

# バイオインフォマティクス人材育成カリ キュラム(次世代シークエンサ)速習 コース

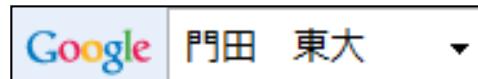
## 3. データ解析基礎 | 3-1. R 基礎1

東京大学・大学院農学生命科学研究科  
アグリバイオインフォマティクス教育研究ユニット

門田幸二(かどた こうじ)

[kadota@iu.a.u-tokyo.ac.jp](mailto:kadota@iu.a.u-tokyo.ac.jp)

<http://www.iu.a.u-tokyo.ac.jp/~kadota/>



# ファイルのダウンロード

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

## (Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランскриプトーム、正規化、発現変動、統計、モデル、バイオインフォマティクス～  
(last modified 2014/08/22, since 2010)

### What's new?

- このウェブページはフリーソフトRと必要なパッケージをインストール済みである前提で記述しています。初心者は、
  - Rのインストールと起動
  - 基本的な利用法で自習してください。(2014/07/21)
- 2014年10月04日にHPCワークショップ「医療とビッグデータ解析」(9:00-9:20)に引き続いで 中級者向けバイオインフォマティクス入門講習会@仙台国際センター(10:50-12:20)で話します。興味ある方はどうぞ。(2014/07/23)
- 門田幸二著シリーズ Useful R 第7巻トランскриプトーム解析刊行(共立出版)

---

- バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ) | 速習コースで利用する計算機環境構築する一通りの手順を公開しました。(2014/08/11) NEW
- 日本乳酸菌学会誌のNGS関連連載の第1回分PDFを公開しました。関連項目は[こちら](#)。(2014/08/03) NEW
- 参考資料(講義、講習会、本など)の項目を更新しました。(2014/08/19) NEW

---

- はじめに (last modified 2014/01/30)
- 参考資料(講義、講習会、本など) (last modified 2014/08/19) NEW
- 過去のお知らせ (last modified 2014/08/03) NEW
- Rのインストールと起動 (last modified 2014/07/31) NEW
- 基本的な利用法 (last modified 2014/07/20)
- サンプルデータ (last modified 2014/07/17)
- バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ) | 速習コース (last modified 2014/08/22) NEW
- 書籍 |トランскриプトームについて (last modified 2014/05/12)
- 書籍 |トランскриプトーム解析 | 2.3.1 RNA-seqデータ(FASTQファイル) (last modified 2014/04/15) [トップページへ](#)
- 書籍 |トランскриプトーム解析 | 2.3.2 リファレンス配列 (last modified 2014/04/16)

# ファイルのダウンロード

## バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ) | 速習コース NEW

2014年9月にJST-NBDCと東大農アグリバイオ主催で「バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ)速習コース」が開催されます。主催機関のサイト上で情報提供したほうがいいだうということで、受講者が各自でインストールするソフトウェアや、イメージファイルのダウンロードなど準備していただく計算機環境の情報などを示します。

バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ)関連:

- ・NBDCの[速習コース案内サイト](#)(速習コース主催機関)
- ・HPCIの[速習コース受講申込受付サイト](#)(速習コース共催機関)
- ・カリキュラムを策定した[NBDC](#)
- ・「NBDCで実施した調査」
  - 「バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ)」
  - 「カリキュラムで習得する知識」
  - 「カリキュラム フロー」

