgrepコマンドのようなものであり、perlのパーリ「ファイル」 - 「ディレクトリの変更」で解析したいファイル置いてある

1. 目的のタブ区切りテキストファイル(②annotation.txt)中の第1列目をキーワード(③genelist1.txt)中のものが含まれる行全体を出力したい場合:

```

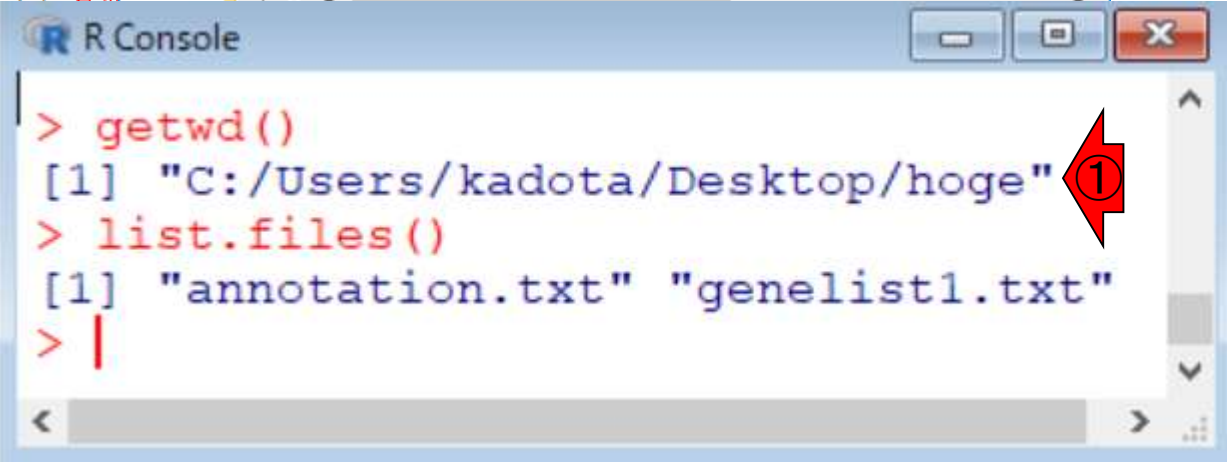
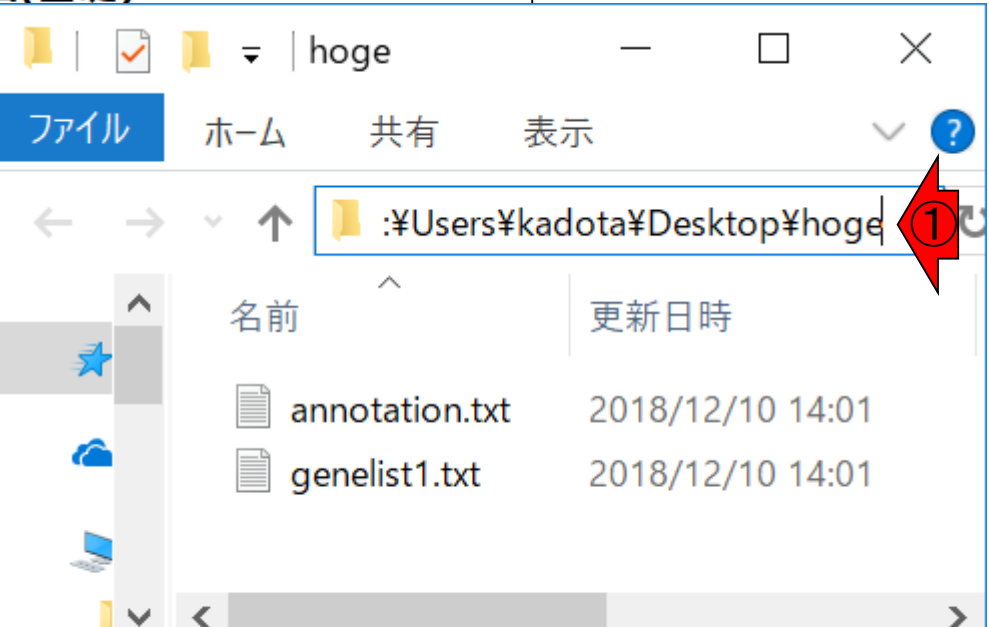
in_f1 <- "annotation.txt" #入力ファイル名を指定し
in_f2 <- "genelist1.txt" #入力ファイル名を指定し
out_f <- "hoge1.txt" #出力ファイル名を指定し
param <- 1 #アンテーションファイル

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

#本番
obj <- is.element(as.character(data[,param]), keywords)
out <- data[obj,] #objがTRUEとならばその行を抽出
dim(out) #オブジェクトoutの行数と列数を取得

#ファイルに保存
write.table(out, out_f, sep="\t", append=F, quote=F)

```



おさらい

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

```
in_f1 <- "annotation.txt" #入力ファイル名を指定してin_f1に格納(アノテーションファイル)
in_f2 <- "genelist1.txt" #入力ファイル名を指定してin_f2に格納(リストファイル)
out_f <- "hoge1.txt" #出力ファイル名を指定してout_fに格納
param <- 1 #アノテーションファイル中の検索したい列番号を指定

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

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

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



おさらい

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

```
in_f1 <- "annotation.txt"
in_f2 <- "genelist1.txt"
out_f <- "hoge1.txt"
param <- 1
```

#入力ファイルの読み込み

```
data <- read.table(in_f1, header=TRUE, sep="\t", as.is=TRUE)
keywords <- readLines(in_f2)
dim(data)
```

#本番

```
obj <- is.element(as.character(data[,1]), keywords)
out <- data[obj,]
dim(out)
```

#ファイルに保存

```
write.table(out, out_f, sep="\t", as.is=TRUE)
```

```
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
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files()
[1] "annotation.txt" "genelist1.txt"
> in_f1 <- "annotation.txt" #入力ファイル$
> in_f2 <- "genelist1.txt" #入力ファイル$
> out_f <- "hoge1.txt" #出力ファイル$
> param <- 1 #アノテーション$
>
> #入力ファイルの読み込み
> data <- read.table(in_f1, header=TRUE, sep="\t", as.is=TRUE) #sep="\t", quot$
> keywords <- readLines(in_f2) #in_f2で指定し$
> dim(data) #オブジェクトd$
[1] 11 4
> |
```

おさらい

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

```
in_f1 <- "annotation.txt"
in_f2 <- "genelist1.txt"
out_f <- "hoge1.txt"
param <- 1
```

#入力ファイルの読み込み

```
data <- read.table(in_f1, head
keywords <- readLines(in_f2)
dim(data)
```

#本番

```
obj <- is.element(as.character(d
out <- data[obj,]
dim(out)
```

#ファイルに保存

```
write.table(out, out_f, sep="\t"
```

```
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] 11 4
> data
  geneName accession description subcellular_location
1    gene1   hoge01  plasma_mem          nuclear
2    gene2   hoge02    hohinu             membrane
3    gene3   hoge03   agribio          endoplasmic
4    gene4   hoge04   genesis          endoplasmic
5    gene5   hoge05     kamo             membrane
6    gene6   hoge06   netteba           humei
7    gene7   hoge07   tebasaki          nuclear
8    gene8   hoge08     biiru             nuclear
9    gene9   hoge09   nihonshu          nuclear
10   gene10   hoge10   agene1            membrane
11   gene11   hoge11   iyaaaa            endoplasmic
> |
```

おさらい

①赤枠内のコードをコピー実行。こんな感じになります。② dataの中身を表示。③keywordsの中身を表示。スペルミスを防ぐため、タブ補完を有効利用しよう！（例:keyまで打ってからTabキーを押す）

1. 目的のタブ区切りテキストファイル(annota
(genelist1.txt)中のものが含まれる行全体を出力する

```
in_f1 <- "annotation.txt"
in_f2 <- "genelist1.txt"
out_f <- "hoge1.txt"
param <- 1
```

#入力ファイルの読み込み

```
data <- read.table(in_f1, header=TRUE)
keywords <- readLines(in_f2)
dim(data)
```

#本番

```
obj <- is.element(as.character(data[,1]), keywords)
out <- data[obj,]
dim(out)
```

#ファイルに保存

```
write.table(out, out_f, sep="\t")
```

```
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
> 
> 
> keywords
[1] "gene1" "gene7" "gene9"
> |
```

	gene name	accession	description	subcellular_location
1	gene1	hoge01	plasma_mem	nuclear
2	gene2	hoge02	hohinu	membrane
3	gene3	hoge03	agribio	endoplasmic
4	gene4	hoge04	genesis	endoplasmic
5	gene5	hoge05	kamo	membrane
6	gene6	hoge06	netteba	humei
7	gene7	hoge07	tebasaki	nuclear
8	gene8	hoge08	biiru	nuclear
9	gene9	hoge09	nihonshu	nuclear
10	gene10	hoge10	agenel	membrane
11	gene11	hoge11	iyaaaa	endoplasmic

目的のおさらい

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

```
in_f1 <- "annotation.txt"  
in_f2 <- "genelist.txt"  
out_f <- "hoge1.txt"  
param <- 1
```

```
#入力ファイルの読み込み  
data <- read.table(in_f1, header=TRUE, as.is=TRUE)  
keywords <- readLines(in_f2)  
dim(data)  
  
#本番  
obj <- is.element(as.character(data[,1]), keywords)  
out <- data[obj,]  
dim(out)  
  
#ファイルに保存  
write.table(out, out_f, sep="\t")
```

RGui (64-bit) Console Output:

	gene name	accession	description	subcellular_location
1	gene1	hoge01	plasma_mem	nuclear
2	gene2	hoge02	hohinu	membrane
3	gene3	hoge03	agribio	endoplasmic
4	gene4	hoge04	genesis	endoplasmic
5	gene5	hoge05	kamo	membrane
6	gene6	hoge06	netteba	humei
7	gene7	hoge07	tebasaki	nuclear
8	gene8	hoge08	biiru	nuclear
9	gene9	hoge09	nihonshu	nuclear
10	gene10	hoge10	agen1	membrane
11	gene11	hoge11	iyaaaa	endoplasmic

```
> keywords  
[1] "gene1" "gene7" "gene9"  
> |
```

目的のおさらい

①の中身である、②dataオブジェクトの、③1列目に対して、④の中身である、⑤keywordsオブジェクトの中の文字列と合致する、⑥行全体を出力するのが目的でした。

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

```
in_f1 <- "annotation.txt"  
in_f2 <- "genelist.txt"  
out_f <- "hoge1.txt"  
param <- 1
```



#入力ファイルの読み込み

```
data <- read.table(in_f1, header=TRUE, as.is=TRUE)  
keywords <- readLines(in_f2)  
dim(data)
```

#本番

```
obj <- is.element(as.character(data[,1]), keywords)  
out <- data[obj,]  
dim(out)
```

#ファイルに保存

```
write.table(out, out_f, sep="\t", as.is=TRUE)
```



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

	gene name	accession	description	subcellular location
1	gene1	hoge01	plasma_mem	nuclear
2	gene2	hoge02	hohinu	membrane
3	gene3	hoge03	agribio	endoplasmic
4	gene4	hoge04	genesis	endoplasmic
5	gene5	hoge05	kamo	membrane
6	gene6	hoge06	netteba	humei
7	gene7	hoge07	tebasaki	nuclear
8	gene8	hoge08	biiru	nuclear
9	gene9	hoge09	nihonshu	nuclear
10	gene10	hoge10	agen1	membrane
11	gene11	hoge11	iyaaaa	endoplasmic

```
> keywords  
[1] "gene1" "gene7" "gene9"  
> |
```

Contents

- 任意のキーワードを含む行を抽出するコードの解説の続き
 - おさらい、行列要素の抽出、`as.character`、`is.element`
 - 行列要素の抽出、`write.table`
- multi-FASTAファイルからの各種情報抽出
 - 基本情報取得(コンティグ数、配列長、N50、GC含量)
 - 課題1
 - コード内部の説明、`readDNAStringSet`、`width`、`sum`、`length`
 - サブセットの抽出(subsetting)、条件判定
 - 実践的な使い方(指定した長さ以上の配列を取得)
- Rパッケージの話
 - Bioconductor概観、Biostrings、推奨インストール手順、マニュアルなど
 - 課題2と課題3

見るだけ

①の中身である、②dataオブジェクトは数値行列です。この中から、③1列目の情報のみを抽出しているのが…

行列要素の抽出

1. 目的のタブ区切りテキストファイル(annotation.txt)中の第1列目をキーとして、リストファイル(genelist.txt)中のものが含まれる行全体を抽出する

```
in_f1 <- "annotation.txt"  
in_f2 <- "genelist.txt"  
out_f <- "hoge1.txt"  
param <- 1
```

```
#入力ファイルの読み込み  
data <- read.table(in_f1, header=TRUE, as.is=TRUE)  
keywords <- readLines(in_f2)  
dim(data)
```

```
#本番  
obj <- is.element(as.character(data[,1]), keywords)  
out <- data[obj,]  
dim(out)
```

```
#ファイルに保存  
write.table(out, out_f, sep="\t")
```

RGui (64-bit) console output:

gene	accession	description	subcellular_location
1	gene1	hoge01	plasma_mem nuclear
2	gene2	hoge02	hohinu membrane
3	gene3	hoge03	agribio endoplasmic
4	gene4	hoge04	genesis endoplasmic
5	gene5	hoge05	kamo membrane
6	gene6	hoge06	netteba humei
7	gene7	hoge07	tebasaki nuclear
8	gene8	hoge08	biiru nuclear
9	gene9	hoge09	nihonshu nuclear
10	gene10	hoge10	agen1 membrane
11	gene11	hoge11	iyaaaa endoplasmic

```
> keywords  
[1] "gene1" "gene7" "gene9"  
> |
```

見るだけ

①の中身である、②dataオブジェクトは数値行列です。この中から、③1列目の情報のみを抽出しているのが、④ data[,param]。

行列要素の抽出

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

```
in_f1 <- "annotation.txt" #入力ファイル名を指定してin_f1に格納(アノテーションファイル)
in_f2 <- "genelist.txt"   #入力ファイル名を指定してin_f2に格納(リストファイル)
out_f  <- "hoge1.txt"     #出力ファイル名を指定してout_fに格納
param <- 1                #アノテーションファイル中の検索したい列番号を指定

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

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

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

④ > data[,param]

見るだけ

行列要素の抽出

①の中身である、②dataオブジェクトは数値行列です。この中から、③1列目の情報のみを抽出しているのが、④ data[,param]。特定の列を抽出したい場合は、⑤コンマ(,)の右側に列番号を指定します。

1. 目的のタブ区切りテキストファイル(annotati (genelist1.txt)中のもが含まれる行全体を抽出

```
in_f1 <- "annotation.txt"
in_f2 <- "genelist1.txt"
out_f <- "hoge1.txt"
param <- 1
```



```
#入力ファイルの読み込み
data <- read.table(in_f1, header=TRUE, as.is=TRUE)
keywords <- readLines(in_f2)
dim(data)
```

```
#本番
obj <- is.element(as.character(dim(data)[1]), keywords)
out <- data[obj,]
dim(out)
```

```
#ファイルに保存
write.table(out, out_f, sep="\t", as.is=TRUE)
```



RGui (64-bit) window showing the R Console output. The console displays a table of data with 11 rows and 5 columns. The first row is highlighted. Below the table, the user enters 'keywords' and 'data[,param]' commands, and the console shows the output of these commands.

4	gene4	hoge04	genesis	endoplasmic
5	gene5	hoge05	kamo	membrane
6	gene6	hoge06	netteba	humei
7	gene7	hoge07	tebasaki	nuclear
8	gene8	hoge08	biiru	nuclear
9	gene9	hoge09	nihonshu	nuclear
10	gene10	hoge10	agen1	membrane
11	gene11	hoge11	iyaaaa	endoplasmic

```
> keywords
[1] "gene1" "gene7" "gene9"
> data[,param]
[1] gene1 gene2 gene3 gene4 gene5 gene6 gene7
[8] gene8 gene9 gene10 gene11
11 Levels: gene1 gene10 gene11 gene2 gene3 ... gene9
> |
```

as.character

1. 目的のタブ区切りテキストファイル(annotati
(genelist1.txt)中のものが含まれる行全

```
in_f1 <- "annotation.txt"
in_f2 <- "genelist1.txt"
out_f <- "hoge1.txt"
param <- 1
```

#入力ファイルの読み込み

```
data <- read.table(in_f1, header=
keywords <- readLines(in_f2)
dim(data)
```

#本番

```
obj <- is.element(as.character
out <- data[obj,]
dim(out)
```

#ファイルに保存

```
write.table(out, out_f, sep="\
```

①の中身である、②dataオブジェクトは数値行列です。この中から、③1列目の情報のみを抽出しているのが、④data[,param]。特定の列を抽出したい場合は、⑤コンマ(,)の右側に列番号を指定します。⑥は、④data[,param]に対してas.character関数(入力として与えるベクトルを文字列に変換する関数だと解釈すればよい)を実行した結果。

```
R Console
8   gene8   hoge08   biiru   nuclear
9   gene9   hoge09   nihonshu   nuclear
10  gene10  hoge10   agene1   membrane
11  gene11  hoge11   iyaaaa   endoplasmic
> keywords
[1] "gene1" "gene7" "gene9"
> data[,param]
[1] gene1 gene2 gene3 gene4 gene5 gene6 gene7
[8] gene8 gene9 gene10 gene11
11 Levels: gene1 gene10 gene11 gene2 gene3 ... gene9
> as.character(data[,param])
[1] "gene1" "gene2" "gene3" "gene4" "gene5"
[6] "gene6" "gene7" "gene8" "gene9" "gene10"
[11] "gene11"
> |
```


as.character

1. 目的のタブ区切りテキストファイル(annotati
(genelist1.txt)中のものが含まれる行全

```
in_f1 <- "annotation.txt"
in_f2 <- "genelist1.txt"
out_f <- "hoge1.txt"
param <- 1
```

#入力ファイルの読み込み

```
data <- read.table(in_f1, header=TRUE)
keywords <- readLines(in_f2)
dim(data)
```

#本番

```
obj <- is.element(as.character(data[,param]), keywords)
out <- data[obj,]
dim(out)
```

#ファイルに保存

```
write.table(out, out_f, sep="\t")
```

①の中身である、②dataオブジェクトは数値行列です。この中から、③1列目の情報のみを抽出しているのが、④ data[,param]。特定の列を抽出したい場合は、⑤コンマ(,)の右側に列番号を指定します。⑥は、④data[,param]に対して as.character関数(入力として与えるベクトルを文字列に変換する関数だと解釈すればよい)を実行した結果。こうすることで、⑦比較したいkeywordsの文字列ベクトルとデータの型が揃うこととなります。このようなas.ほにやら関数を用いることで、データを任意の型に変換できる場合が多いです。

```
8 gene8 hoge08 biiru nuclear
9 gene9 hoge09 nihonshu nuclear
10 gene10 hoge10 agene1 membrane
11 gene11 hoge11 iyaaaa endoplasmic
```

> keywords

```
[1] "gene1" "gene7" "gene9"
```

> data[,param]

```
[1] gene1 gene2 gene3 gene4 gene5 gene6 gene7
[8] gene8 gene9 gene10 gene11
11 Levels: gene1 gene10 gene11 gene2 gene3 ... gene9
```

> as.character(data[,param])

```
[1] "gene1" "gene2" "gene3" "gene4" "gene5"
[6] "gene6" "gene7" "gene8" "gene9" "gene10"
[11] "gene11"
```

> |

Contents

- 任意のキーワードを含む行を抽出するコードの解説の続き
 - おさらい、行列要素の抽出、`as.character`、`is.element`
 - 行列要素の抽出、`write.table`
- multi-FASTAファイルからの各種情報抽出
 - 基本情報取得(コンティグ数、配列長、N50、GC含量)
 - 課題1
 - コード内部の説明、`readDNAStringSet`、`width`、`sum`、`length`
 - サブセットの抽出(subsetting)、条件判定
 - 実践的な使い方(指定した長さ以上の配列を取得)
- Rパッケージの話
 - Bioconductor概観、Biostrings、推奨インストール手順、マニュアルなど
 - 課題2と課題3

is.element

①is.elementは、集合演算用関数(union, intersect, ...)群の中の1つ。②のベクトル中の各要素に対して、③で与えたベクトル中の要素が含まれるか否かをTRUE or FALSEで返す。④の行をコピー実行。

1. 目的のタブ区切りテキストファイル(annotati
(genelist1.txt)中のものが含まれる行全体を出力したい場合:

```
in_f1 <- "annotation.txt" #入力ファイル名を指定してin_f1に格納(アノテーションファイル)
in_f2 <- "genelist1.txt" #入力ファイル名を指定してin_f2に格納(リストファイル)
out_f <- "hoge1.txt" #出力ファイル名を指定してout_fに格納
param <- 1 #アノテーションファイル中の検索したい列番号を指定

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

#本番
obj <- is.element(as.character(data[,param]), keywords) #④件を満たすかどうかを判定した結果をobjに格納
out <- data[obj,] #objがTRUEとなる行のみ抽出した結果をoutに格納
dim(out) #オブジェクトoutの行数と列数を表示

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

is.element

1. 目的のタブ区切りテキストファイル(annotati
(genelist1.txt)中のものが含まれる行全

```
in_f1 <- "annotation.txt"  
in_f2 <- "genelist1.txt"  
out_f <- "hoge1.txt"  
param <- 1
```

#入力ファイルの読み込み

```
data <- read.table(in_f1, header  
keywords <- readLines(in_f2)  
dim(data)
```

#本番

```
obj <- is.element(as.character(d  
out <- data[obj,]  
dim(out)
```

#ファイルに保存

```
write.table(out, out_f, sep="\
```



①is.elementは、集合演算用関数(union, intersect, ...)群の中の一つ。②のベクトル中の各要素に対して、③で与えたベクトル中の要素が含まれるか否かをTRUE or FALSEで返す。④の行をコピー実行。⑤得られたobjオブジェクトの中身を表示。

```
R Console  
> keywords  
[1] "gene1" "gene7" "gene9"  
> data[,param]  
[1] gene1 gene2 gene3 gene4 gene5 gene6 gene7  
[8] gene8 gene9 gene10 gene11  
11 Levels: gene1 gene10 gene11 gene2 gene3 ... gene9  
> as.character(data[,param])  
[1] "gene1" "gene2" "gene3" "gene4" "gene5"  
[6] "gene6" "gene7" "gene8" "gene9" "gene10"  
[11] "gene11"  
> obj <- is.element(as.character(data[,param]), keyword$  
> obj  
[1] TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE  
[9] TRUE FALSE FALSE  
> |
```

is.element

1. 目的のタブ区切りテキストファイル(annotati
(genelist1.txt)中のものが含まれる行全

```
in_f1 <- "annotation.txt"  
in_f2 <- "genelist1.txt"  
out_f <- "hoge1.txt"  
param <- 1
```

#入力ファイルの読み込み

```
data <- read.table(in_f1, head  
keywords <- readLines(in_f2)  
dim(data)
```

#本番

```
obj <- is.element(as.character  
out <- data[obj,]  
dim(out)
```

#ファイルに保存

```
write.table(out, out_f, sep="\
```

①is.elementは、集合演算用関数(union, intersect, ...)群の中の一つ。②のベクトル中の各要素に対して、③で与えたベクトル中の要素が含まれるか否かをTRUE or FALSEで返す。④の行をコピー実行。⑤得られたobjオブジェクトの中身を表示。⑥objの要素数は⑦と同じで、⑦の要素と同一のところがTRUE、それ以外がFALSE。

```
R Console  
> keywords  
[1] "gene1" "gene7" "gene9"  
> data[,param]  
[1] gene1 gene2 gene3 gene4 gene5 gene6 gene7  
[8] gene8 gene9 gene10 gene11  
11 Levels: gene1 gene10 gene11 gene2 gene3 ... gene9  
> as.character(data[,param])  
[1] "gene1" "gene2" "gene3" "gene4" "gene5"  
[6] "gene6" "gene7" "gene8" "gene9" "gene10"  
[11] "gene11"  
> obj <- is.element(as.character(data[,param]), keywords)  
> obj  
[1] TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE  
[9] TRUE FALSE FALSE  
> |
```

is.element

1. 目的のタブ区切りテキストファイル(annotati
(genelist1.txt)中のものが含まれる行全

```
in_f1 <- "annotation.txt"  
in_f2 <- "genelist1.txt"  
out_f <- "hoge1.txt"  
param <- 1
```

#入力ファイルの読み込み

```
data <- read.table(in_f1, head  
keywords <- readLines(in_f2)  
dim(data)
```

#本番

```
obj <- is.element(as.character  
out <- data[obj,]  
dim(out)
```

#ファイルに保存

```
write.table(out, out_f, sep="\
```



```
R Console  
> keywords  
[1] "gene1" "gene7" "gene9"  
> data[,param]  
[1] gene1 gene2 gene3 gene4 gene5 gene6 gene7  
[8] gene8 gene9 gene10 gene11  
11 Levels: gene1 gene10 gene11 gene2 gene3 ... gene9  
> as.character(data[,param])  
[1] "gene1" "gene2" "gene3" "gene4" "gene5"  
[6] "gene6" "gene7" "gene8" "gene9" "gene10"  
[11] "gene11"  
> obj <- is.element(as.character(data[,param]), keywords)  
> obj  
[1] TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE  
[9] TRUE FALSE FALSE
```

①is.elementは、集合演算用関数(union, intersect, ...)群の中の1つ。②のベクトル中の各要素に対して、③で与えたベクトル中の要素が含まれるか否かをTRUE or FALSEで返す。④の行をコピー実行。⑤得られたobjオブジェクトの中身を表示。⑥objの要素数は⑦と同じで、⑦の要素と同一のところがTRUE、それ以外がFALSE。⑧のようなベクトルのことを論理値ベクトルといいます。



%in%も同じです

①is.elementは、集合演算用関数(union, intersect, ...)群の中の1つ。②のベクトル中の各要素に対して、③で与えたベクトル中の要素が含まれるか否かをTRUE or FALSEで返す。④の行をコピー実行。⑤得られたobjオブジェクトの中身を表示。⑥objの要素数は⑦と同じで、⑦の要素と同一のところがTRUE、それ以外がFALSE。⑧のようなベクトルのことを論理値ベクトルといいます。「is.element(x, y)」と「x %in% y」は同じです。比較的よく使いますし、よく見かけます。

1. 目的のタブ区切りテキストファイル(annotati (genelist1.txt)中のものが含まれる行全

```
in_f1 <- "annotation.txt"
in_f2 <- "genelist1.txt"
out_f <- "hoge1.txt"
param <- 1

#入力ファイルの読み込み
data <- read.table(in_f1, head=
keywords <- readLines(in_f2)
dim(data)

#本番
obj <- is.element(as.character
out <- data[obj,]
dim(out)