hoge.zipをデスクトップにダウンロードして解凍しておきましょう

- ・2014年9月5日15:00-18:15、「2-2. バイオ系データベース概論」、初級、実習
- ・小野浩雅(DBCLS)、統合TV、講義資料
- ・基本的な各種バイオ系データベースの理解、統合DBの利用法。
- ・2014年9月8日10:30-12:00、「3-1. R 基礎1」、初級、実習
- ・門田幸二(東京大学)、統合TV、講義資料(20140822, 18:04版)
- ・Rインストール自体は基本的に終了した状態を想定しているものの、最初にlibrary(Biostrings)などいくつかの利用予定パッケージのロードを行い、パッケージのインストールがうまくいっているかどうかを確認(できていなかったヒトの同定および対処)。Rの一般的な利用法。`log`関数などの基本的かつ挙動を完全に把握できる関数を例として、関数内部のオプション変更や「?関数名」で利用法の幅を広げる基本テクを概観。`exp`, `mean`, `median`, `sort`, `length`関数。
  - 9/8-9の2日間で用いるファイル群:[hoge.zip](#) (20140822, 17:24版)
  - Rコード:[rcode 20140908.txt](#)
- ・2014年9月8日13:15-14:45、「3-2. R 基礎2」、初級、実習
- ・門田幸二(東京大学)、統合TV、講義資料(20140811, 0:11版)
- ・翻訳配列の取得を例に「(Rで)塩基配列解析」の基本的な利用法を紹介。塩基配列中にNを

計算機環境構築(Linux系):

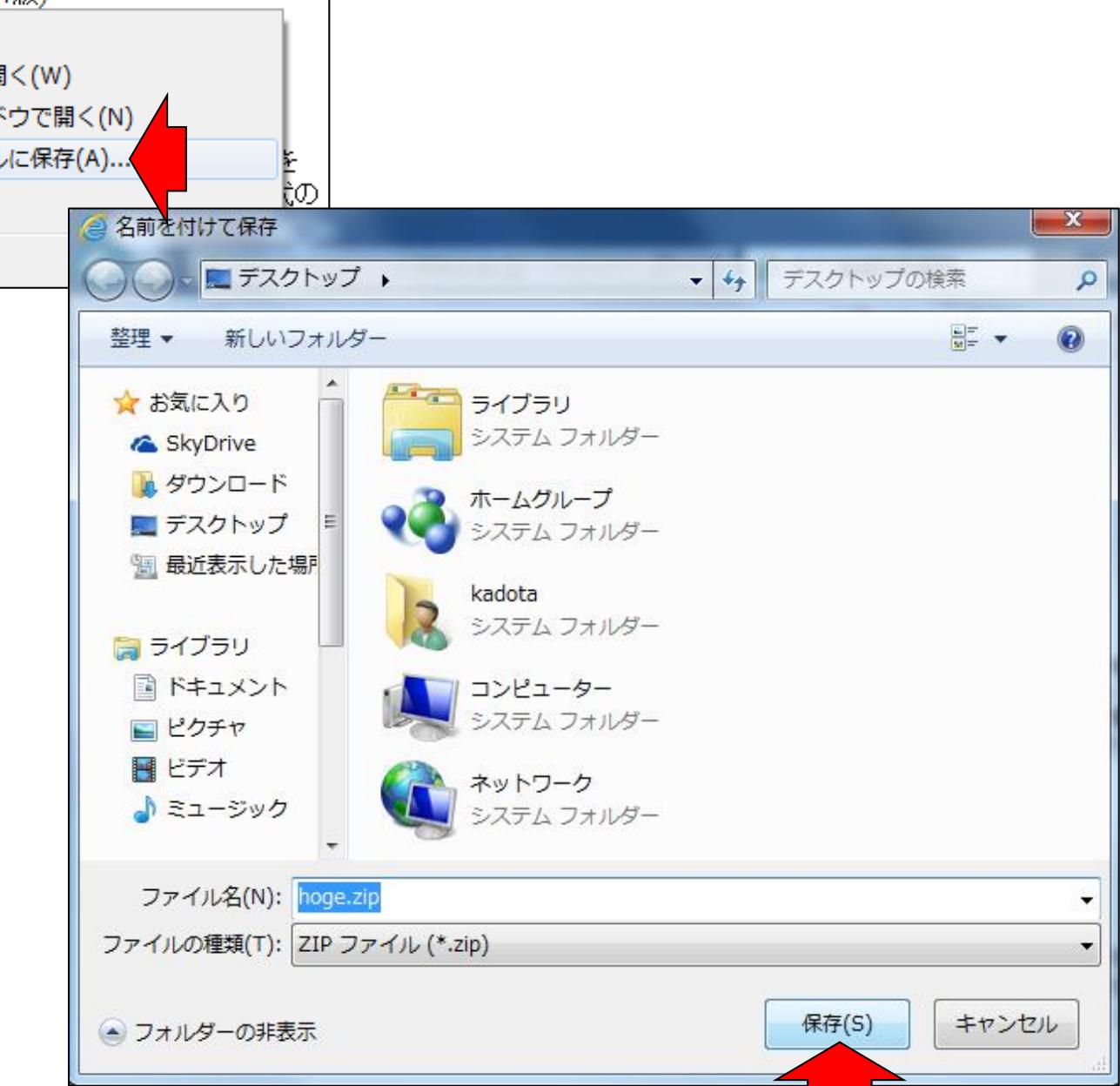
8月初旬をめどにBio-Linuxにブランチ

- ・VirtualBox
  - VirtualBoxをWind...[VirtualBox](#)
- ・Bio-Linux:[Field et al., Nature Methods](#)

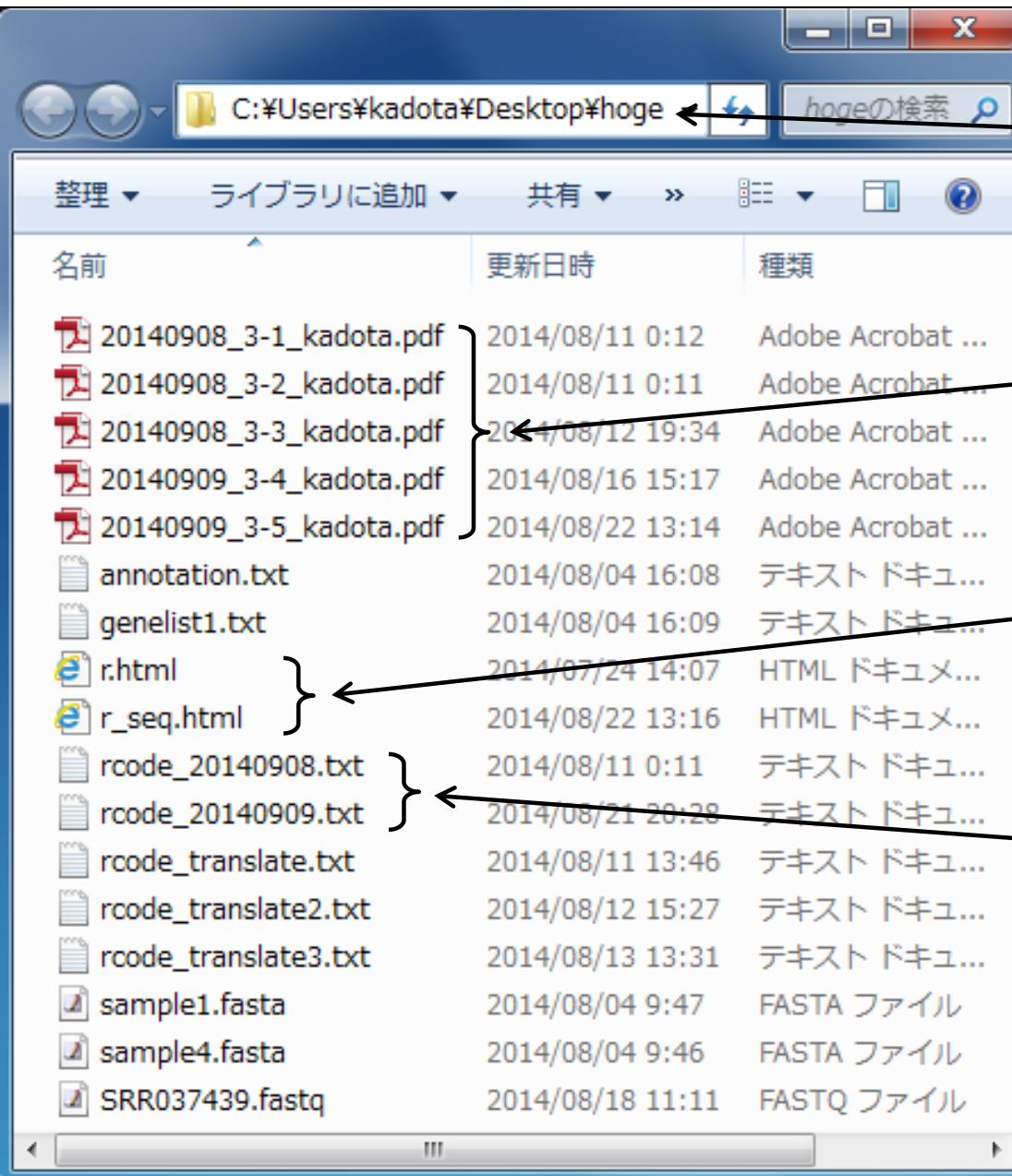
- 2014年9月8日10:30-12:00、「3-1. R 基礎1」、初級、実習
- 門田幸二(東京大学)、統合TV、講義資料(20140822, 18:04版)
- Rインストール自体は基本的に終了した状態を想定しているものの、最初にlibrary(Biostrings)などいくつかの利用予定パッケージのロードを行い、パッケージのインストールがうまくいっているかどうかを確認(できていなかったヒトの同定および対処)。Rの一般的な利用法。log関数などの基本的かつ挙動を完全に把握できる関数を例として、関数内部のオプション変更や「?関数名」で利用法の幅を広げる基本テクを概観。exp, mean, median, sort, length関数。