#ファイルに保存
write.table(out, out_f, sep="\
```

```
RGui
ファイル
R Console
> keywords
[1] "gene1" "gene7" "gene9"
> data[,param]
[1] gene1 gene2 gene3 gene4 gene5 gene6 gene7
[8] gene8 gene9 gene10 gene11
11 Levels: gene1 gene10 gene11 gene2 gene3 ... gene9
> as.character(data[,param])
[1] "gene1" "gene2" "gene3" "gene4" "gene5"
[6] "gene6" "gene7" "gene8" "gene9" "gene10"
[11] "gene11"
> obj <- is.element(as.character(data[,param]), keyword$
> obj
[1] TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE
[9] TRUE FALSE FALSE
```



Contents

- 任意のキーワードを含む行を抽出するコードの解説の続き
 - おさらい、行列要素の抽出、`as.character`、`is.element`
 - 行列要素の抽出、`write.table`
- multi-FASTAファイルからの各種情報抽出
 - 基本情報取得(コンティグ数、配列長、N50、GC含量)
 - 課題1
 - コード内部の説明、`readDNAStringSet`、`width`、`sum`、`length`
 - サブセットの抽出(subsetting)、条件判定
 - 実践的な使い方(指定した長さ以上の配列を取得)
- Rパッケージの話
 - Bioconductor概観、Biostrings、推奨インストール手順、マニュアルなど
 - 課題2と課題3

①このobjオブジェクトは、②の部分で用いられています。

行列要素の抽出

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

```
in_f1 <- "annotation.txt"  
in_f2 <- "genelist.txt"  
out_f <- "hoge1.txt"  
param <- 1
```

#入力ファイルの読み込み

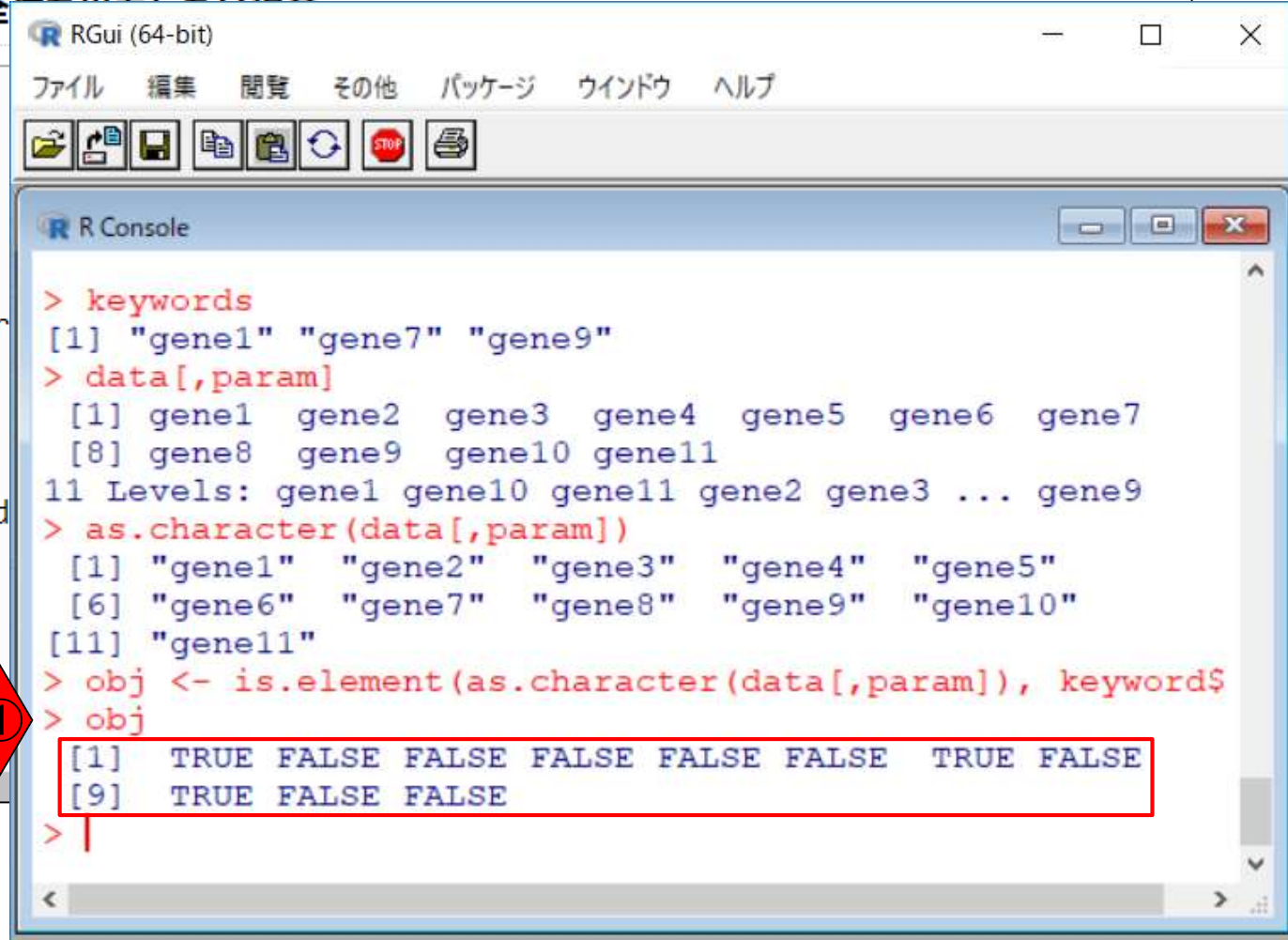
```
data <- read.table(in_f1, header=TRUE, as.is=TRUE)  
keywords <- readLines(in_f2)  
dim(data)
```

#本番

```
obj <- is.element(as.character(data[,1]), keywords)  
out <- data[obj,]  
dim(out)
```

#ファイルに保存

```
write.table(out, out_f, sep="\t")
```



```
> keywords  
[1] "gene1" "gene7" "gene9"  
> data[,param]  
[1] gene1 gene2 gene3 gene4 gene5 gene6 gene7  
[8] gene8 gene9 gene10 gene11  
11 Levels: gene1 gene10 gene11 gene2 gene3 ... gene9  
> as.character(data[,param])  
[1] "gene1" "gene2" "gene3" "gene4" "gene5"  
[6] "gene6" "gene7" "gene8" "gene9" "gene10"  
[11] "gene11"  
> obj <- is.element(as.character(data[,param]), keywords)  
> obj  
[1] TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE  
[9] TRUE FALSE FALSE
```

行列要素の抽出

①このobjオブジェクトは、②の部分で用いられています。②のobjは、行列dataから特定の行を抽出するために用いられている。①obj中の③TRUEの位置が、抽出したい行を指し示す④keywords中の要素の位置に相当することからも明らか。

1. 目的のタブ区切りテキストファイル(annotati
(genelist1.txt)中のものが含まれる行全

```
in_f1 <- "annotation.txt"  
in_f2 <- "genelist1.txt"  
out_f <- "hoge1.txt"  
param <- 1
```

#入力ファイルの読み込み

```
data <- read.table(in_f1, head=1, as.is=TRUE, sep="\t",  
dim(data)
```

#本番

```
obj <- is.element(as.character(data[,param]), keywords)  
out <- data[obj,]  
dim(out)
```

#ファイルに保存

```
write.table(out, out_f, sep="\t", as.is=TRUE)
```

```
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  
> keywords  
[1] "gene1" "gene7" "gene9"  
> data[,param]  
[1] gene1 gene2 gene3 gene4 gene5 gene6 gene7  
[8] gene8 gene9 gene10 gene11  
11 Levels: gene1 gene10 gene11 gene2 gene3 ... gene9  
> as.character(data[,param])  
[1] "gene1" "gene2" "gene3" "gene4" "gene5"  
[6] "gene6" "gene7" "gene8" "gene9" "gene10"  
[11] "gene11"  
> obj <- is.element(as.character(data[,param]), keywords)  
> obj  
[1] TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE  
[9] TRUE FALSE FALSE
```

行列要素の抽出

1. 目的のタブ区切りテキストファイル(annotati
(genelist1.txt)中のものが含まれる行全

```
in_f1 <- "annotation.txt"  
in_f2 <- "genelist1.txt"  
out_f <- "hoge1.txt"  
param <- 1
```

#入力ファイルの読み込み

```
data <- read.table(in_f1, head  
keywords <- readLines(in_f2)  
dim(data)
```

#本番

```
obj <- is.element(as.character(d  
out <- data[obj,]  
dim(out)
```

#ファイルに保存

```
write.table(out, out_f, sep="\
```



```
R Console  
> keywords  
[1] "gene1" "gene7" "gene9"  
> data[,param]  
[1] gene1 gene2 gene3 gene4 gene5 gene6 gene7  
[8] gene8 gene9 gene10 gene11  
11 Levels: gene1 gene10 gene11 gene2 gene3 ... gene9  
> as.character(data[,param])  
[1] "gene1" "gene2" "gene3" "gene4" "gene5"  
[6] "gene6" "gene7" "gene8" "gene9" "gene10"  
[11] "gene11"  
> obj <- is.element(as.character(data[,param]), keyword$  
> obj  
[1] TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE  
[9] TRUE FALSE FALSE
```

①このobjオブジェクトは、②の部分で用いられています。②のobjは、行列dataから特定の行を抽出するために用いられている。①obj中の③TRUEの位置が、抽出したい行を指し示す④keywords中の要素の位置に相当することからも明らか。行列dataから特定の行を抽出したい場合は、⑤コンマ(,)の左側に②のようなTRUE or FALSEからなる論理値ベクトルや、行番号情報を与えます。



行列要素の抽出

①赤枠内をコピー実行。これは行列dataから、objの要素がTRUEとなる行のみを抽出した結果を、outというオブジェクト名でまず保存している。そして、dim関数で行列outが、②3行×4列だという情報を得ている。

1. 目的のタブ区切りテキストファイル(annotati
(genelist1.txt)中のものが含まれる行全

```
in_f1 <- "annotation.txt"  
in_f2 <- "genelist1.txt"  
out_f <- "hoge1.txt"  
param <- 1
```

#入力ファイルの読み込み

```
data <- read.table(in_f1, header  
keywords <- readLines(in_f2)  
dim(data)
```

#本番

```
obj <- is.element(as.character(d  
out <- data[obj,]  
dim(out)
```

#ファイルに保存

```
write.table(out, out_f, sep="\t"
```

```
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] gene1 gene2 gene3 gene4 gene5 gene6 gene7  
[8] gene8 gene9 gene10 gene11  
11 Levels: gene1 gene10 gene11 gene2 gene3 ... gene9  
> as.character(data[,param])  
[1] "gene1" "gene2" "gene3" "gene4" "gene5"  
[6] "gene6" "gene7" "gene8" "gene9" "gene10"  
[11] "gene11"  
> obj <- is.element(as.character(data[,param]), keyword$  
> obj  
[1] TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE  
[9] TRUE FALSE FALSE  
> out <- data[obj,]  
> dim(out)  
[1] 3 4  
> |  
#objがTRUEとな$  
#オブジェクトo$
```

Contents

- 任意のキーワードを含む行を抽出するコードの解説の続き
 - おさらい、行列要素の抽出、`as.character`、`is.element`
 - 行列要素の抽出、`write.table`
- multi-FASTAファイルからの各種情報抽出
 - 基本情報取得(コンティグ数、配列長、N50、GC含量)
 - 課題1
 - コード内部の説明、`readDNAStringSet`、`width`、`sum`、`length`
 - サブセットの抽出(subsetting)、条件判定
 - 実践的な使い方(指定した長さ以上の配列を取得)
- Rパッケージの話
 - Bioconductor概観、Biostrings、推奨インストール手順、マニュアルなど
 - 課題2と課題3

write.table

①行列outの中身を表示。この②outが、③で指定した出力ファイル名からなる、④out_fに保存したいものです。

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

```
in_f1 <- "annotation.txt"
in_f2 <- "genelist.txt"
out_f <- "hoge1.txt"
param <- 1
```

#入力ファイルの読み込み

```
data <- read.table(in_f1, header=TRUE, as.is=TRUE)
keywords <- readLines(in_f2)
dim(data)
```

#本番

```
obj <- is.element(as.character(data[,param]), keywords)
out <- data[obj,]
dim(out)
```

#ファイルに保存

```
write.table(out, out_f, sep="\t", as.is=TRUE)
```

```
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

[6] "gene6" "gene7" "gene8" "gene9" "gene10"
[11] "gene11"
> obj <- is.element(as.character(data[,param]), keywords)
> obj
[1] TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE
[9] TRUE FALSE FALSE
> out <- data[obj,] #objがTRUEとなす
> dim(out) #オブジェクトo$
[1] 3 4
> out
  gene_name accession description subcellular_location
1   gene1     hoge01  plasma_mem             nuclear
7   gene7     hoge07   tebasaki             nuclear
9   gene9     hoge09   nihonshu             nuclear
> |
```


write.table

①write.table関数部分の全体像。ここで見えているような例題内で閉じている場合は特に意識する必要はないが、他の項目や例題のコードと組み合わせて利用したい場合には、ある程度オプションの意味を理解しておく必要がある。

1. 目的のタブ区切りテキストファイル(annotati
(genelist1.txt)中のものが含まれる行全体を出力したい場合：

```
in_f1 <- "annotation.txt" #入力ファイル名を指定してin_f1に格納(アノテーションファイル)
in_f2 <- "genelist1.txt" #入力ファイル名を指定してin_f2に格納(リストファイル)
out_f <- "hoge1.txt" #出力ファイル名を指定してout_fに格納
param <- 1 #アノテーションファイル中の検索したい列番号を指定

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

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

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

①

write.table

①write.table関数部分の全体像。②は、③と同じく区切り文字はタブにせよという宣言。

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

```
in_f1 <- "annotation.txt" #入力ファイル名を指定してin_f1に格納(アノテーションファイル)
in_f2 <- "genelist.txt"   #入力ファイル名を指定してin_f2に格納(リストファイル)
out_f  <- "hoge1.txt"     #出力ファイル名を指定してout_fに格納
param <- 1                #アノテーションファイル中の検索したい列番号を指定

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

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

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

①

②

③

write.table

①write.table関数部分の全体像。②は、③と同じく区切り文字はタブにせよという宣言。④は、⑤outオブジェクト中の行名情報を出力しない(False)という宣言。

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

```
in_f1 <- "annotation.txt" #入力ファイル名を指定してin_f1に格納(アノテーションファイル)
in_f2 <- "genelist.txt"   #入力ファイル名を指定してin_f2に格納(リストファイル)
out_f  <- "hoge1.txt"     #出力ファイル名を指定してout_fに格納
param <- 1                #アノテーションファイル中の検索したい列番号を指定

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

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

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

①

⑤

②

④

write.table

1. 目的のタブ区切りテキストファイル(annotati
(genelist1.txt)中のものが含まれる行全体を出カ

```
in_f1 <- "annotation.txt"  
in_f2 <- "genelist1.txt"  
out_f <- "hoge1.txt"  
param <- 1
```

#入力ファイルの読み込み

```
data <- read.table(in_f1, header=TRUE, sep="\t", quote="")#in_f1で指定したファイルの読み込み  
keywords <- readLines(in_f2)  
dim(data)
```

#本番

```
obj <- is.element(as.character(data[,param]), keywords)#条件を満たすかどうかを判定した結果をobjに格納  
out <- data[obj,]  
dim(out)
```

#ファイルに保存

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

①write.table関数部分の全体像。②は、③と同じく区切り文字はタブにせよという宣言。④は、⑤outオブジェクト中の行名情報を出カしない(False)という宣言。⑥それ以外のオプションについては作成当時の記憶があいまいだが、不都合が生じていないのでつけたままにしています。←そういうのもあるということです。

#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

#アンテーションファイル中の検索したい列番号を指定

#in_f1で指定したファイルの読み込み

#in_f2で指定したファイルの読み込み

#オブジェクトdataの行数と列数を表示

#条件を満たすかどうかを判定した結果をobjに格納

#objがTRUEとなる行のみ抽出した結果をoutに格納

#オブジェクトoutの行数と列数を表示

① ⑤ ② ⑥ ④

Contents

- 任意のキーワードを含む行を抽出するコードの解説の続き
 - おさらい、行列要素の抽出、`as.character`、`is.element`
 - 行列要素の抽出、`write.table`
- multi-FASTAファイルからの各種情報抽出
 - 基本情報取得(コンティグ数、配列長、N50、GC含量)
 - 課題1
 - コード内部の説明、`readDNAStringSet`、`width`、`sum`、`length`
 - サブセットの抽出(subsetting)、条件判定
 - 実践的な使い方(指定した長さ以上の配列を取得)
- Rパッケージの話
 - Bioconductor概観、Biostrings、推奨インストール手順、マニュアルなど
 - 課題2と課題3

FASTA形式

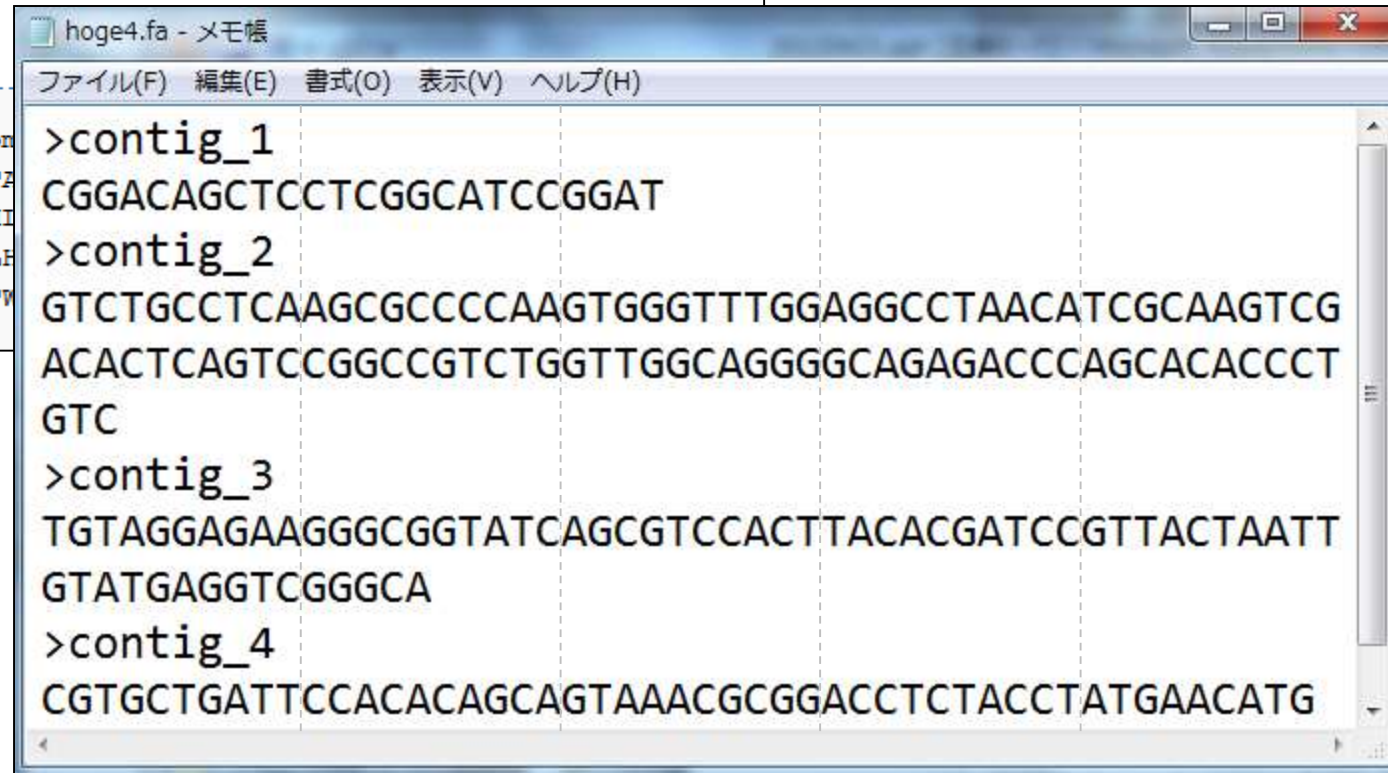
Rでmulti-FASTAファイルを読み込んで自在に解析できます。ゲノム配列解析≒FASTA形式ファイルの解析。ここでは全体像を完全に把握すべくhoge4.faファイル(ダウンロードは後程)を仮想ゲノム配列ファイルとして取り扱う

FASTAフォーマット [編集]

FASTAでは、シーケンスデータの記述形式としてFASTAフォーマットという形式を使う。FASTAフォーマットはブレンテキストである。1つのシーケンスのデータは、">"で始まる1行のヘッダ行と、2行目以降の実際のシーケンス文字列で構成される。ヘッダ行では、">"の次にシーケンスデータを識別するための文字列を記述し、続けてそのシーケンスデータを説明する文字列を記述する(両方とも省略してよい)。ヘッダ行の">"と識別文字列の間にスペースを入れてはいけない。FASTAフォーマットの全ての行は、80文字未満とすることが推奨される。">"で始まる別の行が出現すると、そこでシーケンスデータが区切れ、別のシーケンスデータが始まる。

FASTA ファイルフォーマットの例を示す。

```
>gi|5524211|gb|AAD44166.1| cytochrom  
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLI  
EWIWWGGSVDKATLNRFFAFHFILPFTMVA  
LLILILLLLLLLALLSPDMLGDPDNHMPAD  
GLMPFLHTSKHRSMMLRPLSQALFWTLTMD  
IENY
```



```
hoge4.fa - メモ帳  
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)  
>contig_1  
CGGACAGCTCCTCGGCATCCGGAT  
>contig_2  
GTCTGCCTCAAGCGCCCCAAGTGGGTTTGGAGGCCTAACATCGCAAGTCG  
ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCT  
GTC  
>contig_3  
TGTAGGAGAAGGGCGGTATCAGCGTCCACTTACACGATCCGTTACTAATT  
GTATGAGGTCGGGCA  
>contig_4  
CGTGCTGATTCCACACAGCAGTAAACGCGGACCTCTACCTATGAACATG
```

ゲノム配列

(Rで)塩基配列解析

(last modified 2018/03/29, since 2010)

このウェブページに従ってフリーな利用法(Windows)もありません。

What's new?

- アグリバイ
- 東大以外の
- 17:15より東
- Silhouetteス
- scores(シル)
- Silhouetteス
- NEW
- 「平成29年

- 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から取得](#) (last modified 2017/04/11) **NEW**
- イントロ | 一般 | [配列取得 | ゲノム配列 | BSgenome](#) (last modified 2015/04/22)
- イントロ | 一般 | [配列取得 | ゲノム配列 | 公共DBから取得](#) (last modified 2017/04/11) **NEW**

イントロ | 一般 | 配列取得 | ゲノム配列 | 公共DBから **NEW**

- UCSCの Sequence and Annotation Downloads (Tyner et al., Nucleic Acids Res., 2017)
 - ヒト; Human (H.sapiens)
 - ラット; Rat (R.norvegicus)
 - ネコ; Cat (F.catus)
 - ウサギ; Rabbit (O.cuniculus)
 - ニワトリ; Chicken (G.gallus)
 - イヌ; Dog (C.familiaris)
 - ウマ; Horse (E.caballus)
 - ...
- Helix Systems Scientific Databases (アップデートの日付順になっている。RefSeqやESTなど様々なデータベースを一度にみられる)
- イネ: [RAP-DB](#) (Sakai et al., Plant Cell Physiol., 2013)
 - 「ダウンロード」-「Genome assemblies」のところの [Download](#)。IRGSP-1.0_genome.fasta.gz (116MB程度)の圧縮ファイル。
- シロイヌナズナ: [The Arabidopsis Information Resource \(TAIR\)](#) (Lamesch et al., Nucleic Acids Res., 2012)
 - 「ダウンロード」-「Genes」-「TAIR10 genome release」-「TAIR10 chromosome files」の [TAIR10 chr_all.fas](#) (120MB程度)
- Ensembl Genomes (Yates et al., Nucleic Acids Res., 2016)
 - バクテリア (Bacteria)
 - 乳酸菌 (Lactobacillus casei 12A)
 - 乳酸菌 (Lactobacillus casei A2-362)
 - 乳酸菌 (Lactobacillus casei BL23)
 - 菌類 (Fungi)
 - 後生動物 (Metazoa)

基本情報取得

multi-FASTAファイルを読み込んで、トータルの配列長、染色体数(コンティグ数)、配列長の平均、中央値、最大値、最小値、N50、GC含量を計算した結果を返すコードを実行してみよう

入力: `hoge4.fa`

```
hoge4.fa - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
>contig_1
CGGACAGCTCCTCGGCATCCGGAT
>contig_2
GTCTGCCTCAAGCGCCCCAAGTGGGTTTGGAGGCCTAACATCGCAAGTCG
ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCT
GTC
>contig_3
TGTAGGAGAAGGGCGGTATCAGCGTCCACTTACACGATCCGTTACTAATT
GTATGAGGTCGGGCA
>contig_4
CGTGCTGATTCCACACAGCAGTAAACGCGGACCTCTACCTATGAACATG
```

出力: `hoge1.txt`

Total length (bp)	241
Number of contigs	4
Average length	60.25
Median length	57
Max length	103
Min length	24
N50	65
GC content	0.577



基本情報取得

①「... | FASTA形式 | 基本情報を取得」。②
コードの最初のほうに入力ファイルと出力ファイル
を記述するので、コピペ実行結果としてどう
いう名前のファイルが出力されるべきかわかる

- イントロ | NGS | アノテーション情報取得 | TxDb | [GenomicFeatures\(Lawrence 2013\)](#) (last modified 2016/04/22)
- イントロ | NGS | アノテーション情報取得 | TxDb | [GFF/GTF形式ファイルから](#) (last modified 2016/04/22)
- イントロ | NGS | 読み込み | BSgenome | [基本情報を取得](#) (last modified 2016/04/22)
- イントロ | NGS | 読み込み | FASTA形式 | [基本情報を取得](#) (last modified 2016/04/22)
- イントロ | NGS | 読み込み | FASTA形式 | [description行の記述を整形](#) (last modified 2014/04/05)
- イントロ | NGS | 読み込み | FASTQ形式 | [基礎](#) (last modified 2015/07/26)
- イントロ | NGS | 読み込み | FASTQ形式 | [応用](#) (last modified 2015/06/18)
- イントロ | NGS | 読み込み | FASTQ形式 | [description行の記述を整形](#) (last modified 2014/08/21)
- イントロ | NGS | 読み込み | [SAM/BAM形式](#) (last modified 2016/09/14)
- イントロ | NGS | 読み込み | [Illuminaの*](#)
- イントロ | NGS | 読み込み | [Illuminaの*](#)
- [イントロ | ファイル形式の変換 | について](#)
- イントロ | ファイル形式の変換 | [BAM ->](#)
- イントロ | ファイル形式の変換 | [FASTQ ->](#)
- イントロ | ファイル形式の変換 | [Genbank](#)
- イントロ | ファイル形式の変換 | [qseq ->](#)

イントロ | NGS | 読み込み | FASTA形式 | 基本情報を取得

multi-FASTAファイルを読み込んで、Total lengthやaverage lengthなどの各種情報取得を行うためのやり方を示します。例題6以降は、ヒトやマウスレベルの巨大ファイルを取り扱うためのコードです。具体的には、塩基数を整数(integer)ではなく実数(real number)として取り扱うためのas.numeric関数を追加しています。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