- 9/8-9の2日間で用いるファイル群: [hoge.zip](#) (20140822, 17:24版)
- Rコード: [rcode 20140908.txt](#)

- 2014年9月8日13:15-14:45、「3-2. R 基礎2」、初級
- 門田幸二(東京大学)、統合TV、講義資料(20140822, 18:04版)
- 翻訳配列の取得を例に「(Rで)塩基配列解析」の含む場合のエラー例とその対処法。RGui画面タブ区切りテキストファイルからの情報抽出。Pandasとその対処法。



# ファイルのダウンロード



デスクトップ上にhogeフォルダがあり、サンプルファイルが見られるという前提で行います。

9/8-9の講義資料PDF

ダブルクリックでローカルに参考ウェブページを開けます。ネットワーク不調時や負荷軽減用。

9/8-9の講義資料PDF中のRコード。コピペ用。

# NGS速習コース全貌のおさらい

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

## (Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランскриプトーム、正規化、発現変動、統計、モデル、バイオインフォマティクス～  
(last modified 2014/08/22, since 2010)

### What's new?

- このウェブページはフリーソフトRと必要なパッケージをインストール済みである前提で記述しています。初心者は、
  - Rのインストールと起動
  - 基本的な利用法で自習してください。(2014/07/21)
- 2014年10月04日にHPCワークショップ「医療とビッグデータ解析」(9:00-9:20)に引き続いで 中級者向けバイオインフォマティクス入門講習会@仙台国際センター(10:50-12:20)で話します。興味ある方はどうぞ。(2014/07/23)
- 門田幸二著シリーズ Useful R 第7巻トランскриプトーム解析刊行(共立出版)

---

- バイオインフォマティクス人材育成カリキュラム(次世代シークエンサ) | 速習コースで利用する計算機環境構築する一通りの手順を公開しました。(2014/08/11) NEW
- 日本乳酸菌学会誌のNGS関連連載の第1回分PDFを公開しました。関連項目は[こちら](#)。(2014/08/03) NEW
- 参考資料(講義、講習会、本など)の項目を更新しました。(2014/08/19) NEW

---

- はじめに (last modified 2014/01/30)
- 参考資料(講義、講習会、本など) (last modified 2014/08/19) NEW
- 過去のお知らせ (last modified 2014/08/03) NEW
- Rのインストールと起動 (last modified 2014/07/31) NEW
- 基本的な利用法 (last modified 2014/07/20)
- サンプルデータ (last modified 2014/07/17)
- バイオインフォマティクス人材育成カリキュラム(次世代シークエンサ) | 速習コース (last modified 2014/08/22) NEW
- 書籍 |トランскриプトームについて (last modified 2014/05/12)
- 書籍 |トランскриプトーム解析 | 2.3.1 RNA-seqデータ(FASTQファイル) (last modified 2014/04/15) [トップページへ](#)
- 書籍 |トランскриプトーム解析 | 2.3.2 リファレンス配列 (last modified 2014/04/16)