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

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

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

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

#本番(基本情報取得)
Total_len <- sum(width(fasta)) #配列の「トータルの長さ」を取得
Number_of_contigs <- length(fasta) #「配列数」を取得
Average_len <- mean(width(fasta)) #配列の「平均長」を取得
Median_len <- median(width(fasta)) #配列の「中央値」を取得
Max_len <- max(width(fasta)) #配列の長さの「最大値」を取得
```

ダウンロードと確認

イントロ | NGS | 読み込み | FASTA形式 | 基本情報

①例題の入力ファイル(hoge4.fa)をダウンロード。②R上で作業ディレクトリの確認。③貸与PCは、ここがstudent。④作業ディレクトリに解析したい入力ファイルがあることを確認。他のファイルがあってもよい

multi-FASTAファイルを読み込んで、Total lengthやaverage lengthなどの各種情報取得を行うためのやり方を示します。例題6以降は、ヒトやマウスレベルの巨大ファイルを取り扱うためのコードです。具体的には、塩基数を整数(integer)ではなく実数(real number)として取り扱うためのas.numeric関数を追加しています。

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

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:

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

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

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

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

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

```
R Console
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files()
[1] "hoge4.fa"
> |
```

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。

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

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

#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 <- readDNASTri

#本番(基本情報取得)
Total_len <- sum(wid
Number_of_contigs <-
Average_len <- mean(
Median_len <- median
Max len <- max(width
Min len <- min(width

#本番(N50情報取得)
sorted <- rev(sort(wid
obj <- (cumsum(sorted) >= Total_len*0.5)#条件を満たすかどうか
N50 <- sorted[obj][1] #objがTRUEとなる1番
```

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

このコマンド群を指定してin_fに格納
を指定してout_fに格納
み込み
み込み
指定したファイルの読み込み
「総長さ」を取得
「平均長」を取得
「中央値」を取得
「最大値」を取得
「最小値」を取得

R Console

```
> getwd()
[1] "C:/Users/
> list.files()
[1] "hoge4.fa"
>
>
> |
```

コピー	Ctrl+C
ペースト ②	Ctrl+V
コマンドのペースト	
コピー&ペースト	Ctrl+X
ウインドウの消去	Ctrl+L
全て選択	
パッファに出力	Ctrl+W

コピー後に、①list.files()で、②出力ファイルとして指定した、③hoge1.txtが作成されていることを確認

実行結果

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

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

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

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

#本番(基本情報取得)
Total_len <- sum(width(fasta)) #配列の「長さ」の総和
Number_of_contigs <- length(fasta) #「配列数」
Average_len <- mean(width(fasta)) #配列の「長さ」の平均
Median_len <- median(width(fasta)) #配列の「長さ」の中央値
Max_len <- max(width(fasta)) #配列の「長さ」の最大値
Min_len <- min(width(fasta)) #配列の「長さ」の最小値

#本番(N50情報取得)
sorted <- rev(sort(width(fasta))) #長さ情報
obj <- (cumsum(sorted) >= Total_len*0.5)#条件を満たす配列のインデックス
N50 <- sorted[obj][1] #objがTRUEの最初の要素
```

```
R Console
> tmp <- rbind(tmp, c("N50", N50))
> tmp <- rbind(tmp, c("GC content", GC_content$GC_content))
> write.table(tmp, out_f, sep="\t", append=F, $)
> list.files()
[1] "hoge1.txt" "hoge4.fa"
> tmp
      [,1]      [,2]
[1,] "Total length (bp)" "241"
[2,] "Number of contigs" "4"
[3,] "Average length" "60.25"
[4,] "Median length" "57"
[5,] "Max length" "103"
[6,] "Min length" "24"
[7,] "N50" "65"
[8,] "GC content" "0.576763485477178"
> |
```


実行結果

①の出力ファイルをテキストエディタやExcelで眺めてもよいが、②オブジェクトtmpの中身を③write.table関数を用いて出力しているだけなので、④R上で眺めている

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

```
N50 <- sorted[obj][1] #objがTRUEとなる1番最初の要素のみ抽出した結果を  
  
#本番(GC含量情報取得)  
hoge <- alphabetFrequency(fasta) #A,C,G,T,..の数を配列ごとにカウントした結果をh  
#CG <- rowSums(hoge[,2:3]) #C,Gの総数を計算してCGIに格納(2015年9月12日以前  
#ACGT <- rowSums(hoge[,1:4]) #A,C,G,Tの総数を計算してACGTに格納(2015年9月12  
CG <- apply(as.matrix(hoge[,2:3]), 1, sum) #C,Gの総数を計算してCGIに格納(2015年9月12日  
ACGT <- apply(as.matrix(hoge[,1:4]), 1, sum) #A,C,G,Tの総数を計算してACGTに格納(2015年9月12日  
GC_content <- sum(CG)/sum(ACGT) #トータルGC含量
```

#ファイルに保存

```
tmp <- NULL  
tmp <- rbind(tmp, c("Total length (bp)", Total_length))  
tmp <- rbind(tmp, c("Number of contigs", Number_of_contigs))  
tmp <- rbind(tmp, c("Average length", Average_length))  
tmp <- rbind(tmp, c("Median length", Median_length))  
tmp <- rbind(tmp, c("Max length", Max_length))  
tmp <- rbind(tmp, c("Min length", Min_length))  
tmp <- rbind(tmp, c("N50", N50))  
tmp <- rbind(tmp, c("GC content", GC_content))  
write.table(tmp, out_f, sep="\t", append=F, quote=F)
```

```
R Console  
> tmp <- rbind(tmp, c("N50", N50))  
> tmp <- rbind(tmp, c("GC content", GC_content))  
> write.table(tmp, out_f, sep="\t", append=F, quote=F)  
> list.files()  
[1] "hoge1.txt" "hoge4.fa"  
> tmp  
      [,1]      [,2]  
[1,] "Total length (bp)" "241"  
[2,] "Number of contigs" "4"  
[3,] "Average length"    "60.25"  
[4,] "Median length"     "57"  
[5,] "Max length"        "103"  
[6,] "Min length"        "24"  
[7,] "N50"                "65"  
[8,] "GC content"        "0.576763485477178"  
> |
```

入出力の関係を確認

①contig_1が最短、contig_2が最長。②N50の値は65 bpであり、③contig_3の長さと同じ

入力: hoge4.fa

ID	Length
contig_1	24
contig_2	103
contig_3	65
contig_4	49

```
hoge4.fa - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
>contig_1
CGGACAGCTCCTCGGCATCCGGAT
>contig_2
GTCTGCCTCAAGCGCCCCAAGTGGGTTTGGAGGCCTAACATCGCAAGTCG
ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCT
GTC
>contig_3
TGTAGGAGAAGGGCGGTATCAGCGTCCACTTACACGATCCGTTACTAATT
GTATGAGGTCGGGCA
>contig_4
CGTGCTGATTCCACACAGCAGTAAACGCGGACCTCTACCTATGAACATG
```

出力: hoge1.txt

Total length (bp)	241
Number of contigs	4
Average length	60.25
Median length	57
Max length	103
Min length	24
N50	65
GC content	0.577

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

N50

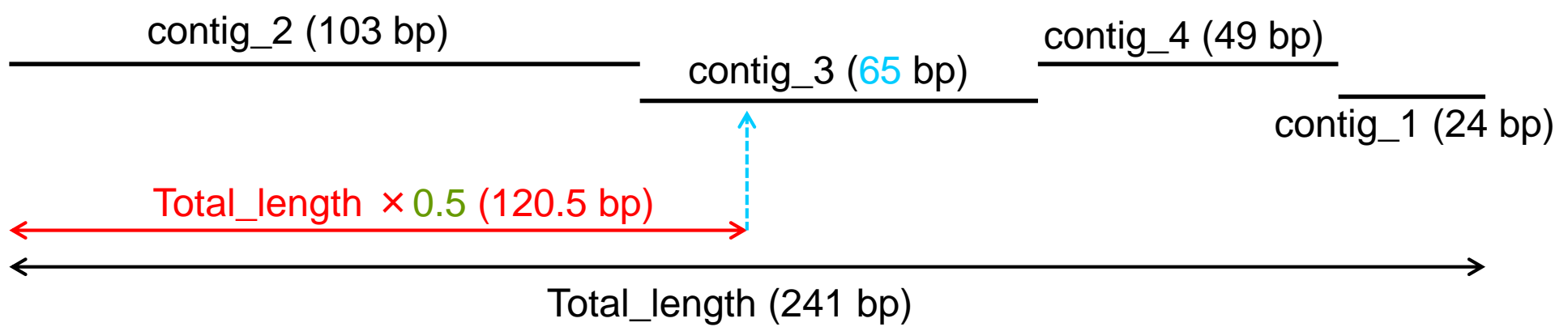
■ アセンブル結果の評価基準の1つ

- 長いコンティグから足していってTotal_lengthの50%に達したときのコンティグの長さ
- 一般に数値が大きいほどよい

出力: hoge1.txt

ID	Length
contig_1	24
contig_2	103
contig_3	65
contig_4	49

Total length (bp)	241
Number of contigs	4
Average length	60.25
Median length	57
Max length	103
Min length	24
N50	65
GC content	0.577



Contents

- 任意のキーワードを含む行を抽出するコードの解説の続き
 - おさらい、行列要素の抽出、`as.character`、`is.element`
 - 行列要素の抽出、`write.table`
- multi-FASTAファイルからの各種情報抽出
 - 基本情報取得(コンティグ数、配列長、N50、GC含量)
 - 課題1
 - コード内部の説明、`readDNAStringSet`、`width`、`sum`、`length`
 - サブセットの抽出(subsetting)、条件判定
 - 実践的な使い方(指定した長さ以上の配列を取得)
- Rパッケージの話
 - Bioconductor概観、Biostrings、推奨インストール手順、マニュアルなど
 - 課題2と課題3

課題1

左記のコードを、①hoge4.faの代わりに②hoge8.faを入力として実行し、1. 全配列長(配列長の総和)、2. N50の値、および3. GC含量を示せ。

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

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

#必要なパッケージをロー
library(Biostrings)

#入力ファイルの読み込み
fasta <- readDNASTri

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

#本番(N50情報取得)
sorted <- rev(sort(w
obj <- (cumsum(sorte
N50 <- sorted[obj][1

>contig_1
CACGTTGCATAT
>contig_2
NAGACAGCTCAACAAC
>contig_3
GTCTGCCTCAAGCGAAACAAGTGG AATTGGAGGCCTAACATCGCAAGTCG
ACACTCAGTCCGGNNGTCTGGTAAGCAGGGGCAGANNCCCAGCACACCT
>contig_4
CGTGCTGATANAACACAGCAGTAAACGCGGACCTCTACCTATGAACA
>contig_5
AGTGCTGATTCCACACAGCAGTAAACGCGGACCTCTACCTATG
>contig_6
AANNCGTTNGCAGNANACNNTG
>contig_7
TGTAGGAGAAGAAAGGTATCAGCGTCCACTTACACGATCCGTTACTAATT
GTATGAGGTCNNGCA
```

② hoge8.fa

課題1

左記のコードを、①hoge4.faの代わりに②hoge8.faを入力として実行し、1. 全配列長(配列長の総和)、2. N50の値、および3. GC含量を示せ。③hoge8.faはここにあります。

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

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

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

```
#入力ファイルの読み込み
fasta <- readDNASTri
```

```
#本番(基本情報取得)
```

```
Total_len <- sum(wid
Number_of_contigs <-
Average_len <- mean(
Median_len <- median
Max_len <- max(width
Min_len <- min(width
```

```
#本番(N50情報取得)
```

```
sorted <- rev(sort(w
obj <- (cumsum(sorte
N50 <- sorted[obj][1
```

```
>contig_1
CACGTTGCA
>contig_2
NAGACAGCT
>contig_3
GTCTGCCTC
ACACTCAGT
>contig_4
CGTGCTGAT
>contig_5
AGTGCTGAT
>contig_6
AANNCGTTN
>contig_7
TGTAGGAGA
GTATGAGGT
```

#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

講義日程 (2019年度)

- 2019年04月08日 (PC使用)
講義資料PDF
学会(国外): [ISCB](#)
学会(国内): [JSBi](#)
QAサイト: [Biostar \(Parnell et al., PLoS Comput Biol., 2011\)](#)
QAサイト: [SEQanswers \(Li et al., Bioinformatics, 2012\)](#)
学習教材: [バイオインフォマティクス人材育成のための講習会\(平成26-29年度\)](#)
学習教材: [\(Rで\)塩基配列解析](#)
学習教材: [\(Rで\)塩基配列解析のサブ RStudio](#)
- 2019年04月15日 (PC使用)
講義資料PDF
[\(Rで\)塩基配列解析](#)
[\(Rで\)塩基配列解析のサブ](#)
- 2019年04月22日 (PC使用)
講義資料PDF
[\(Rで\)塩基配列解析](#)
[hoge8.fa \(課題用\)](#)
- 2019年05月13日 (PC使用)
講義資料PDF

② hoge8.fa

CGCAAGTCG
GCACACCT

TGAACA

TG

TTACTAATT

課題1

①作業ディレクトリの確認と、②解析したいファイル(hoge8.fa)の存在確認。

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

```
in_f <- "hoge4.fa"  
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

```
library(Biostrings)
```

#入力ファイルの読み込み

```
fasta <- readFASTA(in_f)
```

#本番(基本情報)

```
Total_len <- sum(nchar(fasta))
```

```
Number_of_seqs <- length(fasta)
```

```
Average_len <- Total_len / Number_of_seqs
```

```
Median_len <- median(nchar(fasta))
```

```
Max_len <- max(nchar(fasta))
```

```
Min_len <- min(nchar(fasta))
```

#本番(N50情報)

```
sorted <- rev(sort(nchar(fasta)))
```

```
obj <- (cumsum(sorted) >= Total_len * 0.5)
```

```
N50 <- sorted[obj][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

```
> getwd()  
[1] "C:/Users/kadota/Desktop/hoge"  
> list.files()  
[1] "hoge8.fa"  
> |
```

課題1

①作業ディレクトリの確認と、②解析したいファイル(hoge8.fa)の存在確認。③ファイル、④新しいスクリプトで、推奨エディタを起動。

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

```
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

```
library(Biostrings)
```

#入力ファイルの読み込み

```
fasta <- readFASTA(in_f)
```

#本番(基本情報)

```
Total_len <- sum(nchar(fasta))
```

```
Number_of_seqs <- length(fasta)
```

```
Average_len <- Total_len / Number_of_seqs
```

```
Median_len <- median(nchar(fasta))
```

```
Max_len <- max(nchar(fasta))
```

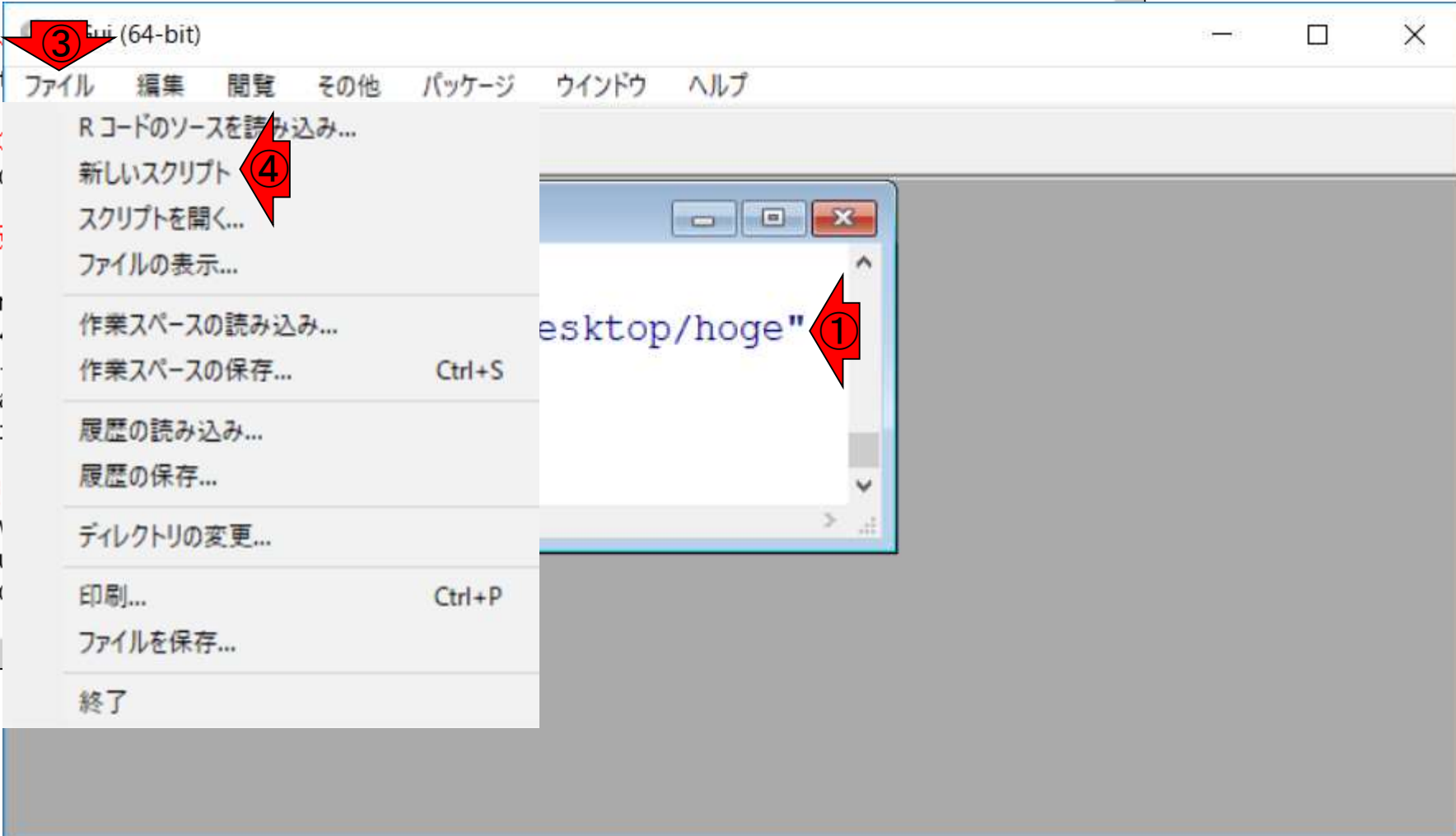
```
Min_len <- min(nchar(fasta))
```

#本番(N50情報)

```
sorted <- rev(sort(nchar(fasta)))
```

```
obj <- (cumsum(sorted) >= Total_len * 0.5)
```

```
N50 <- sorted[obj][1]
```



課題1

①Rエディタを起動後に、テンプレート(例題1)のコードをコピーしたところまで。

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

```
in_f <- "hoge4.fa"
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

```
library(Biostrings)
```

#入力ファイルの読み込み

```
fasta <- readDNASTringSet(in_f, format="fasta")
```

#本番(基本情報取得)

```
Total_len <- sum(width(fasta))
```

```
Number_of_seqs <- length(fasta)
```

```
Average_len <- Total_len / Number_of_seqs
```

```
Median_len <- median(width(fasta))
```

```
Max_len <- max(width(fasta))
```

```
Min_len <- min(width(fasta))
```

#本番(N50情報取得)

```
sorted <- rev(sort(width(fasta)))
```

```
obj <- (cumsum(sorted) >= Total_len * 0.5)
```

```
N50 <- sorted[obj][1]
```

RGui (64-bit) window showing the R Console with the following code and comments:

```
> getwd()
[1] "C:/Users/..."
> list.files()
[1] "hoge4.fa"
> |
#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:

#本番(基本情報取得)
Total_len <- sum(width(fasta)) #配列の「トータル」の長
```


課題1

①Rエディタを起動後に、テンプレート(例題1)のコードをコピーしたところまで。②必要最小限の変更を行い、

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

```
in_f <- "hoge4.fa"
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

```
library(Biostrings)
```

#入力ファイルの読み込み

```
fasta <- readDNASTringSet(in_f, format="fasta")
```

#本番(基本情報取得)

```
Total_len <- sum(width(fasta))
```

```
Number_of_seqs <- length(fasta)
```

```
Average_len <- Total_len / Number_of_seqs
```

```
Median_len <- median(width(fasta))
```

```
Max_len <- max(width(fasta))
```

```
Min_len <- min(width(fasta))
```

#本番(N50情報取得)

```
sorted <- rev(sort(width(fasta)))
```

```
obj <- (cumsum(sorted))
```

```
N50 <- sorted[which(obj >= Total_len * 0.5)[1]]
```

RGui (64-bit) window showing the R Console with the following code and comments:

```
> getwd()
[1] "C:/Users/..."
> list.files()
[1] "hoge4.fa"
> |
> |
#必要なパッケージをロード
library(Biostrings)

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

#本番(基本情報取得)
Total_len <- sum(width(fasta)) #配列の「トータル」の長
```

Red arrows indicate the changes made to the code: ① points to the change of the input file name from "hoge4.fa" to "hoge81.fa", and ② points to the change of the output file name from "hoge1.txt" to "hoge1.txt".

課題1

①Rエディタを起動後に、テンプレート(例題1)のコードをコピーしたところまで。②必要最小限の変更を行い、コード全体を反転させ、③右クリックで④。

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

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

#入力ファイル名を指定してin_flに格納
#出力ファイル名を指定してout_flに格納

#必要なパッケージ

```
library(Biostrings)
```

#入力ファイルの読み込み

```
fasta <- readFASTA(in_f)
```

#本番(基本情報)

```
Total_len <- sum(nchar(fasta))
```

```
Number_of_cor <- length(fasta)
```

```
Average_len <- Total_len / Number_of_cor
```

```
Median_len <- median(nchar(fasta))
```

```
Max_len <- max(nchar(fasta))
```

```
Min_len <- min(nchar(fasta))
```

#本番(N50情報)

```
sorted <- rev(sort(nchar(fasta)))
```

```
obj <- (cumsum(sorted) >= Average_len)
```

```
N50 <- sorted[obj][1]
```

RGui (64-bit) window showing the R Console and R Editor. The R Console contains the following code:

```
> getwd()
[1] "C:/Users/.../R/RGui/bin/x64"
> list.files("hoge4.fa")
[1] "hoge4.fa"
> |
> |
```

The R Editor shows the following code:

```
tmp <- readFASTA("hoge4.fa")
tmp <- readFASTA("hoge4.fa")
tmp <- readFASTA("hoge4.fa")
tmp <- readFASTA("hoge4.fa")
tmp <- readFASTA("hoge4.fa")
tmp <- readFASTA("hoge4.fa")
tmp <- readFASTA("hoge4.fa")
tmp <- readFASTA("hoge4.fa")
tmp <- readFASTA("hoge4.fa")
tmp <- readFASTA("hoge4.fa")
tmp <- readFASTA("hoge4.fa")
write.table(tmp, "hoge1.txt", as.is=T, quote=F, row.names=F, col.names=F)
```

A context menu is open over the code in the R Editor, with the following options:

- カーソル行または選択中の R コードを実行 (Ctrl+R)
- やり直し (Ctrl+Z)
- カット (Ctrl+X)
- コピー (Ctrl+C)
- ペースト (Ctrl+V)
- 消去
- 全て選択 (Ctrl+A)

Red arrows indicate the steps: ① Copying the code, ② Pasting it into the editor, ③ Right-clicking, and ④ Selecting Ctrl+R.

課題1

①Rエディタを起動後に、テンプレート(例題1)のコードをコピーしたところまで。②必要最小限の変更を行い、コード全体を反転させ、③右クリックで④。コード全体を反転させた後に⑤「Ctrl + R」でもよいし、⑥を押してもよい。

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

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

#入力ファイル名を指
#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

```
library(Biostrings)
```

#入力ファイルの読み込み

```
fasta <- readFasta(in_f)
```

#本番(基本情報)

```
Total_len <- sum(nchar(fasta))
Number_of_seqs <- length(fasta)
Average_len <- Total_len / Number_of_seqs
Median_len <- median(nchar(fasta))
Max_len <- max(nchar(fasta))
Min_len <- min(nchar(fasta))
```

#本番(N50情報)

```
sorted <- rev(sort(nchar(fasta)))
obj <- (cumsum(sorted) >= Total_len * 0.5)
N50 <- sorted[obj][1]
```

The screenshot shows the RGui interface. The R Console window displays the following code:

```
> getw
[1] "c
> list
[1] "h
> |
tmp <- N
tmp <- rbi
tmp <- rbi
tmp <- rbi
tmp <- rbi
tmp <- rbi
tmp <- rbi
tmp <- rbi
tmp <- rbi
write.tabl
```

The R Editor window shows the same code being edited. A context menu is open over the code, with the following items:

- カーソル行または選択中の R コードを実行 (Ctrl+R)
- やり直し (Ctrl+Z)
- カット (Ctrl+X)
- コピー (Ctrl+C)
- ペースト (Ctrl+V)
- 消去
- 全て選択 (Ctrl+A)

Red arrows indicate the sequence of actions: ① points to the RGui menu, ② points to the copy icon, ③ points to the select all icon, ④ points to the right-click action, ⑤ points to the Ctrl+R shortcut, and ⑥ points to the save icon in the RGui menu.

課題1

実行後にtmpの中身を表示させた状態。毎年必ず間違ふヒトがいて残念ですが、入力ファイルと見比べれば、得られた結果の妥当性がわかるはずです。

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

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

```
#入力ファイル名を指定してin_flに格納
#出力ファイル名を指定してout_flに格納
```

```
#必要なパッケージ
```

```
library(Biostrings)
```

```
#入力ファイルの読み込み
```

```
fasta <- readFasta(in_f)
```

```
#本番(基本情報)
```

```
Total_len <- sum(widths(fasta))
```

```
Number_of_contigs <- length(fasta)
```

```
Average_len <- Total_len / Number_of_contigs
```

```
Median_len <- median(widths(fasta))
```

```
Max_len <- max(widths(fasta))
```

```
Min_len <- min(widths(fasta))
```

```
#本番(N50情報)
```

```
sorted <- rev(sort(widths(fasta)))
```

```
obj <- (cumsum(sorted) >= Total_len * 0.5)
```

```
N50 <- sorted[obj][1]
```

	[,1]	[,2]
[1,]	"Total length (bp)"	"304"
[2,]	"Number of contigs"	"7"
[3,]	"Average length"	"43.4285714285714"
[4,]	"Median length"	"43"
[5,]	"Max length"	"99"
[6,]	"Min length"	"12"
[7,]	"N50"	"65"
[8,]	"GC content"	"0.501730103806228"

Contents

- 任意のキーワードを含む行を抽出するコードの解説の続き
 - おさらい、行列要素の抽出、`as.character`、`is.element`
 - 行列要素の抽出、`write.table`
- multi-FASTAファイルからの各種情報抽出
 - 基本情報取得(コンティグ数、配列長、N50、GC含量)
 - 課題1
 - コード内部の説明、`readDNAStringSet`、`width`、`sum`、`length`
 - サブセットの抽出(subsetting)、条件判定
 - 実践的な使い方(指定した長さ以上の配列を取得)
- Rパッケージの話
 - Bioconductor概観、Biostrings、推奨インストール手順、マニュアルなど
 - 課題2と課題3

①「… | FASTA形式 | 基本情報を取得」。②例題1のコード内部を重要な部分に絞って説明します。課題1で利用したものと同じです。

コード内部の説明

- ・ [イントロ](#) | [NGS](#) | [アノテーション情報取得](#) | [TxDb](#) | [GenomicFeatures\(Lawrence 2013\)](#) (last modified 2015/02/19)
- ・ [イントロ](#) | [NGS](#) | [アノテーション情報取得](#) | [TxDb](#) | [GFF/GTF形式ファイルから](#) (last modified 2016/02/09)
- ・ [イントロ](#) | [NGS](#) | [読み込み](#) | [BSgenome](#) | [基本情報を取得](#) (last modified 2016/04/22)
- ・ [イントロ](#) | [NGS](#) | [読み込み](#) | [FASTA形式](#) | [基本情報を取得](#) (last modified 2016/04/22)
- ・ [イントロ](#) | [NGS](#) | [読み込み](#) | [FASTA形式](#) | [description行の記述を整形](#) (last modified 2014/04/05)
- ・ [イントロ](#) | [NGS](#) | [読み込み](#) | [FASTQ形式](#) | [基礎](#) (last modified 2015/07/26)
- ・ [イントロ](#) | [NGS](#) | [読み込み](#) | [FASTQ形式](#) | [応用](#) (last modified 2015/06/18)
- ・ [イントロ](#) | [NGS](#) | [読み込み](#) | [FASTQ形式](#) | [description行の記述を整形](#) (last modified 2014/08/21)
- ・ [イントロ](#) | [NGS](#) | [読み込み](#) | [SAM/BAM形式](#) (last modified 2016/09/14)
- ・ [イントロ](#) | [NGS](#) | [読み込み](#) | [Illuminaの*](#)
- ・ [イントロ](#) | [NGS](#) | [読み込み](#) | [Illuminaの*](#)
- ・ [イントロ](#) | [ファイル形式の変換](#) | [BAM](#) →
- ・ [イントロ](#) | [ファイル形式の変換](#) | [FASTQ](#) →
- ・ [イントロ](#) | [ファイル形式の変換](#) | [Genbank](#)
- ・ [イントロ](#) | [ファイル形式の変換](#) | [qseq](#)

イントロ | NGS | 読み込み | FASTA形式 | 基本情報を取得

multi-FASTAファイルを読み込んで、Total lengthやaverage lengthなどの各種情報取得を行うためのやり方を示します。例題6以降は、ヒトやマウスレベルの巨大ファイルを取り扱うためのコードです。具体的には、塩基数を整数(integer)ではなく実数(real number)として取り扱うためのas.numeric関数を追加しています。「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピー。

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:

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

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

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

#本番(基本情報取得)
Total_len <- sum(width(fasta)) #配列の「トータルの長さ」を取得
Number_of_contigs <- length(fasta) #「配列数」を取得
Average_len <- mean(width(fasta)) #配列の「平均長」を取得
Median_len <- median(width(fasta)) #配列の「中央値」を取得
Max_len <- max(width(fasta)) #配列の長さの「最大値」を取得
```


コード内部の説明

左上に拡大表示しただけです。①multi-FASTAファイルを、②readDNASTringSet関数の、③入力として読み込んだ結果を、④fastaというオブジェクト名で格納しています。

1. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して

```
in_f <- "hoge4.fa" #① #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

#④(基本情報取得) #② #③
Total_len <- sum(width(fasta)) #配列の「トータル長さ」を取得
Number_of_contigs <- length(fasta) #「配列数」を取得
Average_len <- mean(width(fasta)) #配列の「平均長」を取得
Median_len <- median(width(fasta)) #配列の「中央値」を取得
Max_len <- max(width(fasta)) #配列の長さの「最大値」を取得
Min_len <- min(width(fasta)) #配列の長さの「最小値」を取得

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

コード内部の説明

左上に拡大表示しただけです。①multi-FASTAファイルを、②readDNAStrngSet関数の、③入力として読み込んだ結果を、④fastaというオブジェクト名で格納しています。①hoge4.faの中身は⑤ですので、④のfastaオブジェクトは⑤の情報を含んでいるはずですが。

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

```
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 <- readDNAStrngSet(in_f, format="fasta")#in_fで指定したファイルを読み込み
```



```
#基本情報取得
```

```
Total_len <- sum(width(fasta))
Number_of_contigs <- length(fasta)
Average_len <- mean(width(fasta))
Median_len <- median(width(fasta))
Max_len <- max(width(fasta))
Min_len <- min(width(fasta))
```

```
#本番(N50情報取得)
```

```
sorted <- rev(sort(width(fasta)))
obj <- (cumsum(sorted) >= Total_len)
N50 <- sorted[obj][1]
```

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

コード内部の説明

赤枠のfastaオブジェクト作成部分までをコピー実行。R
コンソール画面の見栄えがヒトによって異なると思いますが、
入力ファイル(hoge4.fa)が存在する場所に正しく作業ディレクトリの変更ができていればOK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

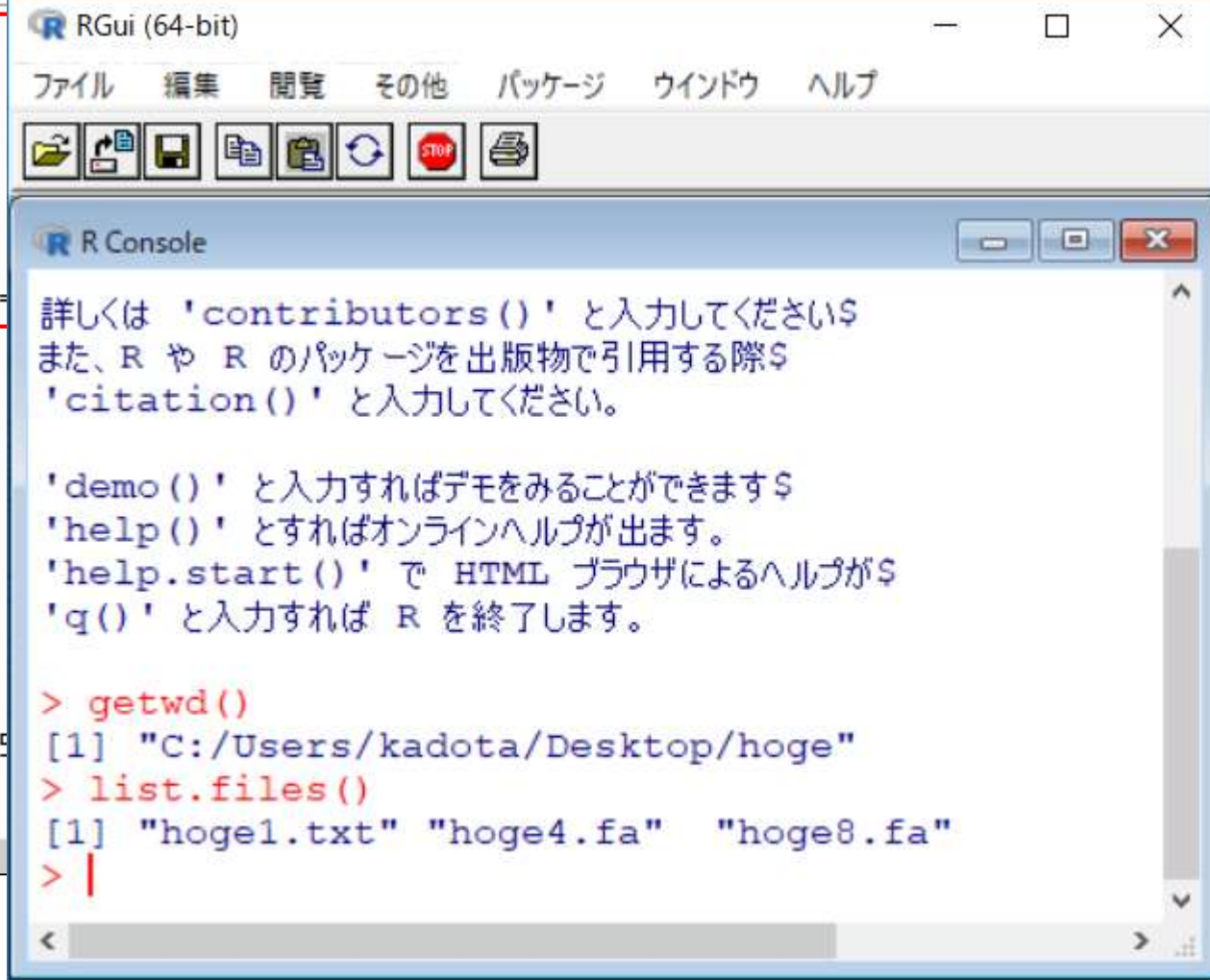
```
in_f <- "hoge4.fa"
out_f <- "hoge1.txt"

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

#本番(N50情報取得)
sorted <- rev(sort(width(fasta)))
obj <- (cumsum(sorted) >= Total_len*0.5)
N50 <- sorted[obj][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
詳しくは 'contributions()' 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$
'citation()' と入力してください。

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

> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files()
[1] "hoge1.txt" "hoge4.fa" "hoge8.fa"
> |
```

赤枠のfastaオブジェクト作成部分までをコピー実行。概ねこんな感じになります。

コード内部の説明

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

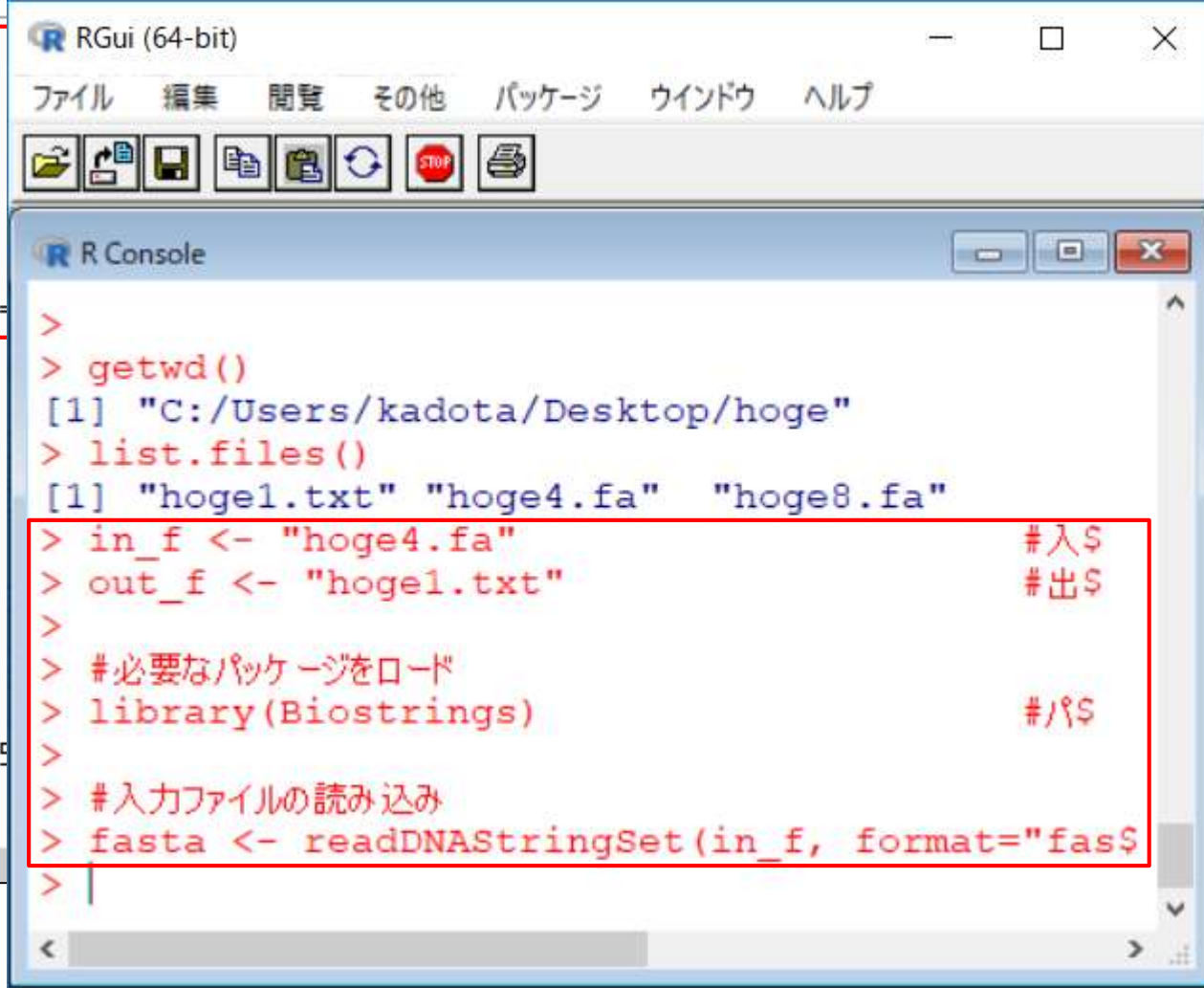
```
in_f <- "hoge4.fa"
out_f <- "hoge1.txt"

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

#本番(N50情報取得)
sorted <- rev(sort(width(fasta)))
obj <- (cumsum(sorted) >= Total_len*0.5)
N50 <- sorted[obj][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
>
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files()
[1] "hoge1.txt" "hoge4.fa" "hoge8.fa"
> in_f <- "hoge4.fa" #入$
> out_f <- "hoge1.txt" #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="fasta")
>
> |
```


コード内部の説明

赤枠のfastaオブジェクト作成部分までをコピー実行。概ねこんな感じになります。①fastaオブジェクトの中身を見てみましょう。

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

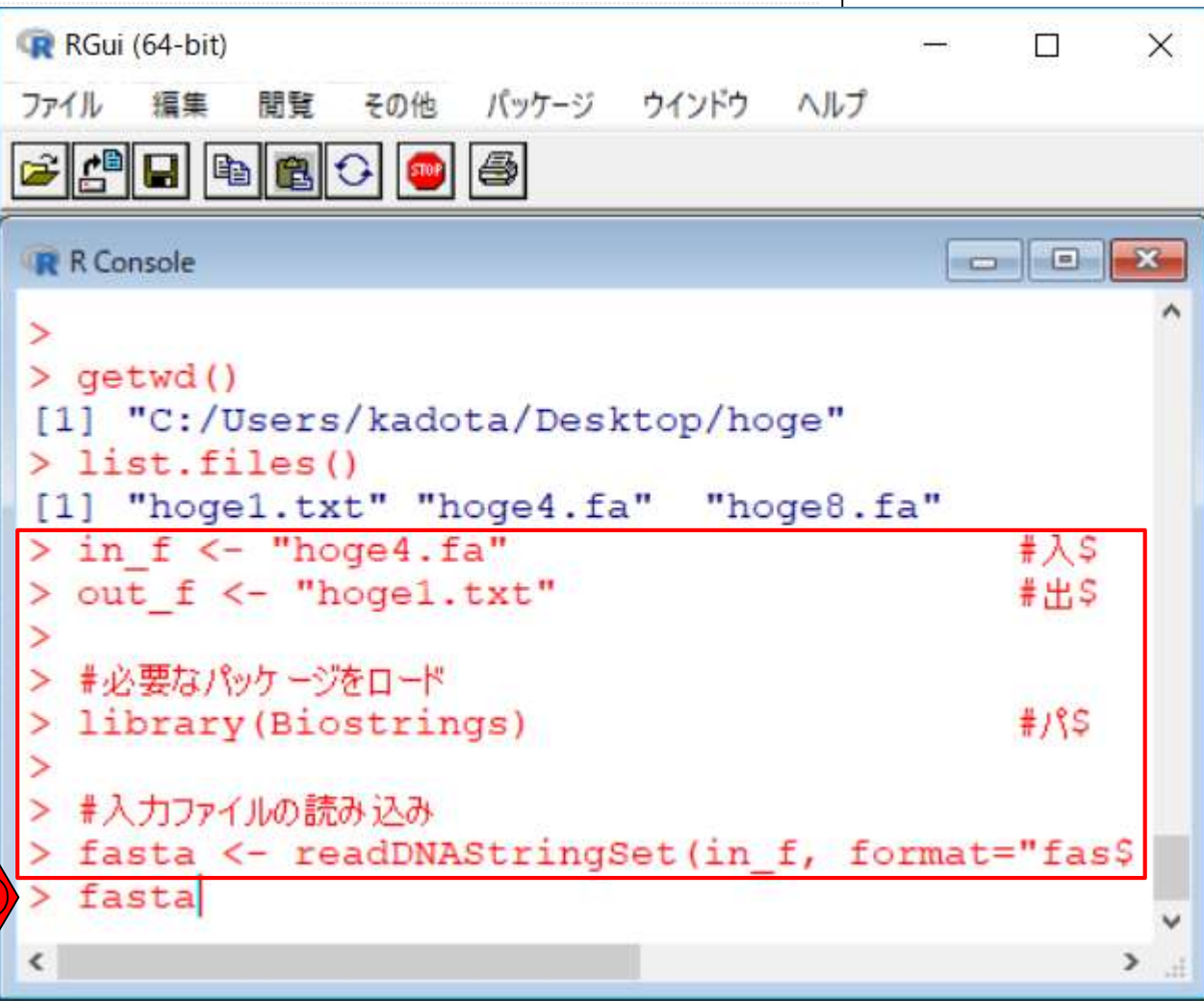
```
in_f <- "hoge4.fa"
out_f <- "hoge1.txt"

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

#本番(N50情報取得)
sorted <- rev(sort(width(fasta)))
obj <- (cumsum(sorted) >= Total_len*0.5)
N50 <- sorted[obj][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
>
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files()
[1] "hoge1.txt" "hoge4.fa" "hoge8.fa"
> in_f <- "hoge4.fa" #入$
> out_f <- "hoge1.txt" #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="fasta")
> fasta
```


コード内部の説明

赤枠のfastaオブジェクト作成部分までをコピー実行。概ねこんな感じになります。①fastaオブジェクトの中身を見てみましょう。概ねこんな感じになります。

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

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

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

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



```
>
> #必要なパッケージをロード
> library(Biostrings) #パッケージ
>
> #入力ファイルの読み込み
> fasta <- readDNASTringSet(in_f, format="fasta")
> fasta
A DNASTringSet instance of length 4
  width seq          names
[1]   24 CGGACA...CCGGAT contig_1
[2]  103 GTCTGC...CCTGTC contig_2
[3]   65 TGTAGG...CGGGCA contig_3
[4]   49 CGTGCT...AACATG contig_4
> |
```

①fastaオブジェクトの中身と、②入力ファイルの中身が対応していることがわかります。

コード内部の説明

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

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

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

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

①

```
>contig_1
CGGACAGCTCCTCGGCATCCGGAT
>contig_2
GTCTGCCTCAAGCGCCCCAAGTGGGTTTGGAGGCCTAACATCGCAAGTCG
ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCT
GTC
>contig_3
>
TGTAGGAGAAGGGCGGTATCAGCGTCCACTTACACGATCCGTTACTAATT
GTATGAGGTCGGGCA
>contig_4
>
CGTGCTGATTCCACACAGCAGTAAACGCGGACCTCTACCTATGAACATG
> fasta
A DNASTringSet instance of length 4
  width seq          names
[1]   24 CGGACA...CCGGAT contig_1
[2]  103 GTCTGC...CCTGTC contig_2
[3]   65 TGTAGG...CGGGCA contig_3
[4]   49 CGTGCT...AACATG contig_4
> |
```

②

コード内部の説明

①は、全部で4配列からなることを示します。②の4は、4番目の配列という意味。③の1は、1番目の配列という意味です。

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multi-FASTAファイル([hoge4.fa](#))の場合:

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

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

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

③

②

```
>contig_1
CGGACAGCTCCTCGGCATCCGGAT
>contig_2
GTCTGCCTCAAGCGCCCCAAGTGGGTTTGGAGGCCTAACATCGCAAGTCG
ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCT
GTC
>contig_3
>
>contig_4
CGTGCTGATTCCACACAGCAGTAAACGCGGACCTCTACCTATGAACATG
> fasta
A DNASTringSet instance of length 4
  width seq          names
[1]    24 CGGACA...CCGGAT contig_1
[2]   103 GTCTGC...CCTGTC contig_2
[3]    65 TGTAGG...CGGGCA contig_3
[4]    49 CGTGCT...AACATG contig_4
> |
```

①

コード内部の説明

①width列の数値情報は、配列長に相当します。②4番目の配列は49塩基、③1番目の配列は24塩基なので妥当ですね。

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

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

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

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

```
>contig_1
CGGACAGCTCCTCGGCATCCGGAT
>contig_2
GTCTGCCTCAAGCGCCCCAAGTGGGTTTGGAGGCCTAACATCGCAAGTCG
ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCT
GTC
>contig_3
>
>TGTAGGAGAAGGGCGGTATCAGCGTCCACTTACACGATCCGTTACTAATT
GTATGAGGTCGGGCA
>contig_4
>
>CGTGCTGATTCCACACAGCAGTAAACGCGGACCTCTACCTATGAACATG
>
> fasta
A DNASTringSet instance of length 4
  width seq          names
[1]    24 CGGACA...CCGGAT contig_1
[2]   103 GTCTGC...CCTGTC contig_2
[3]    65 TGTAGG...CGGGCA contig_3
[4]    49 CGTGCT...AACATG contig_4
> |
```

①が、②DNAMStringSetという形式で格納されている、③
fastaオブジェクトの主な中身である塩基配列情報。

コード内部の説明

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

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

#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 <- readDNAMStringSet(in_f, format="fasta")

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

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

```
> contig_1
CGGACAGCTCCTCGGCATCCGGAT
> contig_2
GTCTGCCTCAAGCGCCCAAGTGGGTTTGGAGGCCTAACATCGCAAGTCG
ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCT
GTC
> contig_3
>
> contig_4
CGTGCTGATTCCACACAGCAGTAAACGCGGACCTCTACCTATGAACATG
> fasta
A DNAMStringSet instance of length 4
  width seq          names
[1]   24 CGGACA...CCGGAT contig_1
[2]  103 GTCTGC...CCTGTC contig_2
[3]   65 TGTAGG...CGGGCA contig_3
[4]   49 CGTGCT...AACATG contig_4
> |
```


コード内部の説明

①が、②DNAMStringSetという形式で格納されている、③ fastaオブジェクトの主な中身である塩基配列情報。Rコンソール画面の横幅次第で見栄え(特に塩基配列の最初と最後の塩基数)が異なりますが、気にしなくてよい。

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

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

#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 <- readDNAMStringSet(in_f, format="fasta")

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

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

```
RGui (C...)
ファイル
R Con...
>contig_1
CGGACAGCTCCTCGGCATCCGGAT
>contig_2
GTCTGCCCTCAAGCGCCCCAAGTGGGTTTGGAGGCCTAACATCGCAAGTCG
ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCT
GTC
>contig_3
TGTAGGAGAAGGGCGGTATCAGCGTCCACTTACACGATCCGTTACTAATT
GTATGAGGTCGGGCA
>contig_4
CGTGCTGATTCCACACAGCAGTAAACGCGGACCTCTACCTATGAACATG
> fasta
A DNAMStringSet instance of length 4
  width seq          names
[1]   24 CGGACAG...TCCGGAT contig_1
[2]  103 GTCTGCC...CCCTGTC contig_2
[3]   65 TGTAGGA...TCGGGCA contig_3
[4]   49 CGTGCTG...GAACATG contig_4
> |
```



Contents

- 任意のキーワードを含む行を抽出するコードの解説の続き
 - おさらい、行列要素の抽出、`as.character`、`is.element`
 - 行列要素の抽出、`write.table`
- multi-FASTAファイルからの各種情報抽出
 - 基本情報取得(コンティグ数、配列長、N50、GC含量)
 - 課題1
 - コード内部の説明、`readDNAStringSet`、`width`、`sum`、`length`
 - サブセットの抽出(subsetting)、条件判定
 - 実践的な使い方(指定した長さ以上の配列を取得)
- Rパッケージの話
 - Bioconductor概観、Biostrings、推奨インストール手順、マニュアルなど
 - 課題2と課題3

width

①fastaオブジェクト中の、②配列長情報は、③列名がwidthとなっていることから想像できますが…

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

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

#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, from

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

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

```
RGui (64-bit)
ファイル
R Console
> contig_1
CGGACAGCTCCTCGGCATCCGGAT
> contig_2
GTCTGCCTCAAGCGCCCCAAGTGGGTTTGGAGGCCTAACATCGCAAGTCG
ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCT
GTC
> contig_3
TGTAGGAGAAGGGCGGTATCAGCGTCCACTTACACGATCCGTTACTAATT
GTATGAGGTCGGGCA
> contig_4
CGTGCTGATTCCACACAGCAGTAAACGCGGACCTCTACCTATGAACATG
> fasta
A DNASTringSet instance of length 4
  width seq          names
[1]   24 CGGACAG...TCCGGAT contig_1
[2]  103 GTCTGCC...CCCTGTC contig_2
[3]   65 TGTAGGA...TCGGGCA contig_3
[4]   49 CGTGCTG...GAACATG contig_4
> |
```

width

①fastaオブジェクト中の、②配列長情報は、③列名がwidthとなっていることから想像できますが、④width(fasta)とやればよいです。

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multi-FASTAファイル([hoge4.fa](#))の場合:

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

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

#本番(N50情報取得)
sorted <- rev(sort(width(fasta)))
obj <- (cumsum(sorted) >= Total_len)
N50 <- sorted[obj][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
> library(Biostrings) #パス$
>
> #入力ファイルの読み込み
> fasta <- readDNASTringSet(in_f, format="fasta")
> fasta
A DNASTringSet instance of length 4
  width seq          names
[1]   24 CGGACAG...TCCGGAT contig_1
[2]  103 GTCTGCC...CCCTGTC contig_2
[3]   65 TGTAGGA...TCGGGCA contig_3
[4]   49 CGTGCTG...GAACATG contig_4
> width(fasta)
[1] 24 103 65 49
> |
```


width

①fastaオブジェクト中の、②配列長情報は、③列名がwidthとなっていることから想像できますが、④width(fasta)とやればよいです。⑤結果は、こんな感じで見えていることからわかるように、数値ベクトル(整数のベクトル)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

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

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

#本番(N50情報取得)
sorted <- rev(sort(width(fasta)))
obj <- (cumsum(sorted) >= Total_len)
N50 <- sorted[obj][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
> library(Biostrings) #パス$
>
> #入力ファイルの読み込み
> fasta <- readDNASTringSet(in_f, format="fasta")
> fasta
A DNASTringSet instance of length 4
  width seq          names
[1]   24 CGGACAG...TCCGGAT contig_1
[2]  103 GTCTGCC...CCCTGTC contig_2
[3]   65 TGTAGGA...TCGGGCA contig_3
[4]   49 CGTGCTG...GAACATG contig_4
> width(fasta)
[1] 24 103 65 49
> |
```


sum

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

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

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

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

①fastaオブジェクト中の、②配列長情報は、③列名がwidthとなっていることから想像できますが、④width(fasta)とやればよいです。⑤結果は、こんな感じで見えていることからわかるように、数値ベクトル(整数のベクトル)です。⑤の数値ベクトルの総和が、トータルの配列長に相当します。⑥sum関数で総和を計算できます。上下左右の矢印キーを使って最小限の労力で打ち込もう!