# NGS速習コース全貌のおさらい

## バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ) | 速習コース NEW

2014年9月にJST-NBDCと東大農アグリバイオ主催で「バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ)速習コース」が開催されます。主催機関のサイト上で情報提供したほうがいいだろうということで、受講者が各自でインストールするソフトウェアや、イメージファイルのダウンロードなど準備していただく計算機環境の情報などを示します。

バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ)関連:

- ・NBDCの速習コース案内サイト(速習コース主催機関)
- ・HPCIの速習コース受講申込受付サイト(速習コース共催機関)
- ・カリキュラムを策定したNBDC運営委員会人材育成分科会
- ・「NBDCで実施した調査」のバイオインフォマティクス人材育成のためのカリキュラム
  - 「バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ)」のPDF ([generation-sequencer.pdf](#))
  - 「カリキュラムで習得できる技能」のPDF ([learning-skills.pdf](#))
  - 「カリキュラム フロー図」のPDF ([flow-diagram.pdf](#))

習得できる技能は、「速習」と「速習以外」を含めたもの

計算機環境構築(Linux系):

8月初旬をめどにBio-Linuxに

### バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ)で習得できる技能

1. コンピュータリテラシーとサーバ設計  
プログラミング、ウェブ開発、並列計算ができる
2. 配列インフォマティクス  
データベースの理解や検索などの基礎的な配列比較解析ができる
3. データ解析基礎  
Rを利用した統計解析や基礎的な塩基配列解析ができる
4. 次世代シーケンサ

#### 要素・基礎技術

・アセンブル、マッピング

#### 個別解析技術

・RNA-seq  
・転写組成解析

#### 応用解析技術

・比較ゲノム解析  
複数種のゲノムの比較解析ができる

# NGS速習コース全貌のおさらい

## バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ) | 速習コース NEW

2014年9月にJST-NBDCと東大農アグリバイオ主催で「バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ)速習コース」が開催されます。主催機関のサイト上で情報提供したほうがいいだろうということで、受講者が各自でインストールするソフトウェアや、イメージファイルのダウンロードなど準備していただく計算機環境の情報などを示します。

バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ)関連:

- ・NBDCの速習コース案内サイト(速習コース主催機関)
- ・HPCIの速習コース受講申込受付サイト(速習コース共催機関)
- ・カリキュラムを策定したNBDC運営委員会人材育成分科会
- ・「NBDCで実施した調査」のバイオインフォマティクス人材育成のためのカリキュラム
  - 「バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ)」のPDF ([generation-sequencer.pdf](#))
  - 「カリキュラムで習得できる技能」のPDF ([learning-skills.pdf](#))
  - 「カリキュラム フロー図」のPDF ([flow-diagram.pdf](#))

「速習」と「速習以外」の個別の項目での習得技術が  
おおまかに書かれている



計算機環境構築(Linux系):

8月初旬をめどにBio-Linuxにプリインストールされていないプログラムなども含めたイメージファイルの提供を行う予定です。

# NGS速習コース全貌のおさらい

## バイオインフォマティクス人材育成カリキュラム(次世代シークエンサ)

本カリキュラムは、次世代シークエンサデータを扱うにあたり最低限必要とされる知識・技術を2週間程度で身につけることを想定した「速習」と、時間をかけて習得することを想定した「速習以外」に分かれています。