```
RGui (64)
ファイル 編集
R Console
> library(Biostrings) #パス
>
> #入力ファイルの読み込み
> fasta <- readDNASTringSet(in_f, format="fasta")
> fasta
A DNASTringSet instance of length 4
  width seq          names
[1]   24 CGGACAG...TCCGGAT contig_1
[2]  103 GTCTGCC...CCCTGTC contig_2
[3]   65 TGTAGGA...TCGGGCA contig_3
[4]   49 CGTGCTG...GAACATG contig_4
> width(fasta)
[1] 24 103 65 49
> sum(width(fasta))
```

sum

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

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

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

#本番(N50情報取得)
sorted <- rev(sort(width(fasta)))
obj <- (cumsum(sorted) >= Total_len)
N50 <- sorted[obj][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="fasta")
> fasta
A DNASTringSet instance of length 4
  width seq          names
[1]   24 CGGACAG...TCCGGAT contig_1
[2]  103 GTCTGCC...CCCTGTC contig_2
[3]   65 TGTAGGA...TCGGGCA contig_3
[4]   49 CGTGCTG...GAACATG contig_4
> width(fasta)
[1] 24 103 65 49
> sum(width(fasta))
[1] 241
> |
```

①sum関数実行結果は、241塩基。妥当ですね。①と同じことを行っているのが、②です。

sum

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

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

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

#本番(N50情報取得)
sorted <- rev(sort(width(fasta)))
obj <- (cumsum(sorted) >= Total_len)[1]
N50 <- sorted[obj][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="fasta")
> fasta
A DNASTringSet instance of length 4
  width seq          names
[1]   24 CGGACAG...TCCGGAT contig_1
[2]  103 GTCTGCC...CCCTGTC contig_2
[3]   65 TGTAGGA...TCGGGCA contig_3
[4]   49 CGTGCTG...GAACATG contig_4
> width(fasta)
[1] 24 103 65 49
> sum(width(fasta))
[1] 241
> |
```

Contents

- 任意のキーワードを含む行を抽出するコードの解説の続き
 - おさらい、行列要素の抽出、`as.character`、`is.element`
 - 行列要素の抽出、`write.table`
- multi-FASTAファイルからの各種情報抽出
 - 基本情報取得(コンティグ数、配列長、N50、GC含量)
 - 課題1
 - コード内部の説明、`readDNAStringSet`、`width`、`sum`、`length`
 - サブセットの抽出(subsetting)、条件判定
 - 実践的な使い方(指定した長さ以上の配列を取得)
- Rパッケージの話
 - Bioconductor概観、Biostrings、推奨インストール手順、マニュアルなど
 - 課題2と課題3

ベクトルの要素数を調べるときに使うのは、①length関数。

length

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

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

#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, from

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

#本番(N50情報取得)
sorted <- rev(sort(width(fasta)))
obj <- (cumsum(sorted) >= Total_len)
N50 <- sorted[obj][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

> fasta
A DNASTringSet instance of length 4
  width seq          names
[1]   24 CGGACAG...TCCGGAT contig_1
[2]  103 GTCTGCC...CCCTGTC contig_2
[3]   65 TGTAGGA...TCGGGCA contig_3
[4]   49 CGTGCTG...GAACATG contig_4
> width(fasta)
[1] 24 103 65 49
> sum(width(fasta))
[1] 241
> length(fasta)
[1] 4
> |
```


ベクトルの要素数を調べる際に使うのは、①length関数。②fastaオブジェクトは、③DNAMStringSet形式ではあるが、入力として受け付けてくれます。

length

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

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

#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 <- readDNAMStringSet(in_f, from

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

#本番(N50情報取得)
sorted <- rev(sort(width(fasta)))
obj <- (cumsum(sorted) >= Total_len)
N50 <- sorted[obj][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
> fasta
A DNAMStringSet instance of length 4
width seq names
[1] 24 CGGACAG...TCCGGAT contig_1
[2] 103 GTCTGCC...CCCTGTC contig_2
[3] 65 TGTAGGA...TCGGGCA contig_3
[4] 49 CGTGCTG...GAACATG contig_4
> width(fasta)
[1] 24 103 65 49
> sum(width(fasta))
[1] 241
> length(fasta)
[1] 4
> |
```

length

ベクトルの要素数を調べるときに使うのは、①length関数。②fastaオブジェクトは、③DNASTringSet形式ではあるが、入力として受け付けてくれます。私は、④length 4 という表示から、なんとなく予想をつけて試したりします。

1. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して

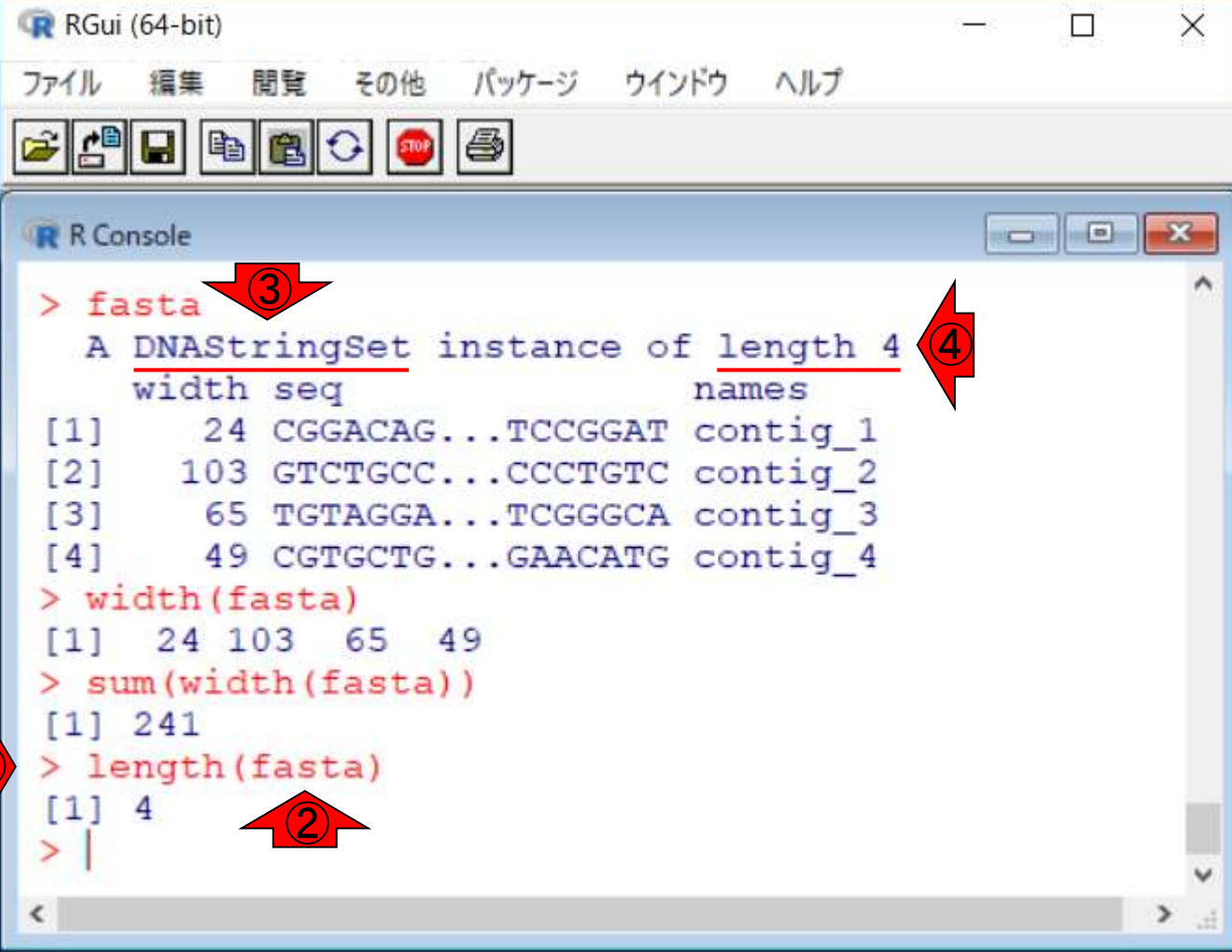
```
in_f <- "hoge4.fa"
out_f <- "hoge1.txt"

#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, from

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

#本番(N50情報取得)
sorted <- rev(sort(width(fasta)))
obj <- (cumsum(sorted) >= Total_len)
N50 <- sorted[obj][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
> fasta
A DNASTringSet instance of length 4
  width seq          names
[1]  24 CGGACAG...TCCGGAT contig_1
[2] 103 GTCTGCC...CCCTGTC contig_2
[3]  65 TGTAGGA...TCGGGCA contig_3
[4]  49 CGTGCTG...GAACATG contig_4
> width(fasta)
[1] 24 103 65 49
> sum(width(fasta))
[1] 241
> length(fasta)
[1] 4
> |
```

length

ベクトルの要素数を調べるときに使うのは、①length関数。②fastaオブジェクトは、③DNAMStringSet形式ではあるが、入力として受け付けてくれます。私は、④length 4 という表示から、なんとなく予想をつけて試したりします。①length(fasta)は、⑤と同じであり、配列数情報として保持しています。

1. [イントロ](#) | [一般](#) | [ランダムな塩基配列を作成](#)の4.を実行して

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

#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 <- readDNAMStringSet(in_f, format="fasta")

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

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

The screenshot shows the RGui interface with the R Console window open. The console displays the following output:

```
> fasta
A DNAMStringSet instance of length 4
  width seq          names
[1]   24 CGGACAG...TCCGGAT contig_1
[2]  103 GTCTGCC...CCCTGTC contig_2
[3]   65 TGTAGGA...TCGGGCA contig_3
[4]   49 CGTGCTG...GAACATG contig_4
> width(fasta)
[1] 24 103 65 49
> sum(width(fasta))
[1] 241
> length(fasta)
[1] 4
> |
```

Red arrows point to specific elements in the output: ③ points to the `fasta` command, ④ points to the `length 4` text, ⑤ points to the `length(fasta)` command, and ② points to the `length(fasta)` command's output.

①で行っているような、width(fasta)実行結果をsum関数の入力とするような書き方は、最初のうちは難解かもしれません。

分解して理解する

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

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

#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, from

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

#本番(N50情報取得)
sorted <- rev(sort(width(fasta)))
obj <- (cumsum(sorted) >= Total_len)
N50 <- sorted[obj][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
> fasta
A DNASTringSet instance of length 4
  width seq          names
[1]   24 CGGACAG...TCCGGAT contig_1
[2]  103 GTCTGCC...CCCTGTC contig_2
[3]   65 TGTAGGA...TCGGGCA contig_3
[4]   49 CGTGCTG...GAACATG contig_4
> width(fasta)
[1] 24 103 65 49
> sum(width(fasta))
[1] 241
> length(fasta)
[1] 4
> |
```


分解して理解する

①で行っているような、width(fasta)実行結果をsum関数の入力とするような書き方は、最初のうちは難解かもしれません。これは、例えば②のようにして、一旦width(fasta)の実行結果をggeという別のものに格納してから、length(gge)とするのと実質的に同じです。

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

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

#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, from

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

#本番(N50情報取得)
sorted <- rev(sort(width(fasta)))
obj <- (cumsum(sorted) >= Total_len)
N50 <- sorted[obj][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
[3] 65 TGTAGGA...TCGGGCA contig_3
[4] 49 CGTGCTG...GAACATG contig_4
> width(fasta)
[1] 24 103 65 49
> sum(width(fasta))
[1] 241
> length(fasta)
[1] 4
> gge <- width(fasta)
> gge
[1] 24 103 65 49
> length(gge)
[1] 4
> |
```

Contents

- 任意のキーワードを含む行を抽出するコードの解説の続き
 - おさらい、行列要素の抽出、`as.character`、`is.element`
 - 行列要素の抽出、`write.table`
- multi-FASTAファイルからの各種情報抽出
 - 基本情報取得(コンティグ数、配列長、N50、GC含量)
 - 課題1
 - コード内部の説明、`readDNAStringSet`、`width`、`sum`、`length`
 - サブセットの抽出(subsetting)、条件判定
 - 実践的な使い方(指定した長さ以上の配列を取得)
- Rパッケージの話
 - Bioconductor概観、Biostrings、推奨インストール手順、マニュアルなど
 - 課題2と課題3

①こんな感じで2番目の要素のみ抽出することができます。

サブセットの抽出

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

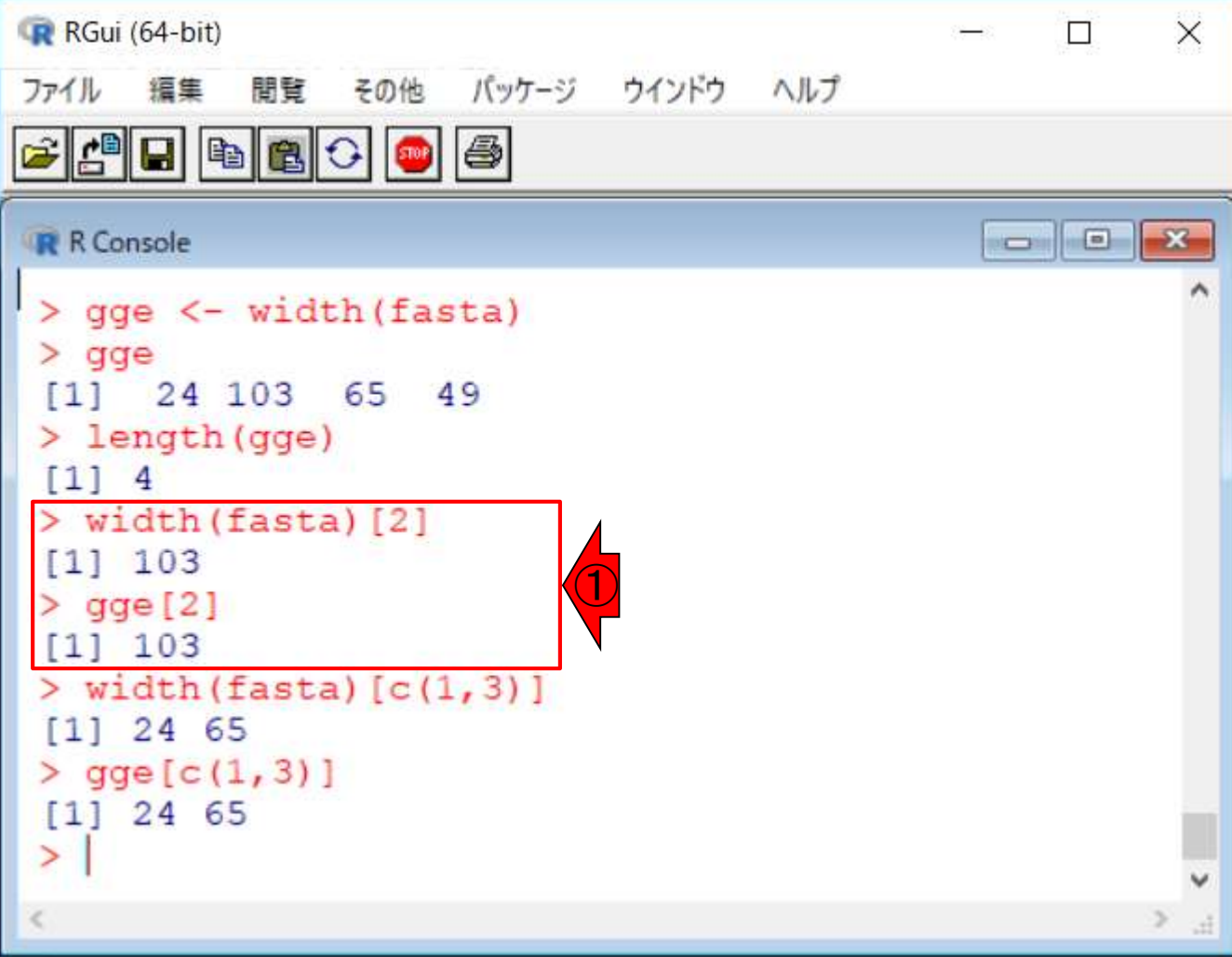
```
in_f <- "hoge4.fa"
out_f <- "hoge1.txt"

#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, from

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

#本番(N50情報取得)
sorted <- rev(sort(width(fasta)))
obj <- (cumsum(sorted) >= Total_len)
N50 <- sorted[obj][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
> gge <- width(fasta)
> gge
[1] 24 103 65 49
> length(gge)
[1] 4
> width(fasta)[2]
[1] 103
> gge[2]
[1] 103
> width(fasta)[c(1,3)]
[1] 24 65
> gge[c(1,3)]
[1] 24 65
> |
```

①こんな感じで2番目の要素のみ抽出することができます。②1と3番目の要素のみ抽出したい場合です。

サブセットの抽出

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

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

#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, from

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

#本番(N50情報取得)
sorted <- rev(sort(width(fasta)))
obj <- (cumsum(sorted) >= Total_len
N50 <- sorted[obj][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
> gge <- width(fasta)
> gge
[1] 24 103 65 49
> length(gge)
[1] 4
> width(fasta)[2]
[1] 103
> gge[2]
[1] 103
> width(fasta)[c(1,3)]
[1] 24 65
> gge[c(1,3)]
[1] 24 65
> |
```


①こんな感じで2番目の要素のみ抽出することができます。②1と3番目の要素のみ抽出したい場合です。③こんな具合で、様々な形で任意の数値ベクトルを指定することで、要素番号に相当する情報を抽出可能です。

サブセットの抽出

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

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

#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, from

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

#本番(N50情報取得)
sorted <- rev(sort(width(fasta)))
obj <- (cumsum(sorted) >= Total_len)
N50 <- sorted[obj][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
[1] 103
> width(fasta)[c(1, 3)]
[1] 24 65
> ggc[c(1, 3)]
[1] 24 65
> c(1, 3)
[1] 1 3
> c(1, 3, 4)
[1] 1 3 4
> 2:4
[1] 2 3 4
> c(1, 3:4)
[1] 1 3 4
> |
```

Contents

- 任意のキーワードを含む行を抽出するコードの解説の続き
 - おさらい、行列要素の抽出、`as.character`、`is.element`
 - 行列要素の抽出、`write.table`
- multi-FASTAファイルからの各種情報抽出
 - 基本情報取得(コンティグ数、配列長、N50、GC含量)
 - 課題1
 - コード内部の説明、`readDNAStringSet`、`width`、`sum`、`length`
 - サブセットの抽出(subsetting)、条件判定
 - 実践的な使い方(指定した長さ以上の配列を取得)
- Rパッケージの話
 - Bioconductor概観、Biostrings、推奨インストール手順、マニュアルなど
 - 課題2と課題3

条件判定

①width(fasta)の数値ベクトルの中から、103という数値と一致する要素をTRUE、それ以外をFALSEとして返しています。TRUE or FALSEからなるベクトルを論理値ベクトル(logical vector)と言います。ヒトによって記述の仕方が異なります。例えば、②as.logical関数も示していますが、特に深い意味はないということが分かります。

1. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して

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

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

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

```
RGui (64-bit)
ファイル 編集 開発 表示 拡張機能

R Console
> fasta
A DNASTringSet instance of length 4
  width seq          names
[1]   24 CGGACAG...TCCGGAT contig_1
[2]  103 GTCTGCC...CCCTGTC contig_2
[3]   65 TGTAGGA...TCGGGCA contig_3
[4]   49 CGTGCTG...GAACATG contig_4
> width(fasta) == 103
[1] FALSE  TRUE FALSE FALSE
> (width(fasta) == 103)
[1] FALSE  TRUE FALSE FALSE
> as.logical(width(fasta) == 103)
[1] FALSE  TRUE FALSE FALSE
> |
```

条件判定

「(Rで)塩基配列解析」上では、①条件判定結果をobjという名前で利用する 경우가ほとんどです。この論理値ベクトルobjは、②のような感じで、sum関数と併用して、TRUEの要素数(この場合は103塩基となる配列数)を調べたり…

1. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して

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

#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, from

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

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

```
RGui (64
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
> (width(fasta) == 103)
[1] FALSE TRUE FALSE FALSE
> as.logical(width(fasta) == 103)
[1] FALSE TRUE FALSE FALSE
> obj <- as.logical(width(fasta) == 103)
> obj
[1] FALSE TRUE FALSE FALSE
> sum(obj)
[1] 1
> fasta[obj]
A DNASTringSet instance of length 1
width seq names
[1] 103 GTCTGCC...CCCTGTC contig_2
> |
```


条件判定

「(Rで)塩基配列解析」上では、①条件判定結果をobjという名前で利用する 경우가ほとんどです。この論理値ベクトルobjは、②のような感じで、sum関数と併用して、TRUEの要素数(この場合は103塩基となる配列数)を調べたり、③fastaオブジェクトから、要素番号がTRUEのもののみ取り出したりする目的で使います。

1. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して

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

#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, from

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

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

```
RGui (64
ファイル 編

R Console

> (width(fasta) == 103)
[1] FALSE TRUE FALSE FALSE
> as.logical(width(fasta) == 103)
[1] FALSE TRUE FALSE FALSE
> obj <- as.logical(width(fasta) == 103)
> obj
[1] FALSE TRUE FALSE FALSE
> sum(obj)
[1] 1
> fasta[obj]
A DNASTringSet instance of length 1
width seq names
[1] 103 GTCTGCC...CCCTGTC contig_2
> |
```

条件判定

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

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

#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, from

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

#本番(N50情報取得)
sorted <- rev(sort(width(fasta)))
obj <- (cumsum(sorted) >= Total_len)
N50 <- sorted[obj][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

width seq names
[1] 103 GTCTGCC...CCCTGTC contig_2
> fasta[c(1,3)]
A DNASTringSet instance of length 2
width seq names
[1] 24 CGGACAG...TCCGGAT contig_1
[2] 65 TGTAGGA...TCGGGCA contig_3
> fasta[2:4]
A DNASTringSet instance of length 3
width seq names
[1] 103 GTCTGCC...CCCTGTC contig_2
[2] 65 TGTAGGA...TCGGGCA contig_3
[3] 49 CGTGCTG...GAACATG contig_4
> |
```

Contents

- 任意のキーワードを含む行を抽出するコードの解説の続き
 - おさらい、行列要素の抽出、`as.character`、`is.element`
 - 行列要素の抽出、`write.table`
- multi-FASTAファイルからの各種情報抽出
 - 基本情報取得(コンティグ数、配列長、N50、GC含量)
 - 課題1
 - コード内部の説明、`readDNAStringSet`、`width`、`sum`、`length`
 - サブセットの抽出(subsetting)、条件判定
 - 実践的な使い方(指定した長さ以上の配列を取得)
- Rパッケージの話
 - Bioconductor概観、Biostrings、推奨インストール手順、マニュアルなど
 - 課題2と課題3

実践的な使い方

①実践的な使い方の一例。これは、ゲノムアセンブリによって、数多くのコンティグが得られ、200塩基以上など一定の長さ以上の配列群のみにしたい場合に使えます。
②は50塩基以上の例ですが、50のところを200に変更すればよいだけです。このような目的に使える項目は…

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

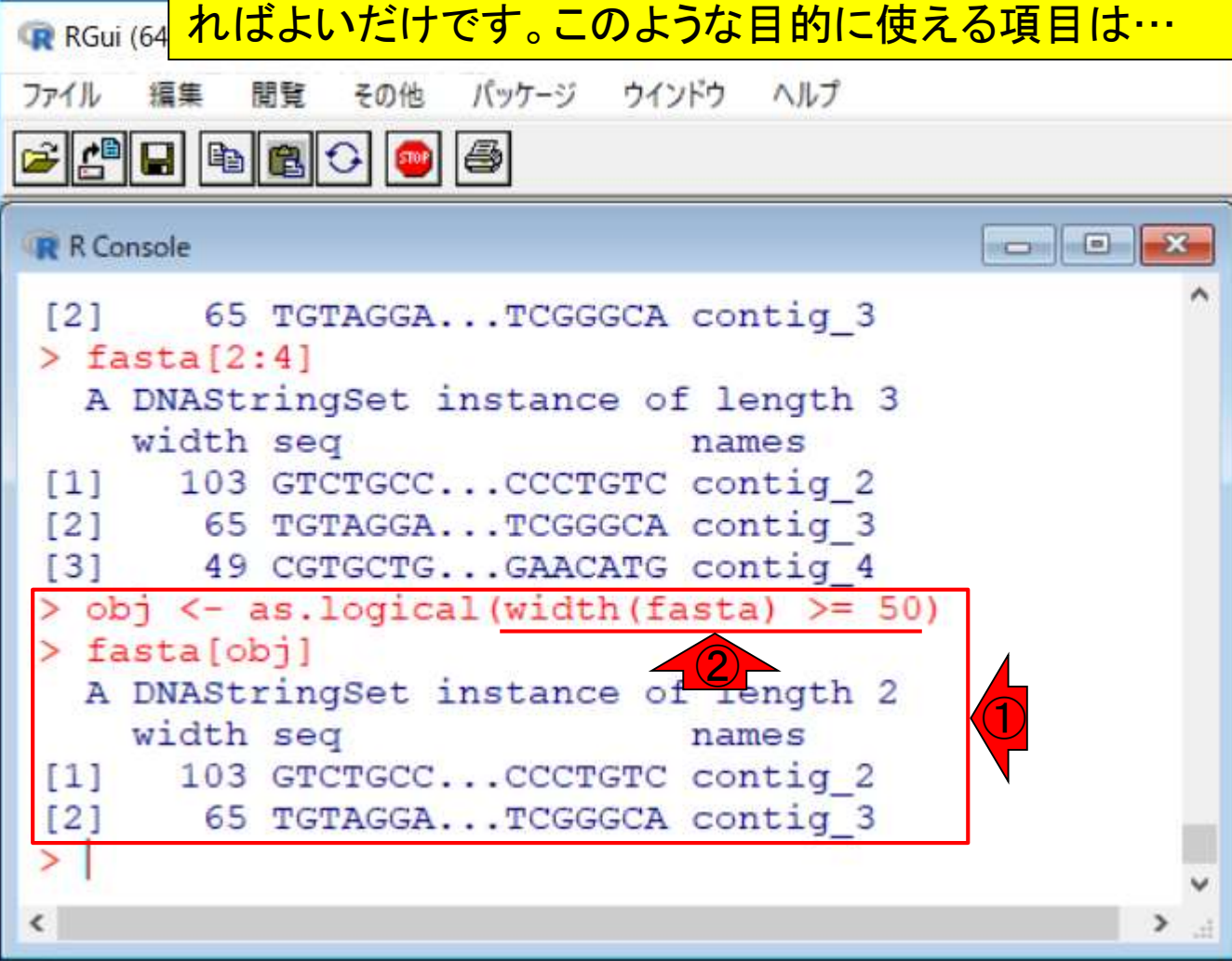
```
in_f <- "hoge4.fa"
out_f <- "hoge1.txt"

#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, from

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

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



```
RGui (64)
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

[2] 65 TGTAGGA...TCGGGCA contig_3
> fasta[2:4]
A DNASTringSet instance of length 3
width seq names
[1] 103 GTCTGCC...CCCTGTC contig_2
[2] 65 TGTAGGA...TCGGGCA contig_3
[3] 49 CGTGCTG...GAACATG contig_4
> obj <- as.logical(width(fasta) >= 50)
> fasta[obj]
A DNASTringSet instance of length 2
width seq names
[1] 103 GTCTGCC...CCCTGTC contig_2
[2] 65 TGTAGGA...TCGGGCA contig_3
> |
```


実践的な使い方

①「(Rで)塩基配列解析」の、②です。項目名の通り、指定した長さ以上の配列を抽出したい場合に、テンプレートとして利用できます。

(Rで)塩基配列解析

保護されていない通信 | www.iu.a.u-tokyo.ac.jp/~kadota/...

(Rで)塩基配列解析

①

- 前処理 | フィルタリング | [ACGT以外のcharacter"-"をNに変換](#) (last modified 2013/06/18)
- 前処理 | フィルタリング | [ACGT以外の文字数が閾値以下の配列を抽出](#) (last modified 2015/06/18)
- 前処理 | フィルタリング | [重複のない配列セットを作成](#) (last modified 2013/06/18)
- 前処理 | フィルタリング | [指定した長さ以上の配列を抽出](#) (last modified 2016/02/08)
- 前処理 | フィルタリング | [任意のリード\(サブセット\)を抽出](#) (last modified 2014/08/21)
- 前処理 | フィルタリング | [指定した長さの範囲の配列を抽出](#) (last modified 2015/02/06)
- 前処理 | フィルタリング | [指定した長さ以上の配列を抽出](#) (last modified 2015/02/06)
- 前処理 | フィルタリング | [指定した長さ以上の配列を抽出](#) (last modified 2015/02/06)
- 前処理 | フィルタリング | [指定した長さ以上の配列を抽出](#) (last modified 2015/02/06)
- 前処理 | フィルタリング | [指定した長さ以上の配列を抽出](#) (last modified 2015/02/06)
- 前処理 | フィルタリング | [指定した長さ以上の配列を抽出](#) (last modified 2015/02/06)
- 前処理 | フィルタリング | [指定した長さ以上の配列を抽出](#) (last modified 2015/02/06)

②

前処理 | フィルタリング | 指定した長さ以上の配列を抽出

FASTA形式やFASTQ形式ファイルを入力として、指定した配列長以上の配列を抽出するやり方を示します。
「ファイル」 - 「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピー。

1. multi-FASTAファイル(hoge4.fa)の場合:

[イントロ](#) | [一般](#) | [ランダムな塩基配列を作成](#)の4.を実行して得られたファイルです。

```
in_f <- "hoge4.fa" #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.fasta" #出力ファイル名を指定してout_fに格納
param_length <- 50 #配列長の閾値を指定

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

[トップページへ](#)

実践的な使い方

前処理 | フィルタリング | 指定した長さ以上の配列を抽出 ①

FASTA形式やFASTQ形式ファイルを入力として、指定した配列長以上の配列を抽出するやり方を示します。
「ファイル」 - 「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピー。

1. multi-FASTAファイル([hoge4.fa](#))の場合: 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ファイルです。

```

in_f <- "hoge4.fa"           #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.fasta"      #出力ファイル名を指定してout_fに格納
param_length <- 50         #配列長の閾値を指定

#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
fasta                                #確認してるだけです

#本番
obj <- as.logical(width(fasta) >= param_length)#条件を満たすかどうかを判定した結果をobjに格納
fasta <- fasta[obj]         #objがTRUEとなる要素のみ抽出した結果をfastaに格納
fasta                                #確認してるだけです

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

```

[トップページへ](#)

実践的な使い方

①の項目の、②例題1を表示。③入出力ファイルともに、FASTA形式ファイル。デフォルトは50塩基以上のものを抽出するようにしている。

(Rで)塩基配列解析

保護されていない通信 | www.iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#preproce...

前処理 | フィルタリング | 指定した長さ以上の配列を抽出 ①

FASTA形式やFASTQ形式ファイルを入力として、指定した配列長以上の配列を抽出するやり方を示します。
「ファイル」 - 「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピー。

1. multi-FASTAファイル(hoge4.fa)の場合: 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ファイルです。

```
in_f <- "hoge4.fa"           #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.fasta"       #出力ファイル名を指定してout_fに格納
param_length <- 50          #配列長の閾値を指定

#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
fasta                                     #確認してるだけです

#本番
obj <- as.logical(width(fasta) >= param_length)#条件を満たすかどうかを判定した結果をobjに格納
fasta <- fasta[obj]          #objがTRUEとなる要素のみ抽出した結果をfastaに格納
fasta                         #確認してるだけです

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

[トップページへ](#)

実践的な使い方

①の項目の、②例題1を表示。③入出力ファイルともに、FASTA形式ファイル。デフォルトは50塩基以上のものを抽出するようにしている。④が入力ファイル(hoge4.fa)を読み込んでfastaオブジェクトに格納までのところ。

(Rで)塩基配列解析

保護されていない通信 | www.iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#preproce...

前処理 | フィルタリング | 指定した長さ以上の配列を抽出 ①

FASTA形式やFASTQ形式ファイルを入力として、指定した配列長以上の配列を抽出するやり方を示します。
「ファイル」 - 「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピー。

1. multi-FASTAファイル(hoge4.fa)の場合: 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ファイルです。

```
in_f <- "hoge4.fa"           #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.fasta"       #出力ファイル名を指定してout_fに格納
param_length <- 50         #配列長の閾値を指定
```

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

```
#本番
obj <- as.logical(width(fasta) >= param_length)#条件を満たすかどうかを判定した結果をobjに格納
fasta <- fasta[obj]         #objがTRUEとなる要素のみ抽出した結果をfastaに格納
fasta                       #確認してるだけです
```

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

[トップページへ](#)

実践的な使い方

①の項目の、②例題1を表示。③入出力ファイルともに、FASTA形式ファイル。デフォルトは50塩基以上のものを抽出するようにしている。④が入力ファイル(hoge4.fa)を読み込んでfastaオブジェクトに格納までのところ。⑤が条件判定結果として得られた論理値ベクトルobjを用いてサブセットの抽出を行っているところ。抽出結果を同じfastaオブジェクトに格納しているので、この段階で元のfastaオブジェクトはなくなっています。

(Rで)塩基配列解析

保護されていない通信 | www.iu.a.u-tokyo.a

前処理 | フィルタリング | 指定した長さ

FASTA形式やFASTQ形式ファイルを入力として、指定した配列「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. multi-FASTAファイル(hoge4.fa)の場合:

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ファイルです。

```
in_f <- "hoge4.fa"
out_f <- "hoge1.fasta"
param_length <- 50
```

#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
#配列長の閾値を指定

③

```
#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

④

```
#本番
obj <- as.logical(width(fasta) >= param_length)#条件を満たすかどうかを判定した結果をobjに格納
fasta <- fasta[obj]
fasta
```

#objがTRUEとなる要素のみ抽出した結果をfastaに格納
#確認してるだけです

⑤

```
#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](#)

実践的な使い方

①の項目の、②例題1を表示。③入出力ファイルともに、FASTA形式ファイル。デフォルトは50塩基以上のものを抽出するようにしている。④が入力ファイル(hoge4.fa)を読み込んでfastaオブジェクトに格納までのところ。⑤が条件判定結果として得られた論理値ベクトルobjを用いてサブセットの抽出を行っているところ。抽出結果を同じfastaオブジェクトに格納しているので、この段階で元のfastaオブジェクトはなくなっています。⑥がwriteXStringSetという関数を利用して、DNAStrngSet形式のfastaオブジェクトの中身を、FASTA形式ファイル(hoge1.fasta)で出力するコード。

(Rで)塩基配列解析

保護されていない通信 | www.iu.a.u-tokyo.a

前処理 | フィルタリング | 指定した長さ

FASTA形式やFASTQ形式ファイルを入力として、指定した配列「ファイル」 - 「ディレクトリの変更」で解析したいファイル

1. multi-FASTAファイル(hoge4.fa)の場合:

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

```
in_f <- "hoge4.fa"
out_f <- "hoge1.fasta"
param_length <- 50
```

#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
#配列長の閾値を指定

③

```
#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 <- readDNAStrngSet(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

④

```
#本番
obj <- as.logical(width(fasta) >= param_length)#条件を満たすかどうかを判定した結果をobjに格納
fasta <- fasta[obj]
fasta
```

#objがTRUEとなる要素のみ抽出した結果をfastaに格納
#確認してるだけです

⑤

```
#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
```

⑥

トップページへ

必要に応じて変更

全般的に、このようなノリで記述しています。赤字部分で説明もしてありますので、例えば①の部分で「指定した長さ未満」にするなど随時変更して、できることの幅も広げていきましょう。

(Rで)塩基配列解析

保護されていない通信 | www.iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#preproce...

前処理 | フィルタリング | 指定した長さ以上の配列を抽出

FASTA形式やFASTQ形式ファイルを入力として、指定した配列長以上の配列を抽出するやり方を示します。
「ファイル」 - 「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピー。

1. multi-FASTAファイル([hoge4.fa](#))の場合:

[イントロ](#) | [一般](#) | [ランダムな塩基配列を作成](#)の4.を実行して得られたファイルです。

```
in_f <- "hoge4.fa"           #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.fasta"       #出力ファイル名を指定してout_fに格納
param_length <- 50          #配列長の閾値を指定

#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
fasta                                     #確認してるだけです

#本番
obj <- as.logical(width(fasta) >= param_length)#条件を満たすかどうかを判定した結果をobjに格納
fasta <- fasta[obj]          #objがTRUEとなる要素のみ抽出した結果をfastaに格納
fasta                           #確認してるだけです

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

[トップページへ](#)

複数の例題

NGS解析の場合は、FASTAファイル以外にFASTQファイルを取り扱うことも多いです。FASTQファイルで同様なことを行う例題もあります。

(Rで)塩基配列解析

保護されていない通信 | www.iu.a.u-tokyo.ac.jp/~kadota/r_seq.html#preproce...

前処理 | フィルタリング | 指定した長さ以上の配列を抽出

FASTA形式やFASTQ形式ファイルを入力として、指定した配列長以上の配列を抽出するやり方を示します。
「ファイル」 - 「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピー。

1. multi-FASTAファイル([hoge4.fa](#))の場合:

[イントロ](#) | [一般](#) | [ランダムな塩基配列を作成](#)の4.を実行して得られたファイルです。

```
in_f <- "hoge4.fa"           #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.fasta"       #出力ファイル名を指定してout_fに格納
param_length <- 50          #配列長の閾値を指定

#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
fasta                                     #確認してるだけです

#本番
obj <- as.logical(width(fasta) >= param_length)#条件を満たすかどうかを判定した結果をobjに格納
fasta <- fasta[obj]         #objがTRUEとなる要素のみ抽出した結果をfastaに格納
fasta                         #確認してるだけです

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

[トップページへ](#)

複数の例題

少しページ下部に移動したところ。①例題3は、FASTQファイルを読み込んでFASTQファイルで出力するコード。②例題4は、FASTQファイルを読み込んでFASTAファイルで出力するコード。様々なバリエーションがありますので、コードの中身に興味があるヒトは、一致している部分と違いのある部分を見比べるとよいでしょう。

(Rで)塩基配列解析

① 保護されていない通信 | www.iu.a.u-tokyo.a

3. FASTQ形式ファイル(sample2.fastq)を読み込んでFASTQ形式で保存する場合:

writeFastq関数のデフォルトオプションはcompress=Tで、gzip圧縮ファイルを出力します。ここではcompress=Fとして非圧縮ファイルを出力しています。

```
in_f <- "sample2.fastq" #入力ファイル名を指定してin_fに格納
out_f <- "hoge3.fastq" #出力ファイル名を指定してout_fに格納
param_length <- 26 #配列長の閾値を指定

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

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

#本番
obj <- as.logical(width(fastq) >= param_length) #条件を満たすかどうかを判定した結果をobjに格納
fastq <- fastq[obj] #objがTRUEとなる要素のみ抽出した結果をfastqに格納
sread(fastq) #配列情報を表示
table(width(fastq)) #配列長分布を表示

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

4. FASTQ形式ファイル(sample2.fastq)を読み込んでFASTA形式で保存する場合:

[トップページへ](#)

Contents

- 任意のキーワードを含む行を抽出するコードの解説の続き
 - おさらい、行列要素の抽出、`as.character`、`is.element`
 - 行列要素の抽出、`write.table`
- multi-FASTAファイルからの各種情報抽出
 - 基本情報取得(コンティグ数、配列長、N50、GC含量)
 - 課題1
 - コード内部の説明、`readDNAStringSet`、`width`、`sum`、`length`
 - サブセットの抽出(subsetting)、条件判定
 - 実践的な使い方(指定した長さ以上の配列を取得)
- Rパッケージの話
 - Bioconductor概観、Biostrings、推奨インストール手順、マニュアルなど
 - 課題2と課題3

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。例えば、①library(Biostrings)という呪文を唱えて読み込むことで、Biostringsパッケージが提供する②readDNASTringSet関数などを利用できるのです。

前処理 | フィルタリング | 指定した長さ

FASTA形式やFASTQ形式ファイルを入力として、指定した配列長以上の配列を抽出するやり方を示します。「ファイル」 - 「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピー。

1. multi-FASTAファイル(hoge4.fa)の場合:

[イントロ](#) | [一般](#) | [ランダムな塩基配列を作成](#)の4.を実行して得られたファイルです。

```
in_f <- "hoge4.fa"           #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.fasta"       #出力ファイル名を指定してout_fに格納
param_length <- 50          #配列長の閾値を指定

#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
fasta                                     #確認してるだけです

#本番
obj <- as.logical(width(fasta) >= param_length) #条件を満たすかどうかを判定した結果をobjに格納
fasta <- fasta[obj]          #objがTRUEとなる要素のみ抽出した結果をfastaに格納
fasta                         #確認してるだけです

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

[トップページへ](#)

2大リポジトリ

Rのパッケージ提供サイト(貯蔵庫という意味で「リポジトリ」とも言われます)として有名なのは、①Bioconductorと②CRANです。まずは①Bioconductorから。

講義日程 (2019年度)

1. 2019年04月08日 (PC使用)
講義資料PDF(最終更新: 2019.04.09)
学会(国外): ISCB
学会(国内): JSBi
QAサイト: Biostar (Parnell et al., PLoS Comput Biol., 2011)
QAサイト: SEQanswers (Li et al., Bioinformatics, 2012)
学習教材: バイオインフォマティクス人材育成のための講習会(平成26-29年度)
学習教材: (Rで)塩基配列解析
学習教材: (Rで)塩基配列解析のサブ
RStudio
2. 2019年04月15日 (PC使用)
講義資料PDF
(Rで)塩基配列解析
(Rで)塩基配列解析のサブ
3. 2019年04月22日 (PC使用)
講義資料PDF
(Rで)塩基配列解析
hoge8.fa (課題用)
Bioconductor ①
CRAN ②
4. 2019年05月13日 (PC使用)
講義資料PDF

Bioconductor

①Bioconductorの最新リリース(ver. 3.8)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。③を読むと、4月30日にver. 3.9をリリース予定だそうです。②の部分がnew releaseに切り替え後に増えると思います。が、最初のうちは意味不明だと思うので気にしなくてもよい。

The screenshot shows the Bioconductor website homepage. The browser address bar displays 'Bioconductor - Home' and 'bioconductor.org'. The main navigation bar includes 'Home', 'Install', 'Help', 'Developers', and 'About'. A red arrow labeled '1' points to the 'Home' link. Below the navigation bar, there are four main content sections: 'About Bioconductor', 'Install', 'Learn', and 'Develop'. A red arrow labeled '2' points to the 'Install' section, which contains a list of links including 'Discover 1649 software packages available in Bioconductor release 3.8.' and 'Get started with Bioconductor' with sub-links like 'Install Bioconductor', 'Get support', 'Latest newsletter', 'Follow us on twitter', and 'Install R'. The 'Learn' section lists 'Master Bioconductor tools' with links for 'Courses', 'Support site', 'Package vignettes', 'Literature citations', 'Common work flows', 'FAQ', 'Community resources', and 'Videos'. The 'Develop' section lists 'Contribute to Bioconductor' with links for 'Developer resources', 'Use Bioc 'devel'', 'Devel' packages, 'Package guidelines', 'New package submission', 'Git source control', and 'Build reports'. The 'About Bioconductor' section provides a brief overview of the project and its use of R. The 'News' section lists recent updates, including the '3.9 release schedule' and 'Bioconductor 3.8' availability.

こんな感じになります。

Bioconductor

Bioconductor - BiocViews

bioconductor.org/packages/release/BiocViews.html#__Software

Search:

Home Install Help Developers About

Home » BiocViews

All Packages

Bioconductor version 3.8 (Release)

Autocomplete biocViews search:

Packages found under Software:

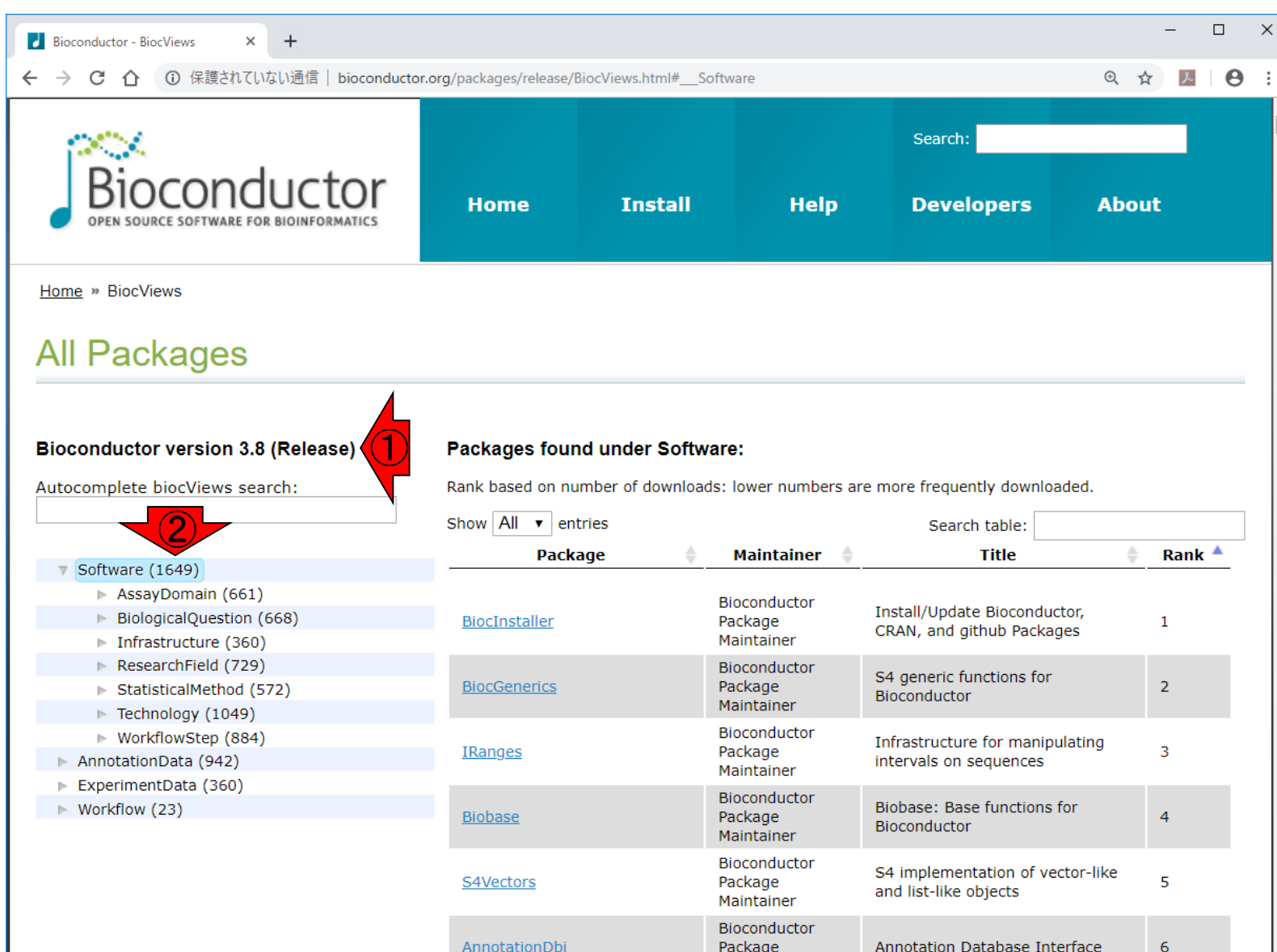
Rank based on number of downloads: lower numbers are more frequently downloaded.

Show 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
AnnotationDbi	Bioconductor Package	Annotation Database Interface	6

Bioconductor

こんな感じになります。これは、①Bioconductor ver. 3.8
で提供されている、②全1649パッケージの結果です。



Bioconductor - BioViews

bioconductor.org/packages/release/BiocViews.html#___Software

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Home Install Help Developers About

Home » BioViews

All Packages

Bioconductor version 3.8 (Release) ①

Autocomplete biocViews search: ②

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
AnnotationDbi	Bioconductor Package	Annotation Database Interface	6

Software (1649)

- AssayDomain (661)
- BiologicalQuestion (668)
- Infrastructure (360)
- ResearchField (729)
- StatisticalMethod (572)
- Technology (1049)
- WorkflowStep (884)
- AnnotationData (942)
- ExperimentData (360)
- Workflow (23)

Bioconductor

こんな感じになります。これは、①Bioconductor ver. 3.8
で提供されている、②全1649パッケージの結果です。③
カテゴリー別がこちら。

Bioconductor - BioViews

bioconductor.org/packages/release/BiocViews.html#_Software

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Home Install Help Developers About

Home » BioViews

All Packages

Bioconductor version 3.8 (Release) ①

Autocomplete biocViews search: ②

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
AnnotationDbi	Bioconductor Package	Annotation Database Interface	6

Software (1649) ③

- AssayDomain (661)
- BiologicalQuestion (668)
- Infrastructure (360)
- ResearchField (729)
- StatisticalMethod (572)
- Technology (1049)
- WorkflowStep (884)
- AnnotationData (942)
- ExperimentData (360)
- Workflow (23)

Bioconductor

こんな感じになります。これは、①Bioconductor ver. 3.8で提供されている、②全1649パッケージの結果です。③カテゴリー別がこちら。④ダウンロード数のランキング順にリストされているのが、⑤こちら。

Bioconductor - BioViews

Home » BioViews

All Packages

Bioconductor version 3.8 (Release)

Autocomplete biocViews search:

Software (1649)

- AssayDomain (661)
- BiologicalQuestion (668)
- Infrastructure (360)
- ResearchField (729)
- StatisticalMethod (572)
- Technology (1049)
- WorkflowStep (884)
- AnnotationData (942)
- 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
AnnotationDbi	Bioconductor Package	Annotation Database Interface	6

Bioconductor

こんな感じになります。これは、①Bioconductor ver. 3.8で提供されている、②全1649パッケージの結果です。③カテゴリー別がこちら。④ダウンロード数のランキング順にリストされているのが、⑤こちら。⑥1位はBiocInstallerというインストール用のRパッケージ。当たり前といえば当たり前ですね。

Bioconductor version 3.8 (Release) ①

Autocomplete biocViews search: ②

Software (1649) ③

- AssayDomain (661)
- BiologicalQuestion (668)
- Infrastructure (360)
- ResearchField (729)
- StatisticalMethod (572)
- Technology (1049)
- WorkflowStep (884)

AnnotationData (942)

ExperimentData (360)

Workflow (23)

Packages found under Software: ④

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

Show All entries Search table:

Package	Maintainer	Title	Rank
BiocInstaller ⑥	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
AnnotationDbi	Bioconductor Package	Annotation Database Interface	6

Bioconductor

①デフォルトはここがAllになっているので、②Search table上で、キーワード検索をして有用そうなパッケージの検索を行ってもよいと思います。

Home » BioViews

All Packages

Bioconductor version 3.8 (Release)

Autocomplete biocViews search:

Packages found under Software:

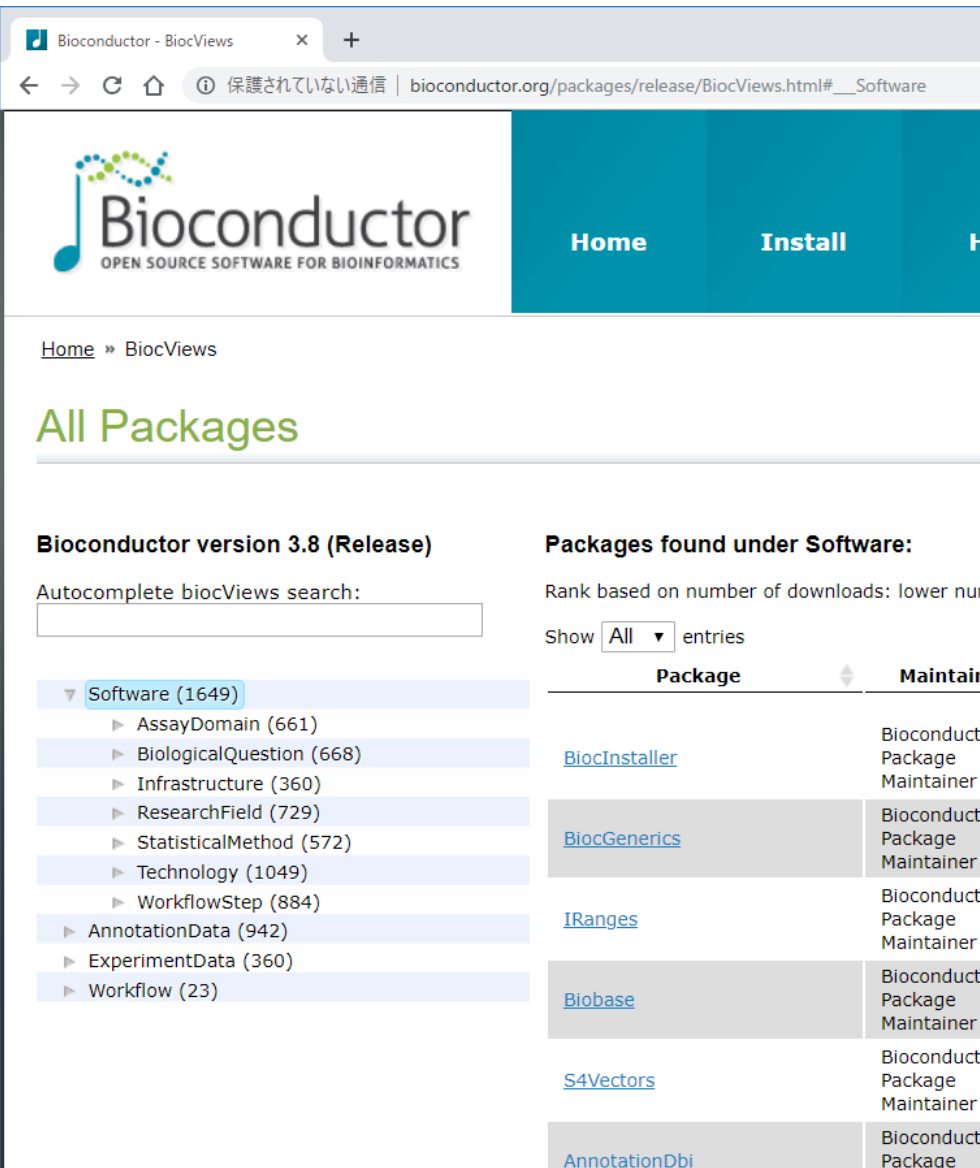
Rank: **1** number of downloads: lower numbers are more frequently downloaded.