「速習」と「速習以外」の個別の項目での習得技術が  
おおまかに書かれている

### 【速習】

大項目	日数		No.	項目	習得技術	初級	講義
1. コンピュータリテラシーとサーバー設計	4日	2日	1-1	OS, ハード構成	・コンピュータの基本の理解		
			1-2	ネットワーク基礎	・インターネット、セキュリティの基本の理解	初級	講義
			1-3	UNIX I	・UNIXの基礎の理解 ・Linux導入	中級	実習
	2日	1-4	スクリプト言語	・Perl ・シェルスクリプト		中級	実習
2. 配列インフォマティクス	1日		2-1	配列解析基礎	・配列、ゲノムデータ記述のフォーマット、アラインメント(DP)、データベース検索(BLAST, BLAT)等の基礎的な配列比較解析の原理と実習	初級	実習
			2-2	バイオ系データベース概論	・基本的な各種バイオ系データベースの理解、統合DBの利用法	初級	実習
3. データ解析基礎	2日		3-1	R 基礎1	・R言語の基礎(インストールから利用まで)	初級	実習
			3-2	R 基礎2	・ファイルの読み込み、行列演算の基本	初級	実習
			3-3	R 各種パッケージ	・Rの各種パッケージのインストール法と代表的なパッケージの利用法	中級	実習
			3-4	R bioconductor I	・bioconductorの利用法	中級	実習
			3-5	R bioconductor II	・FASTA and FASTQ形式ファイルの読み込み。ファイル形式の変換(FASTQ → FASTA)、クオリティチェック、リード配列長分布、フィルタリングやトリミング、GC含量計算など	中級	実習

# Contents (カリキュラム記載事項)

- 3-1. R 基礎1、2014/09/08 10:30–12:00、初級、実習
  - R言語の基礎(インストールから利用まで)
- 3-2. R 基礎2、2014/09/08 13:15–14:45、初級、実習
  - ファイルの読み込み、行列演算の基本
- 3-3. R 各種パッケージ、2014/09/08 15:00–18:15、中級、実習
  - Rの各種パッケージのインストール法と代表的なパッケージの利用法
- 3-4. R bioconductor I、2014/09/09 10:30–14:45、中級、実習
  - Bioconductorの利用法
- 3-5. R bioconductor II、2014/09/09 15:00–18:15、中級、実習
  - FASTA and FASTQ形式ファイルの読み込み、ファイル形式の変換(FASTQ → FASTA)、クオリティチェック、リード配列長分布、フィルタリングやトリミング、GC含量計算など。

# Contents (「3. データ解析基礎」全体)

- 3-1. R 基礎1、2014/09/08 10:30–12:00、初級、実習
  - Rおよびパッケージのインストール、インストール後の確認
  - 基本的な利用法。`log`, `exp`, `mean`, `median`, `sort`, `length`関数。`?`関数名など。
- 3-2. R 基礎2、2014/09/08 13:15–14:45、初級、実習
  - (Rで)塩基配列解析の基本的な利用法(翻訳配列の取得を例に)
  - 行列形式ファイルの解析基礎(アノテーションファイルを例に)
- 3-3. R 各種パッケージ、2014/09/08 15:00–18:15、中級、実習
  - Rの各種パッケージのインストール法と代表的なパッケージBiostringsの利用法
- 3-4. R Bioconductor I、2014/09/09 10:30–14:45、中級、実習
  - データの型、バージョンの違い、警告メッセージとその対処法、Bioconductorサイト概観。
  - `setwd`, `translate`, `rm`, `ls`, `objects`, `sessionInfo`, `readDNAStringSet`関数、プロモータ配列取得。
- 3-5. R Bioconductor II、2014/09/09 15:00–18:15、中級、実習
  - FASTA/FASTQ形式ファイルの操作。ファイル形式の変換、クオリティチェック、フィルタリング、クラスオブジェクト。`alphabetFrequency`, `apply`, `rowSums`, `colSums`, `sample`, `set.seed`関数。

# 「3. データ解析基礎」での目標

- Rの基本的な利用法を知る
- (Rで)塩基配列解析を使いこなす
  - できることの全体像を知る
  - 基本はコピペで実行(ファイル名や必要最小限のパラメータの変更のみ)
  - 原因既知状態でのエラーを沢山経験し、実データ解析時の複合的なエラーを着実に解決する基本的なノウハウを身につける
    - コードの中身をある程度知っておく
    - R本体やパッケージのバージョンの違い(関数名やデフォルトオプションの変更)
    - WindowsとMacintoshの違い
  - 得られた結果の合理的な解釈ができるようになる