Show **All** entries Search table: **2**

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
AnnotationDbi	Bioconductor Package	Annotation Database Interface	6

Bioconductor

①下矢印キーを少しづつ押していったって、②Rankが13位のパッケージを見てください。



Bioconductor - BiocViews

Home » BiocViews

All Packages

Bioconductor version 3.8 (Release)

Autocomplete biocViews search:

▼ Software (1649)

- ▶ AssayDomain (661)
- ▶ BiologicalQuestion (668)
- ▶ Infrastructure (360)
- ▶ ResearchField (729)
- ▶ StatisticalMethod (572)
- ▶ Technology (1049)
- ▶ WorkflowStep (884)
- ▶ AnnotationData (942)
- ▶ ExperimentData (360)
- ▶ Workflow (23)



Packages found under Software:

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

Show 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
AnnotationDbi	Bioconductor Package	Annotation Database Interface	6



Bioconductor

①下矢印キーを少しずつ押していき、②Rankが13位のパッケージを見てください。翻訳配列取得でもお世話になった③Biostringsは、1649パッケージ中13位なので、よく利用されているものです。③Biostringsは、他のパッケージ中で内部的に利用されたりもしています。

S4Vectors	Bioconductor Package Maintainer	S4 implementation of vector-like and list-like objects	5
AnnotationDbi	Bioconductor Package Maintainer	Annotation Database Interface	6
zlibbioc	Bioconductor Package Maintainer	An R packaged zlib-1.2.5	7
BiocParallel	Bioconductor Package Maintainer	Bioconductor facilities for parallel evaluation	8
XVector	Hervé Pagès	Representation and manipulation of external sequences	9
GenomicRanges	Bioconductor Package Maintainer	Representation and manipulation of genomic intervals and variables defined along a genome	10
limma	Gordon Smyth	Linear Models for Microarray Data	11
GenomeInfoDb	Bioconductor Package Maintainer	Utilities for manipulating chromosome and other 'seqname' identifiers	12
Biostrings	H. Pagès	Efficient manipulation of biological strings	13
DelayedArray	Hervé Pagès	Delayed operations on array-like objects	14
SummarizedExperiment	Bioconductor Package Maintainer	SummarizedExperiment container	15
annotate	Bioconductor Package Maintainer	Annotation for microarrays	16
Rsamtools	Bioconductor Package Maintainer	Binary alignment (BAM), FASTA, variant call (BCF), and tabix file import	17
genefilter	Bioconductor Package Maintainer	genefilter: methods for filtering genes from high-throughput experiments	18
	Bioconductor	Representation and manipulation	

Bioconductor

①下矢印キーを少しずつ押していったら、②Rankが13位のパッケージを見てください。翻訳配列取得でもお世話になった③Biostringsは、1649パッケージ中13位なので、よく利用されているものです。③Biostringsは、他のパッケージ中で内部的に利用されたりもしています。④のTitle列の情報が、Biostringsパッケージが何をするものかについて簡潔に示しています。

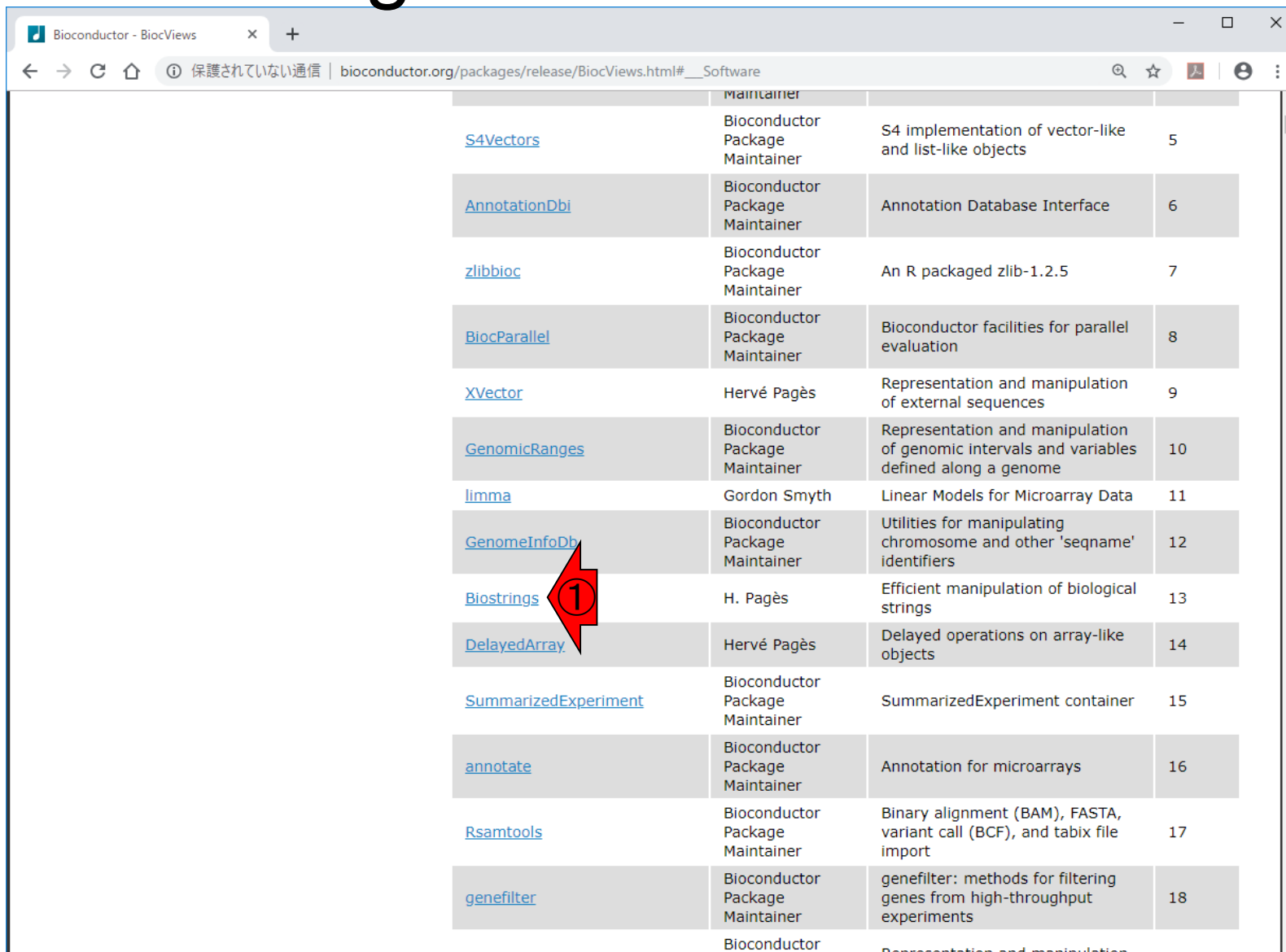
Package	Maintainer	Description	Rank
S4Vectors			6
AnnotationDbi	Bioconductor Package Maintainer	Annotation Database Interface	7
zlibbioc	Bioconductor Package Maintainer	An R packaged zlib-1.2.5	8
BiocParallel	Bioconductor Package Maintainer	Bioconductor facilities for parallel evaluation	9
XVector	Hervé Pagès	Representation and manipulation of external sequences	10
GenomicRanges	Bioconductor Package Maintainer	Representation and manipulation of genomic intervals and variables defined along a genome	11
limma	Gordon Smyth	Linear Models for Microarray Data	12
GenomeInfoDb	Bioconductor Package Maintainer	Utilities for manipulating chromosome, 'seqname' identifiers	13
Biostrings	H. Pagès	Efficient manipulation of biological strings	14
DelayedArray	Hervé Pagès	Delayed operations on array-like objects	15
SummarizedExperiment	Bioconductor Package Maintainer	SummarizedExperiment container	16
annotate	Bioconductor Package Maintainer	Annotation for microarrays	17
Rsamtools	Bioconductor Package Maintainer	Binary alignment (BAM), FASTA, variant call (BCF), and tabix file import	18
genefilter	Bioconductor Package Maintainer	genefilter: methods for filtering genes from high-throughput experiments	

Contents

- 任意のキーワードを含む行を抽出するコードの解説の続き
 - おさらい、行列要素の抽出、`as.character`、`is.element`
 - 行列要素の抽出、`write.table`
- multi-FASTAファイルからの各種情報抽出
 - 基本情報取得(コンティグ数、配列長、N50、GC含量)
 - 課題1
 - コード内部の説明、`readDNAStringSet`、`width`、`sum`、`length`
 - サブセットの抽出(subsetting)、条件判定
 - 実践的な使い方(指定した長さ以上の配列を取得)
- Rパッケージの話
 - Bioconductor概観、`Biostings`、推奨インストール手順、マニュアルなど
 - 課題2と課題3

①Biostringsのところをクリックしてみましょう。

Biostrings



The screenshot shows a web browser window displaying a list of Bioconductor packages. The browser's address bar shows the URL: `bioconductor.org/packages/release/BiocViews.html#___Software`. The page content is a table with columns for package names, maintainers, descriptions, and version numbers. The 'Biostrings' package is highlighted with a red arrow and a circled '1'.

Package Name	Maintainer	Description	Version
S4Vectors	Bioconductor Package Maintainer	S4 implementation of vector-like and list-like objects	5
AnnotationDbi	Bioconductor Package Maintainer	Annotation Database Interface	6
zlibbioc	Bioconductor Package Maintainer	An R packaged zlib-1.2.5	7
BiocParallel	Bioconductor Package Maintainer	Bioconductor facilities for parallel evaluation	8
XVector	Hervé Pagès	Representation and manipulation of external sequences	9
GenomicRanges	Bioconductor Package Maintainer	Representation and manipulation of genomic intervals and variables defined along a genome	10
limma	Gordon Smyth	Linear Models for Microarray Data	11
GenomeInfoDb	Bioconductor Package Maintainer	Utilities for manipulating chromosome and other 'seqname' identifiers	12
Biostrings	H. Pagès	Efficient manipulation of biological strings	13
DelayedArray	Hervé Pagès	Delayed operations on array-like objects	14
SummarizedExperiment	Bioconductor Package Maintainer	SummarizedExperiment container	15
annotate	Bioconductor Package Maintainer	Annotation for microarrays	16
Rsamtools	Bioconductor Package Maintainer	Binary alignment (BAM), FASTA, variant call (BCF), and tabix file import	17
genefilter	Bioconductor Package Maintainer	genefilter: methods for filtering genes from high-throughput experiments	18

Biostrings

The screenshot shows the Bioconductor website for the Biostrings package. The page has a teal header with the Bioconductor logo and navigation links: Home, Install, Help, Developers, and About. A search bar is located in the top right. The main content area features a breadcrumb trail: Home » Bioconductor 3.8 » Software Packages » Biostrings. Below this is the title 'Biostrings' in green. A set of filters includes 'platforms all', 'rank 13 / 1649', 'posts 12 / 0.7 / 2 / 2', 'in Bioc > 14 years', 'build warnings', and 'updated since release'. The DOI is 10.18129/B9.bioc.Biostrings, with social media icons for Facebook and Twitter. The main heading is 'Efficient manipulation of biological strings'. The text describes the package as a Bioconductor version 3.8 release, providing memory-efficient string containers and matching algorithms. It lists the authors (H. Pagès, P. Aboyoun, R. Gentleman, and S. DebRoy) and the maintainer (H. Pagès). A citation is provided for the package version 2.50.2. An 'Installation' section begins with the instruction to start R (version '3.5') and enter the following code:

```
To install this package, start R (version "3.5") and enter:
```

On the right side, there are two sidebar boxes. The 'Documentation' box lists links for Bioconductor documentation, including vignettes, workflows, course and conference material, videos, and community resources. The 'Support' box provides instructions on where to post questions, including the support site and the Bioc-devel mailing list.

Biostrings

こんな感じになります。①Biostringsの、②(Download数における)ランク情報。③Bioconductor上で提供されはじめてから14年以上経過しているのですね。

Bioconductor - Biostrings

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

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Home Install Help Developers About

Home » Bioconductor 3.8 » Software Packages » Biostrings

Biostrings ①

platforms all rank 13 / 1649 ② posts 12 / 0.7 / 2 / 2 in Bioc > 14 years ③

build warnings updated since release

DOI: [10.18129/B9.bioc.Biostrings](https://doi.org/10.18129/B9.bioc.Biostrings)

Efficient manipulation of biological strings

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.

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

Maintainer: H. Pagès <hpages at fredhutch.org>

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

Pagès H, Aboyoun P, Gentleman R, DebRoy S (2019). *Biostrings: Efficient manipulation of biological strings*. R package version 2.50.2.

Installation

To install this package, start R (version "3.5") and enter:

Documentation »

Bioconductor

- Package [vignettes](#) and manuals.
- [Workflows](#) for learning and use.
- [Course and conference](#) material.
- [Videos](#).
- Community [resources](#) and [tutorials](#).

R / [CRAN](#) packages and [documentation](#)

Support »

Please read the [posting guide](#). Post questions about Bioconductor to one of the following locations:

- [Support site](#) - for questions about Bioconductor packages
- [Bioc-devel](#) mailing list - for package developers

Biostrings

こんな感じになります。①Biostringsの、②(Download数における)ランク情報。③Bioconductor上で提供されはじめてから14年以上経過しているのですね。④allは、どのプラットフォーム(Windows, Mac, and Linux)でも利用可能だということを意味する。大抵の場合気にする必要はないが、ときどきWindowsのみ非対応のパッケージ(例: Rsubread)などがありますので気を付けましょう。

The screenshot shows the Bioconductor website for the Biostrings package. The page title is "Biostrings" and the subtitle is "Efficient manipulation of biological strings". The Bioconductor logo is in the top left. The page is divided into several sections: "Biostrings" (with a red arrow 1 pointing to the title), "platforms all" (with a red arrow 4 pointing to the word "all"), "rank 13 / 1649" (with a red arrow 2 pointing to the rank), "posts 12 / 0.7 / 2 / 2" (with a red arrow 3 pointing to the "in Bioc > 14 years" text), and "in Bioc > 14 years" (with a red arrow 3 pointing to the text). The "Documentation" section on the right lists links for "vignettes", "workflows", "course and conference material", "videos", "resources", and "tutorials". The "Support" section on the right lists links for "posting guide", "support site", and "bioc-devel mailing list".

Home » Bioconductor 3.8 » Software Packages » Biostrings

Biostrings

platforms all rank 13 / 1649 posts 12 / 0.7 / 2 / 2 in Bioc > 14 years

build warnings updated since release

DOI: [10.18129/B9.bioc.Biostrings](https://doi.org/10.18129/B9.bioc.Biostrings)

Efficient manipulation of biological strings

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.

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

Maintainer: H. Pagès <hpages at fredhutch.org>

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

Pagès H, Aboyoun P, Gentleman R, DebRoy S (2019). *Biostrings: Efficient manipulation of biological strings*. R package version 2.50.2.

Installation

To install this package, start R (version "3.5") and enter:

```
install.packages("Biostrings")
```

Documentation »

Bioconductor

- Package [vignettes](#) and manuals.
- [Workflows](#) for learning and use.
- [Course and conference](#) material.
- [Videos](#).
- Community [resources](#) and [tutorials](#).

R / [CRAN](#) packages and [documentation](#)

Support »

Please read the [posting guide](#). Post questions about Bioconductor to one of the following locations:

- [Support site](#) - for questions about Bioconductor packages
- [Bioc-devel](#) mailing list - for package developers

①Installationが左上にくる程度まで、②ページ下部に移動。

Biostrings

Bioconductor - Biostrings

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

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Home Install Help Developers About

Home » Bioconductor 3.8 » Software Packages » Biostrings

Biostrings

platforms all rank 13 / 1649 posts 12 / 0.7 / 2 / 2 in Bioc > 14 years

build warnings updated since release

DOI: [10.18129/B9.bioc.Biostrings](https://doi.org/10.18129/B9.bioc.Biostrings)

Efficient manipulation of biological strings

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.

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

Maintainer: H. Pagès <hpages at fredhutch.org>

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

Pagès H, Aboyoun P, Gentleman R, DebRoy S (2019). *Biostrings: Efficient manipulation of biological strings*. R package version 2.50.2.

Installation

To install this package, start R (version "3.5") and enter:

Documentation »

Bioconductor

- Package [vignettes](#) and manuals.
- [Workflows](#) for learning and use.
- [Course and conference](#) material.
- [Videos](#).
- Community [resources](#) and [tutorials](#).

R / [CRAN](#) packages and [documentation](#)

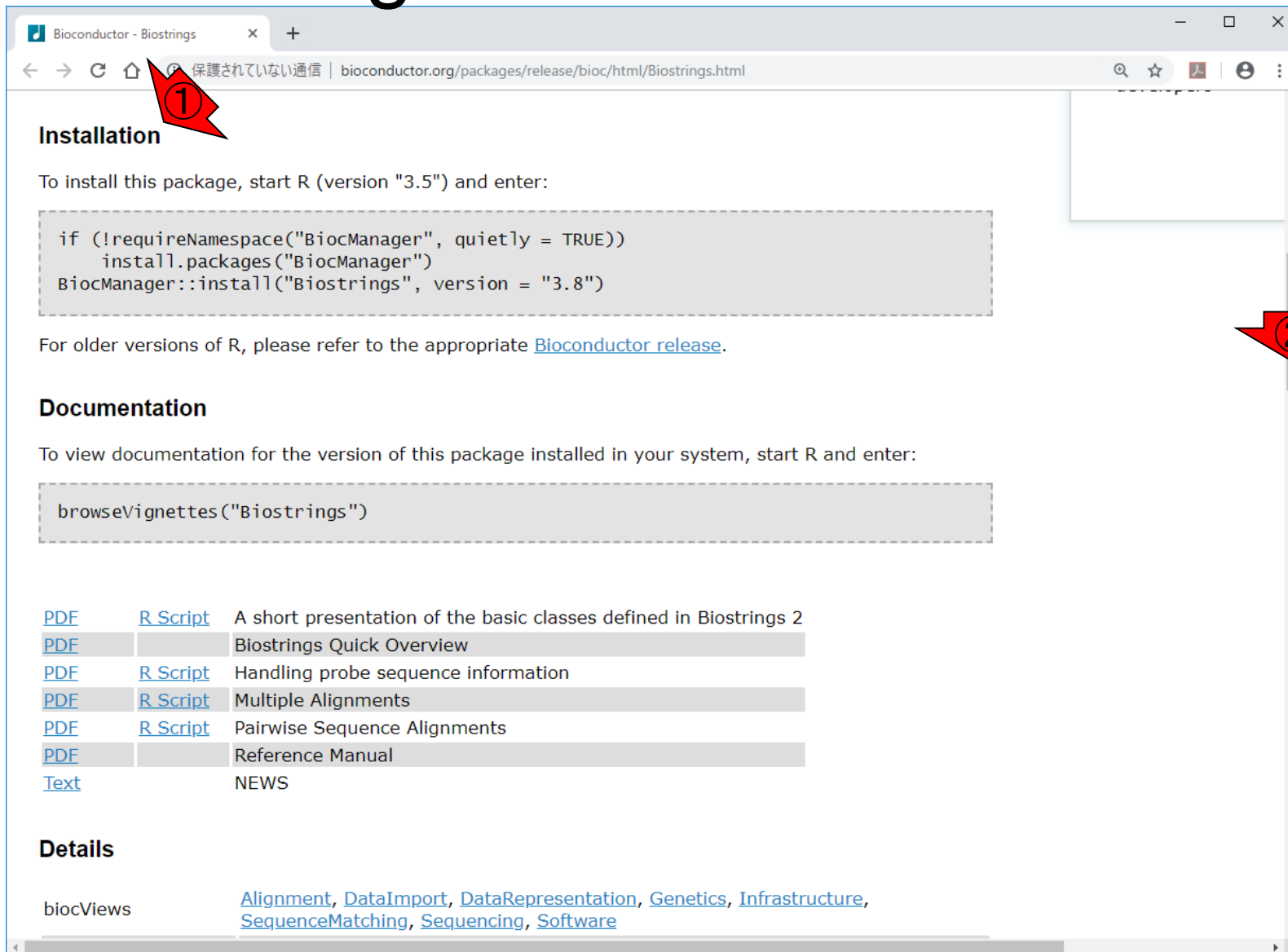
Support »

Please read the [posting guide](#). Post questions about Bioconductor to one of the following locations:

- [Support site](#) - for questions about Bioconductor packages
- [Bioc-devel](#) mailing list - for package developers

Biostrings

①Installationが左上にくる程度まで、②ページ下部に移動。こんな感じです。ちょっと拡大表示しています。



The screenshot shows a web browser window displaying the Bioconductor Biostrings package page. A red arrow labeled '1' points to the 'Installation' section, and another red arrow labeled '2' points to the bottom of the page. The page content includes installation instructions, documentation links, and a list of related resources.

Installation

To install this package, start R (version "3.5") and enter:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install("Biostrings", version = "3.8")
```

For older versions of R, please refer to the appropriate [Bioconductor release](#).

Documentation

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

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

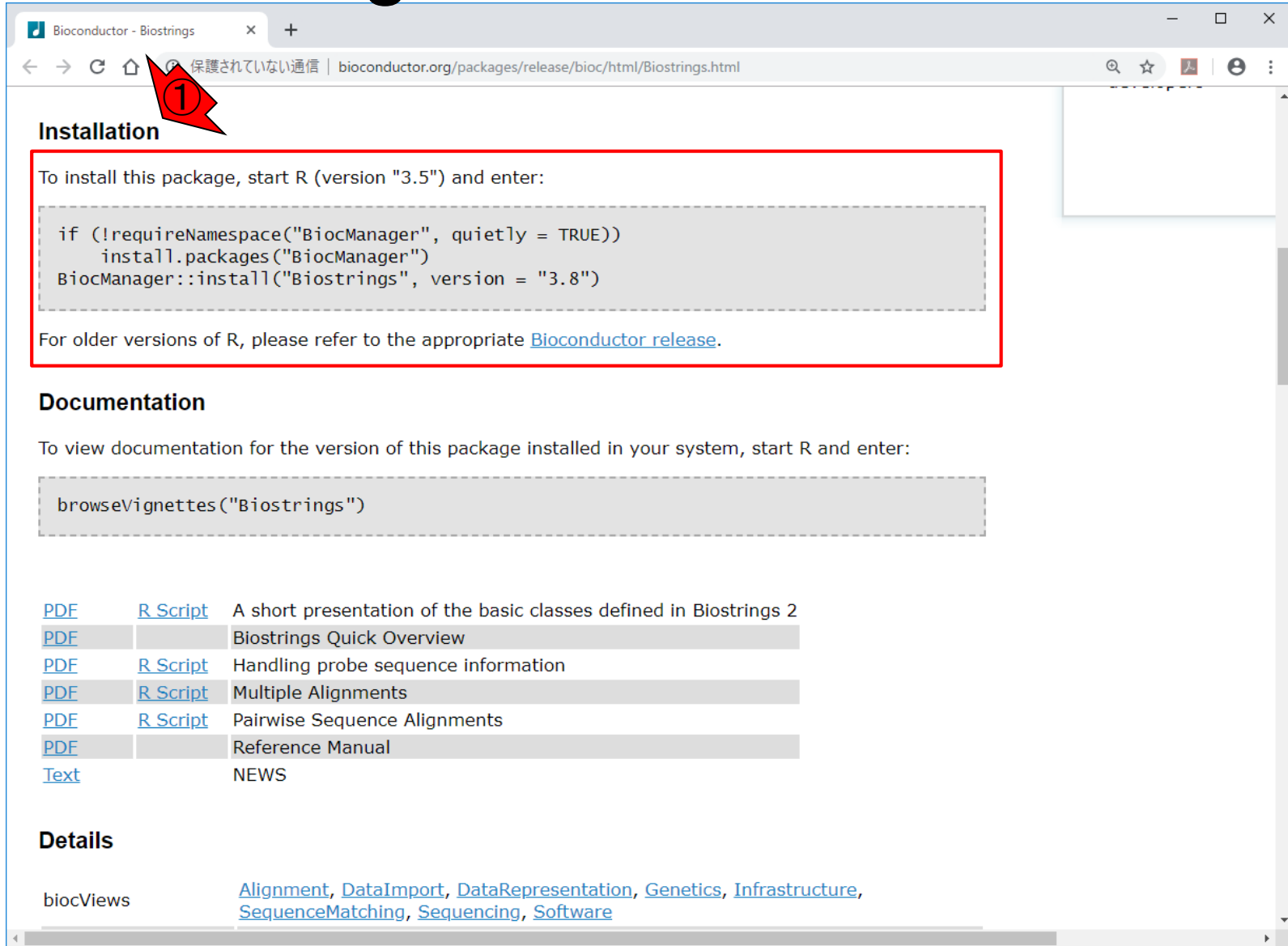
[PDF](#) [R Script](#) A short presentation of the basic classes defined in Biostrings 2
[PDF](#) [R Script](#) Biostrings Quick Overview
[PDF](#) [R Script](#) Handling probe sequence information
[PDF](#) [R Script](#) Multiple Alignments
[PDF](#) [R Script](#) Pairwise Sequence Alignments
[PDF](#)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

Details

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

①Biostringsパッケージのインストール法についての説明部分です。

Biostrings



Installation

To install this package, start R (version "3.5") and enter:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install("Biostrings", version = "3.8")
```

For older versions of R, please refer to the appropriate [Bioconductor release](#).

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](#)

Biostrings

①Biostringsパッケージのインストール法についての説明部分です。概ね1年以上Rを使い続け、バージョンアップに伴う不具合に遭遇するなどの実害を被らない限り身につかないところではあるが…②の3.5というのは、R本体を起動した際に最初に表示される、③3.5.1の部分に相当する情報です。

Bioconductor - Biostrings x +
← → ↻ 🏠 🔒 保護されていない通信 | bioconductor.org/packages/release/bioc/html/Biostrings.h

Installation

To install this package, start R (version "3.5") and enter:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install("Biostrings", version = "3.8")
```

For older versions of R, please refer to the appropriate [Bioconductor release](#).

Documentation

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

```
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](#), [Infrastru](#), [SequenceMatching](#), [Sequencing](#), [Software](#)

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 version 3.5.1 (2018-07-02) -- "Feather Sp$
Copyright (C) 2018 The R Foundation for Sta$
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$
一定の条件に従えば、自由にこれを再配布する$
配布条件の詳細に関しては、'license()' ある$

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$
```


Biostrings

①Biostringsパッケージのインストール法についての説明部分です。概ね1年以上Rを使い続け、バージョンアップに伴う不具合に遭遇するなどの実害を被らない限り身につかないところではあるが…②の3.5というのは、R本体を起動した際に最初に表示される、③3.5.1の部分に相当する情報です。実用上は、もし②と③の数値が異なっていたら、大抵R本体のバージョンが古いことを意味しますので、最新版のRをインストールしましょう。

Bioconductor - Biostrings x +
← → ↻ 🏠 🔒 保護されていない通信 | bioconductor.org/packages/release/bioc/html/Biostrings.h

Installation  

To install this package, start R (version "3.5") and enter:

```
if (!requireNamespace("BiocManager", quietly = TRUE))  
  install.packages("BiocManager")  
BiocManager::install("Biostrings", version = "3.8")
```

For older versions of R, please refer to the appropriate [Bioconductor release](#).

Documentation

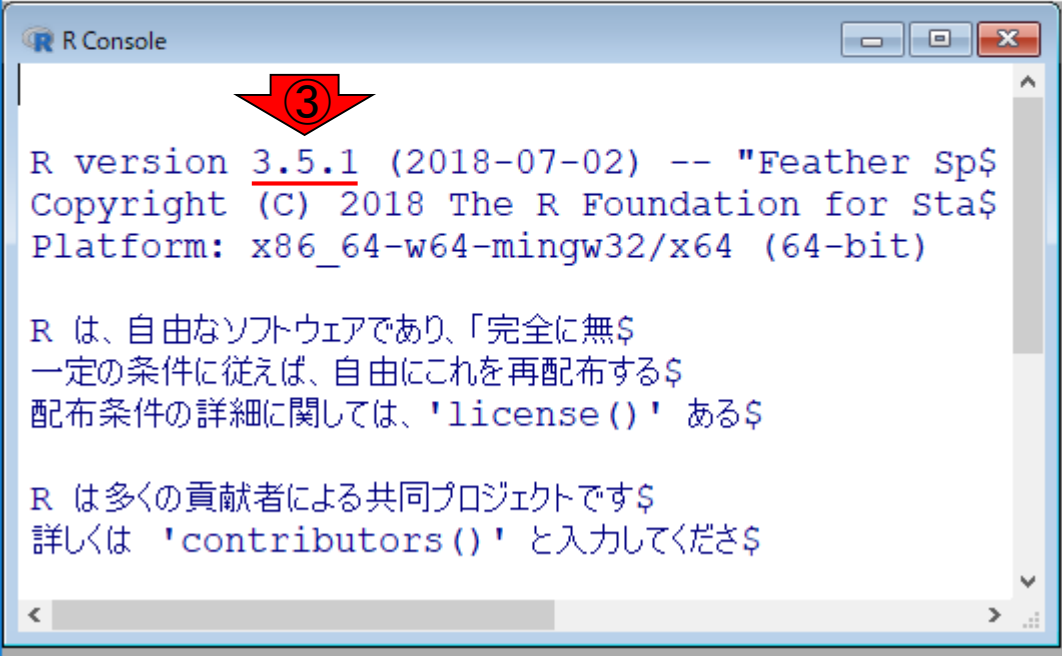
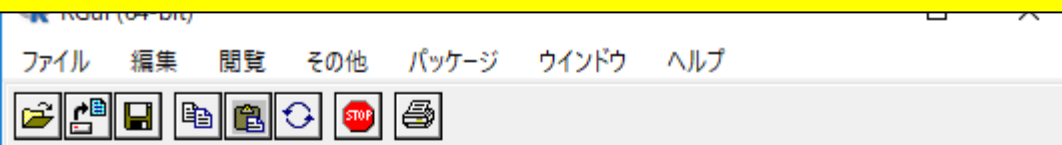
To view documentation for the version of this package installed in your system, start R

```
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](#), [Infrastru](#), [SequenceMatching](#), [Sequencing](#), [Software](#)



Biostrings

「R本体のバージョン」と対応した「Bioconductorのリリース(バージョンのこと)」を提供することできっちり動作確認をしているのですが、慣れないうちはその説明自体が意味不明だと思います。ですので、とにかく最初のうちは、**最新版のR本体をインストールしておけば大抵問題ない**、と覚えておけばよいです。①がR本体とBioconductorのバージョンの対応表です。

Bioconductor - Biostrings x +
 保護されていない通信 | bioconductor.org/packages/release/bioc/html/Biostrings.h

Installation

To install this package, start R (version "3.5") and enter:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install("Biostrings", version = "3.8")
```

For older versions of R, please refer to the appropriate [Bioconductor release](#).



Documentation

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

```
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](#), [Infrastru](#)
[SequenceMatching](#), [Sequencing](#), [Software](#)

RGui (64-bit) window showing the R Console output:

```
R version 3.5.1 (2018-07-02) -- "Feather Sp$
Copyright (C) 2018 The R Foundation for Sta$
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$
一定の条件に従えば、自由にこれを再配布する$
配布条件の詳細に関しては、'license()' ある$

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$
```

Contents

- 任意のキーワードを含む行を抽出するコードの解説の続き
 - おさらい、行列要素の抽出、`as.character`、`is.element`
 - 行列要素の抽出、`write.table`
- multi-FASTAファイルからの各種情報抽出
 - 基本情報取得(コンティグ数、配列長、N50、GC含量)
 - 課題1
 - コード内部の説明、`readDNAStringSet`、`width`、`sum`、`length`
 - サブセットの抽出(subsetting)、条件判定
 - 実践的な使い方(指定した長さ以上の配列を取得)
- Rパッケージの話
 - Bioconductor概観、Biostrings、推奨インストール手順、マニュアルなど
 - 課題2と課題3

見るだけ

インストール済み

Biostringsの具体的なインストール方法は、①のコマンドを、②R Console画面上でコピペ実行するだけです。③enterというのはそういう意味です。実際にはやる必要はありませんし、やらないでください！

Installation

To install this package, start R (version "3.5") and enter:

```
if (!requireNamespace("BiocManager", quietly = TRUE))  
  install.packages("BiocManager")  
BiocManager::install("Biostrings", version = "3.8")
```

For older versions of R, please refer to the appropriate [Bioconductor release](#).

Documentation

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

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

Details

Alignment, DataImport, DataRepresentation, Genetics, Infrastructure, SequenceMatching, Sequencing, Software

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 version 3.5.1 (2018-07-02) -- "Feather Sp$  
Copyright (C) 2018 The R Foundation for Sta$  
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\$
一定の条件に従えば、自由にこれを再配布する\$
配布条件の詳細に関しては、'license()' ある\$

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\$

推奨インストール手順

おさらい。持込および貸与PCは、①のリンク先から辿れる推奨手順(実質的に②のPDFファイル)に従って、「R本体およびパッケージのインストール」が行われています。

(Rで)塩基配列解析

(last modified 2019/04/05, 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](#) **①** [Windows2018.11.15版とMacintosh2018.11.27版](#) **②** 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**([Windows2019.03.12版](#)と [Macintosh2019.03.12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**に移行しました。(2018/07/18)

What's new? ([過去のお知らせはこちら](#))

- 「[解析 | 一般 | アラインメント | について](#)」だった項目名を「[解析 | 一般 | アラインメント | ペアワイズ | について](#)」と「[解析 | 一般 | アラインメント | マルチプル | について](#)」に分離しました。(2019/04/05) **NEW**
- 「[カウント情報取得 | シミュレーションデータ | について](#)」だった項目名を「[カウント情報取得 | シミュレーションデータ | RNA-seq | について](#)」に変更しました。また、「[カウント情報取得 | シミュレーションデータ | scRNA-seq | について](#)」も追加しました。(2019/04/05) **NEW**
- 「[インストール | Rパッケージ | 必要最小限プラスアルファ](#)」を更新しました。(2019/04/05) **NEW**
- 「[解析 | 前処理 | scRNA-seq | について](#)」を追加しました。(2019/04/04) **NEW**
- 「[解析 | クラスタリング | について](#)」だった項目名を、(bulk) RNA-seq用の「[解析 | クラスタリング | RNA-seq | について](#)」と、scRNA-seq用の「[解析 | クラスタリング | scRNA-seq | について](#)」に変更しました。それに伴い、中身や関連する項目名も変更しました。(2019/04/03) **NEW** [トップページへ](#)
- 「[カウント情報取得 | シミュレーションデータ | について](#)」を更新しました。(2019/04/03) **NEW**

推奨インストール手順

②のパッケージのインストール部分のおさらいです。③少しずつ下矢印キーを押していく。

(Rで)塩基配列解析

保護されていない通信 | www.iu.a.u-tokyo.ac.jp/~kadota/r_seq.html

(Rで)塩基配列解析

(last modified 2019/04/05, 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](#)([Windows2019.03.12版](#)と[Macintosh2019.03.12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? (過去のお知らせはこちら)

- 「[解析 | 一般 | アラインメント | について](#)」だった項目名を「[解析 | 一般 | アラインメント | マルチプル | について](#)」に変更しました。
- 「[カウント情報取得 | シミュレーションデータ | について](#)」だった項目名を「[カウント情報取得 | シミュレーションデータ | RNA-seq | について](#)」に変更しました。また、「[カウント情報取得 | シミュレーションデータ | について](#)」も追加しました。(2019/04/05) **NEW**
- 「[インストール | Rパッケージ | 必要最小限プラスアルファ](#)」を更新しました。
- 「[解析 | 前処理 | scRNA-seq | について](#)」を追加しました。(2019/04/03) **NEW**
- 「[解析 | クラスタリング | について](#)」だった項目名を、(bulk) RNA-seq用の「[解析 | クラスタリング | RNA-seq | について](#)」と、scRNA-seq用の「[解析 | クラスタリング | scRNA-seq | について](#)」に変更しました。(2019/04/03) **NEW**
- 「[カウント情報取得 | シミュレーションデータ | について](#)」を更新しました。(2019/04/03) **NEW**

[トップページへ](#)


推奨インストール手順

②のパッケージのインストール部分のおさらいです。③少しずつ下矢印キーを押していく。④「必要最小限プラスアルファ」をクリック。

(Rで塩基配列解析

保護されていない通信 | www.iu.a.u-tokyo.ac.jp/~kadota/r_seq.html

- [はじめに](#) (last modified 2018/08/04)
- [過去のお知らせ](#) (last modified 2019/04/03) **NEW**
- [インストール | について](#) (last modified 2019/01/24)
- インストール | R本体 | 最新版 | [Win用](#) (last modified 2019/01/24)推奨
- インストール | R本体 | 最新版 | [Mac用](#) (last modified 2019/01/24)推奨
- インストール | R本体 | 過去版 | [Win用](#) (last modified 2015/03/22)
- インストール | R本体 | 過去版 | [Mac用](#) (last modified 2015/03/22)
- [インストール | Rパッケージ | について](#) (last modified 2018/11/13)
- インストール | Rパッケージ | [ほぼ全て\(削除予定\)](#) (last modified 2015/05/25)
- インストール | Rパッケージ | [必要最小限プラスアルファ](#) **④** (last modified 2019/04/05)推奨 **NEW**
- インストール | Rパッケージ | [必要最小限\(削除予定\)](#) (last modified 2015/05/25)
- インストール | Rパッケージ | [個別\(2018年11月以降\)](#) (last modified 2018/11/12)
- 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 2019/03/12) **NEW**
- [サンプルデータ](#) (last modified 2018/06/09)
- 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 2016/04/20)
- 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)

[トップページへ](#)

推奨インストール手順

②のパッケージのインストール部分のおさらいです。③少しずつ下矢印キーを押していく。④「必要最小限プラスアルファ」をクリック。こんな感じになります。⑤の部分でやってもらったのが、このウェブサイト上で使うパッケージ群やアグリバイオの講義で使用予定のパッケージ群を一度にコピーでインストールするところです。

(Rで)塩基配列解析

保護されていない通信 | www.iu.a.u-tokyo.ac.jp/~kadota/r_s

インストール | Rパッケージ | 必要最小限プラスアルファ

アグリバイオで所有するノートPCは、Rパッケージの2大リポジトリであるCRANと [Bioconductor](#) (およびGithub) から提供されている以下のパッケージ群をインストールしています。30分程度でインストールが完了します(東大の有線LAN環境)。

1. R本体を起動

2. パッケージ群のインストール



以下を「R コンソール画面」上でコピー&ペースト。どこからダウンロードするか?と聞かれるので、その場合は自分から近いサイトを指定。

```
#前処理(BiocManagerがなければインストール)
if (!requireNamespace("BiocManager", quietly=T))#BiocManagerパッケージがインストールされてなければ...
  install.packages("BiocManager") #BiocManagerをインストールせよ

#本番1(CRANから提供されているパッケージ群)
BiocManager::install("ape", update=F)
BiocManager::install("bio3d", update=F)#2018.11.16追加
BiocManager::install("blockmodeling", update=F)
BiocManager::install("bit", update=F) #2019.04.05追加
BiocManager::install("cclust", update=F)
BiocManager::install("class", update=F)
BiocManager::install("cluster", update=F)
BiocManager::install("clValid", update=F)
BiocManager::install("corrplot", update=F)#2019.01.22追加
BiocManager::install("data.table", update=F)#2018.11.20追加
BiocManager::install("devtools", update=F)#2018.11.20追加
BiocManager::install("dplyr", update=F)#2018.11.20追加
BiocManager::install("DT", update=F) #2018.11.20追加
```

[トップページへ](#)

推奨インストール手順

②のパッケージのインストール部分のおさらいです。③少しずつ下矢印キーを押していく。④「必要最小限プラスアルファ」をクリック。こんな感じになります。⑤の部分でやってもらったのが、このウェブサイト上で使うパッケージ群やアグリバイオの講義で使用予定のパッケージ群を一度にコピーでインストールするところです。⑥ Biostringsはこのあたりに書き込まれています。最初に多少時間がかかっても一気にパッケージ群をインストールしているおかげで、その後は心穏やかに解析部分により多くの労力を割くことができるのです。

(Rで)塩基配列解析

保護されていない通信 | www.iu.a.u-tokyo.ac.jp/~kadota/r_s

1. R本体を起動

2. パッケージ群のインストール

以下を「R コンソール画面」上でコピー&ペースト。どこからダウンロードするか近いサイトを指定。

#本番2(Bioconductorから提供されているゲノム配列以外のパッケージ群)

```

BiocManager::install("affy", update=F)
BiocManager::install("agilp", update=F)
BiocManager::install("annotate", update=F)
BiocManager::install("ArrayExpress", update=F)
BiocManager::install("baySeq", update=F)
BiocManager::install("beadarray", update=F)
BiocManager::install("BeadDataPackR", update=F)
BiocManager::install("betr", update=F)
BiocManager::install("BHC", update=F)
BiocManager::install("biomaRt", update=F)
BiocManager::install("Biostrings", update=F)
BiocManager::install("BSgenome", update=F)
BiocManager::install("bsseq", update=F)
BiocManager::install("Category", update=F)
BiocManager::install("ChIPpeakAnno", update=F)
BiocManager::install("chipseq", update=F)
BiocManager::install("ChIPseqR", update=F)
BiocManager::install("ChIPsim", update=F)
BiocManager::install("clusterStab", update=F)
BiocManager::install("cosmo", update=F)
BiocManager::install("CSAR", update=F)

```



⑥

[トップページへ](#)

Contents

- 任意のキーワードを含む行を抽出するコードの解説の続き
 - おさらい、行列要素の抽出、`as.character`、`is.element`
 - 行列要素の抽出、`write.table`
- multi-FASTAファイルからの各種情報抽出
 - 基本情報取得(コンティグ数、配列長、N50、GC含量)
 - 課題1
 - コード内部の説明、`readDNAStringSet`、`width`、`sum`、`length`
 - サブセットの抽出(subsetting)、条件判定
 - 実践的な使い方(指定した長さ以上の配列を取得)
- Rパッケージの話
 - Bioconductor概観、Biostrings、推奨インストール手順、マニュアルなど
 - 課題2と課題3

Biostrings

Biostringsパッケージのサイトをもう一度拡大して表示しているだけです。①Documentation部分の、特に②の赤枠部分の項目数や情報量は、パッケージごとに異なります。③をクリック

The screenshot shows a web browser window with the URL `bioconductor.org/packages/release/bioc/html/Biostrings.html`. The page is titled "Bioconductor - Biostrings" and has a tab icon. The browser's address bar shows "保護されていない通信 | bioconductor.org/packages/release/bioc/html/Biostrings.html".

Installation

To install this package, start R (version "3.5") and enter:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install("Biostrings", version = "3.8")
```

For older versions of R, please refer to the appropriate [Bioconductor release](#).

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

Biostrings

こんな感じになります。最初は文字が小さいと思いますので、適宜「Ctrlキーとプラス(+)キー」を押すなどして拡大表示してください。

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
<code>length</code>	Return the number of sequences in an object.
<code>names</code>	Return the names of the sequences in an object.
<code>[</code>	Extract sequences from an object.
<code>head, tail</code>	Extract the first or last sequences from an object.
<code>rev</code>	Reverse the order of the sequences in an object.
<code>c</code>	Combine in a single object the sequences from 2 or more objects.
<code>width, nchar</code>	Return the sizes (i.e. number of letters) of all the sequences in an object.
<code>==, !=</code>	Element-wise comparison of the sequences in 2 objects.
<code>match, %in%</code>	Analog to <code>match</code> and <code>%in%</code> on character vectors.
<code>duplicated, unique</code>	Analog to <code>duplicated</code> and <code>unique</code> on character vectors.
<code>sort, order</code>	Analog to <code>sort</code> and <code>order</code> on character vectors, except that the ordering of DNA or Amino Acid sequences doesn't depend on the

Biostrings

①length関数は配列数を調べる際に、そして②width関数は配列ごとの塩基数を調べる際に使いましたね。③少しページ下部に移動。

Biostrings Quick Overview

Hervé Pagès
Fred Hutchinson Cancer Research Center
Seattle, WA

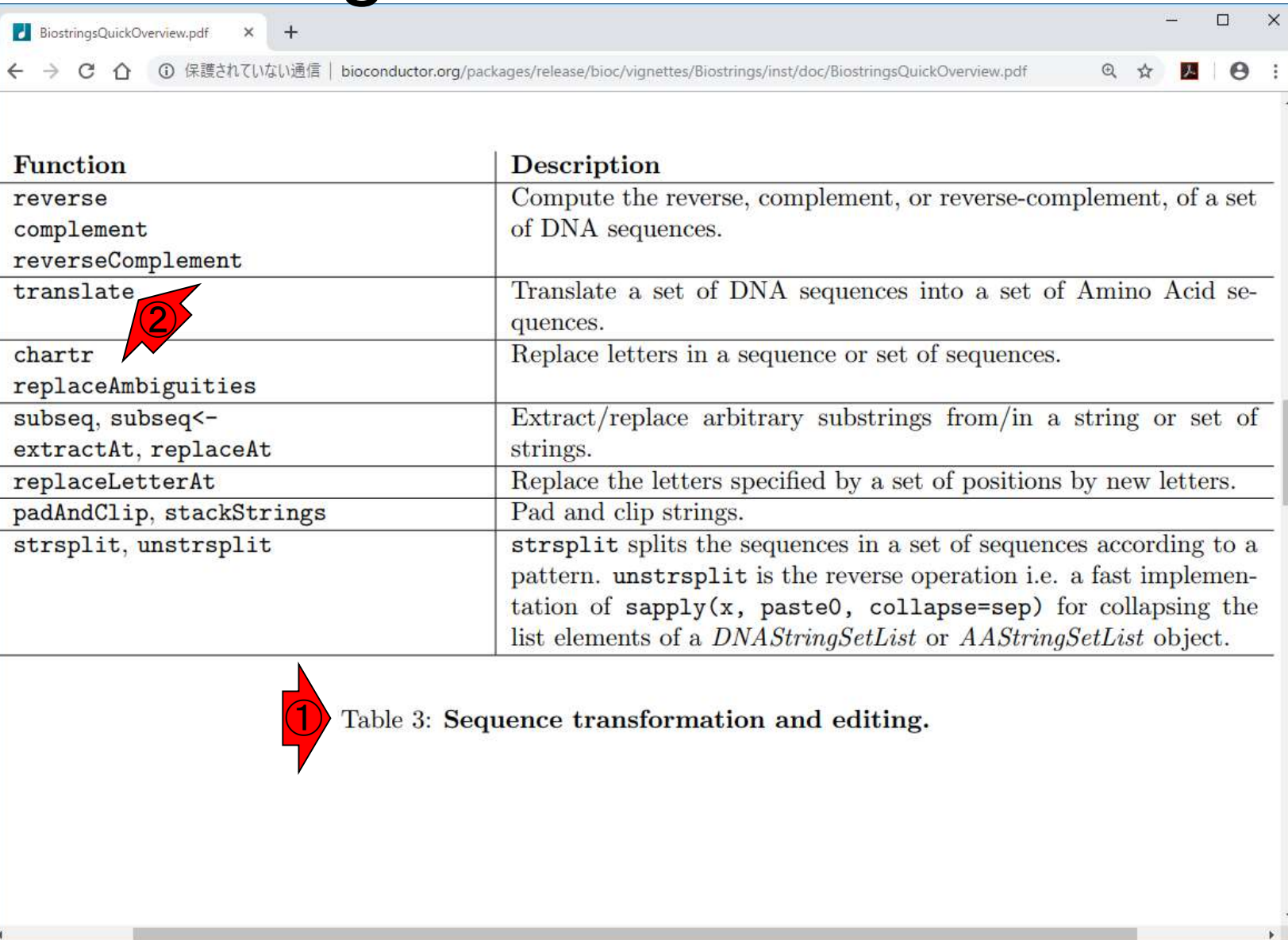
January 3, 2019

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.
match, %in%	Analog to <code>match</code> and <code>%in%</code> on character vectors.
duplicated, unique	Analog to <code>duplicated</code> and <code>unique</code> on character vectors.
sort, order	Analog to <code>sort</code> and <code>order</code> on character vectors, except that the ordering of DNA or Amino Acid sequences doesn't depend on the

Biostrings

2ページ目の上のほうに、①Table 3があります。翻訳配列取得の際に用いた、②translate関数はここにあります。

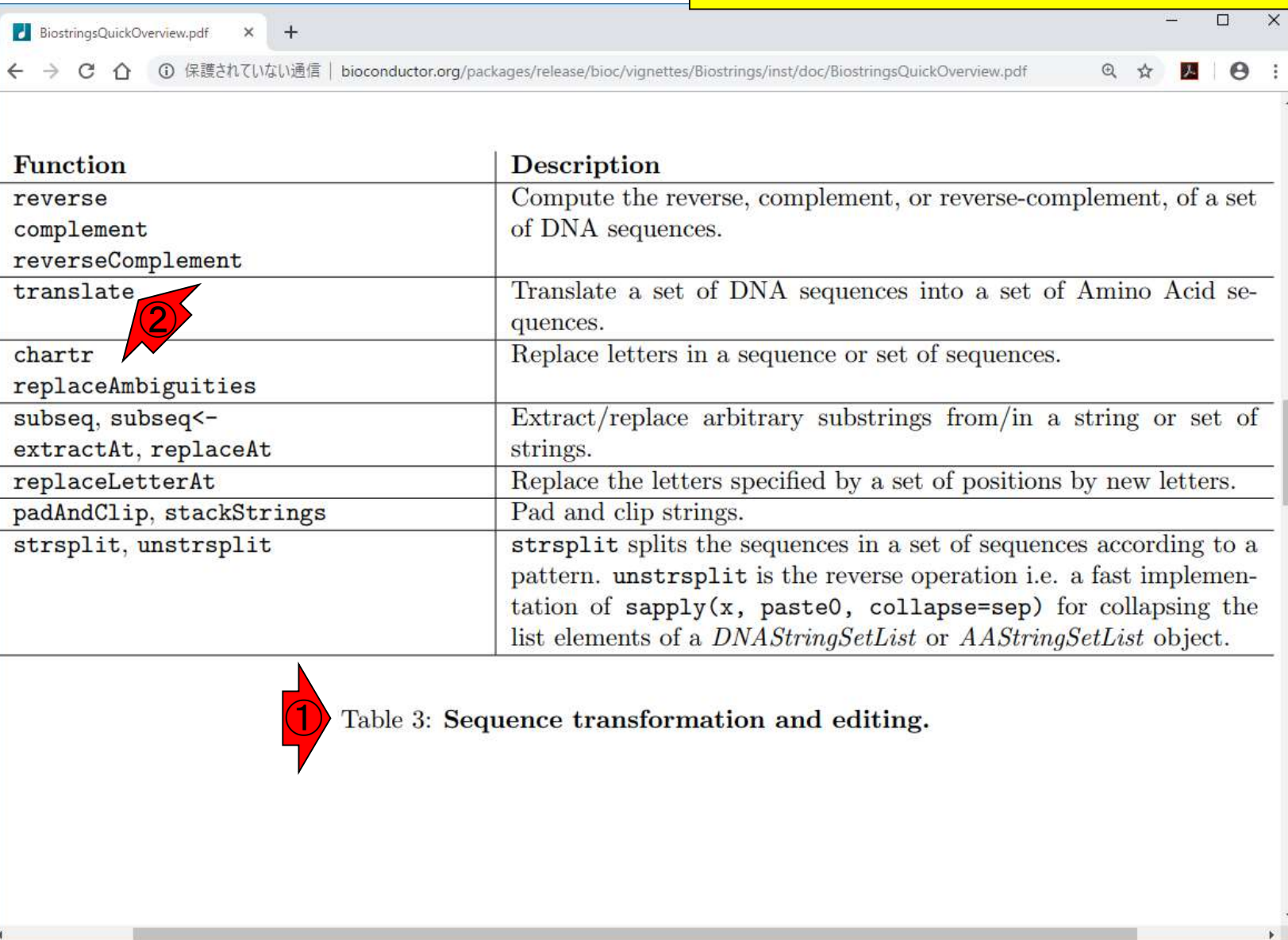


Function	Description
reverse complement reverseComplement	Compute the reverse, complement, or reverse-complement, of a set of DNA sequences.
translate	Translate a set of DNA sequences into a set of Amino Acid sequences.
chartr replaceAmbiguities	Replace letters in a sequence or set of sequences.
subseq, 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	<code>strsplit</code> splits the sequences in a set of sequences according to a pattern. <code>unstrsplit</code> 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.

課題2

Biostringsパッケージが提供する②translate以外の関数を1つ挙げ、簡単に説明せよ。もちろん①Table 3以外の関数でもよい。



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.

課題2のヒント

私は、①Table 3で見えているような比較的分かりやすい関数をたよりに、(一部Biostrings以外のものもありますが)②赤枠内で見えているような項目を習得しました。

Function	Description
reverse	Compute the reverse complement or reverse-complement of a set
complement	
reverseComplement	
translate	
chartr	
replaceAmbiguities	
subseq, subseq<-	
extractAt, replaceAt	
replaceLetterAt	
padAndClip, stackStrings	
strsplit, unstrsplit	

① Table 3: Sequen

- 基本的な利用法 (last modified 2019/03/12) NEW
- サンプルデータ (last modified 2018/06/09)
- 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 2016/04/20)
- 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 (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 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)

②

トップページ

課題3

Bioconductorが提供するパッケージ群を眺め、有用だと思ったBiostrings以外のパッケージを1つ挙げ、その概要を簡単に説明せよ。

Bioconductor - BiocViews

bioconductor.org/packages/release/BiocViews.html#___Software

Search:

Home Install Help Developers About

Home » BiocViews

All Packages

Bioconductor version 3.8 (Release)

Autocomplete biocViews search:

Packages found under Software:

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

Show 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
AnnotationDbi	Bioconductor Package	Annotation Database Interface	6

課題3補足説明

ゲノム情報解析と無関係のパッケージでもよいが、Bioconductorに限定しているのもので、それ以外のサイトで提供されているものは×とします。また、ver. 3.8のリリースに含まれる1649パッケージの中から選んでください。

Bioconductor - BioViews

bioconductor.org/packages/release/BiocViews.html#_Software

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Home Install Help Developers About

Home » BioViews

All Packages

Bioconductor version 3.8 (Release)

Autocomplete biocViews search:

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
AnnotationDbi	Bioconductor Package	Annotation Database Interface	6

- Software (1649)
 - AssayDomain (661)
 - BiologicalQuestion (668)
 - Infrastructure (360)
 - ResearchField (729)
 - StatisticalMethod (572)
 - Technology (1049)
 - WorkflowStep (884)
- AnnotationData (942)
- ExperimentData (360)
- Workflow (23)