# Contents

- 3-1. R 基礎1、2014/09/08 10:30–12:00、初級、実習
  - Rおよびパッケージのインストール、インストール後の確認
    - 参考図書
    - パッケージが正しくインストールされているかどうかの確認
    - エラーメッセージとその対処法
  - 基本的な利用法
    - 四則演算、スクリプトファイルの作成とコピペ、改行の有無に注意
    - コメント行、上下左右の矢印キーを有効利用、エラーメッセージ
    - 関数の利用マニュアル、ベクトル計算、オプションの変更
    - 条件判定、論理値ベクトル
    - ベクトル中の任意の要素を抽出(subsetting)
    - ベクトルの要素数、組合せ(ソート後に最初の3要素を抽出)の基本



約 14,300,000 件 (0.22 秒)

## 参考ウェブページ

[\(Rで\)塩基配列解析](http://www.iu.a.u-tokyo.ac.jp/~kadota/r_seq.htm)[www.iu.a.u-tokyo.ac.jp](http://www.iu.a.u-tokyo.ac.jp/~kadota/r_seq.htm)

このページは、次世代シ  
短い塩基配列(short re  
あり、特にアグリバイオ

[R+Bioconductor](#)[togotv.dbcls.jp/201209](#)

2012/09/25 - 本日の統  
会: AJACSみちのく2か  
員による「R+Biocondu

[Learning R - The](#)  
[cat.hackingisbelieving.](#)

著者: Itoshi NIKAIDO

NGS解析: 1限 Rの基礎  
「Rと Bioconductorを使  
は二階堂 ...

[RNA-seq analys](#)[cat.hackingisbelieving.](#)

**(Rで)塩基配列解析**  
 ~NGS、RNA-seq、ゲノム、トランскриプトーム、正規化、発現変動、統計、モデル、バイオインフォマティクス~  
 (last modified 2014/07/31, since 2010)

---

#### What's new?

- このウェブページはフリーソフト R と必要なパッケージをインストール済みである前提で記述しています。初心者は、1. [Rのインストールと起動](#)および2. [基本的な利用法](#)で自習してください。(2014/07/21) NEW
- 2014年10月04日にHPCIワークショップ「医療とビッグデータ解析」(9:00-9:20)に引き続いて [中級者向けバイオインフォマティクス入門講習会](#) @仙台国際センター(10:50-12:20)で話します。興味ある方はどうぞ。(2014/07/23) NEW
- 門田幸二著 [シリーズ Useful R 第7巻トランскриプトーム解析](#)刊行(共立出版)
- 2014年9月1日～12日に「[バイオインフォマティクス人材育成カリキュラム\(次世代シークエンサ\)速習コース](#)」を開催します。[受講申込](#)は6/24夕方に締め切りました。TA申込枠はあと数名です。(2014/07/21) NEW
- [参考資料\(講義、講習会、本など\)](#)の項目を追加しました。(2014/07/30) NEW

---

- [はじめに](#) (last modified 2014/01/30)
- [参考資料\(講義、講習会、本など\)](#) (last modified 2014/07/30) NEW
- [過去のお知らせ](#) (last modified 2014/07/31) NEW
- [Rのインストールと起動](#) (last modified 2014/07/07) NEW
- [基本的な利用法](#) (last modified 2014/07/20) NEW

[トップページへ](#)

Windows版を例としてR本体および各種パッケージのインストール手順やエラーへの対処法などを記載しています。動作確認はR ver. 3.1.0とBioconductor ver. 2.14で行っています。OSが違えど、基本的な手順は同じ。

# (Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランскriプトーム、正規化、発現変動、統計、モデル、バイ

(last modified 2014/07/31, since 201

## What's new?

- このウェブページはフリーソフトRとは、1. [Rのインストールと起動](#)およ
- 2014年10月04日 Rワークショ
- フマティクス入門講習会@仙台国
- 門田幸二著シリ
- 2014年9月1日～12日に「バイオイ
- します。受講申込は6/24夕方に締
- 参考資料(講義、講習会、本など)

- はじめに (last modified 2014/01/30)
- 参考資料(講義、講習会、本など)
- 過去のお知らせ (last modified 201
- Rのインストールと起動 (last modif
- 基本的な利用法 (last modified 20

## Rのインストールと起動 NEW

基本的には[こちら](#)または[こちら](#)をご覧ください。

よく分からぬ人でWindowsユーザーの方は以下を参考にしてください。2014年7月31日にアップデートしたWindows用のインストール手順は[こちら](#)。2014年5月14日にアップデートしたMac版のインストール手順[こちら](#)(by 孫建強氏)もあります。注意点は、「Mac OS Xのバージョンに関わらず R-3.1.0-snowleopard.pkg をインストールしたほうがよい」です。

### 1. Windows release版のインストールの場合:

②

#### Rのインストーラを「実行」

- 聞かれるがままに「次へ」などを押してとにかくインストールを完了させる
- Windows Vista**の人は(パッケージのインストール中に書き込み権限に関するエラーが出るのを避けるために)「コントロールパネル」→「ユーザーアカウント」→「ユーザーアカウント制御の有効化または無効化」で、「ユーザーアカウント制御(UAC)を使ってコンピュータの保護に役立たせる」のチェックをあらかじめ外しておくことを強くお勧めします。
- インストールが無事完了したら、デスクトップに出現する「R3.X.Y(32 bit)の場合; XやY中の数値はバージョンによって異なります」または「R x64 3.X.Y(64 bitの場合)」アイコンをダブルクリックして起動
- 以下を、「Rコンソール画面上」でコピー&ペーストする。10GB程度のディスク容量を要しますが一番お手軽です。(どこからダウンロードするか?と聞かれるので、その場合は自分のいる場所から近いサイトを指定)

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

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

http://www.iu.a.u-tokyo.ac.jp/~kadota/r\_seq.htm (Rで)塩基配列解析

# (Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランскриプトーム、正規化、発現変動、統計、モデル、バイオインフォマティクス～  
(last modified 2014/07/31, since 2014/07/31)

**What's new?**

- このウェブページはフリーソフト R と  
は、1. [Rのインストールと起動](#)およ  
り
- 2014年10月04日に [HPCワークショ  
ップフォマティクス入門講習会@仙台国  
立大](#) が開催されました。
- 門田幸二著 [シリーズ Useful R 第1  
巻](#) が発売されました。
- 2014年9月1日～12日に「[バイオイ  
ンフォマティクス入門講習会](#)」が開催さ  
れます。[受講申込](#)は6/24夕方に締め  
切られます。
- [参考資料\(講義、講習会、本など\)](#)
- [はじめに](#) (last modified 2014/01/30)
- [参考資料\(講義、講習会、本など\)](#)
- [過去のお知らせ](#) (last modified 2014/07/31)
- [Rのインストールと起動](#) (last modified 2014/07/31)
- [基本的な利用法](#) (last modified 2014/07/31)

**Rのインストールと起動 NEW**

基本的には[こちら](#)または[こちら](#)をご覧ください。  
よく分からぬ人でWindowsユーザーの方は以下を参考にしてください。2014年7月31日にアップデートしたWindows用のインストール手順は[こちら](#)、2014年5月14日にアップデートしたMac版のインストール手順[こちら](#)(by 孫建強氏)もあります。注意点は、「Mac OS Xのバージョンに問わらず R-3.1.0-snowleopard.pkg をインストールしたほうがよい」です。

**1. Windows release版のインストールの場合:**

1. [Rのインストーラ](#)を「実行」
2. 聞かれるがままに「次へ」などを押してとにかく進む
3. **Windows Vista**の人は(パッケージのインストール時に)「コントロールパネル」→「ユーザーアカウント」→「ユーザーアカウント制御(UAC)を使ってコンピュータを操作する」を強くお勧めします。
4. インストールが無事完了したら、デスクトップに出現する「R3.X.Y(32 bit)の場合; XやY中の数値はバージョンによって異なります」または「R x64 3.X.Y(64 bit)の場合」アイコンをダブルクリックして起動
5. 以下の「Rコンソール画面上」でコピー&ペーストする。10GB程度のディスク容量を要しますが一番お手軽です。(どこからダウンロードするか?と聞かれるので、その場合は自分のいる場所から近いサイトを指定)

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

**より詳細なインストール手順を示したPDFはこちら。NGS速習コース受講生はこちらを参考にして正しくインストール済みという前提です。**

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

http://www.iu.a.u-tokyo.ac.jp/~kadota/r\_seq.htm (Rで)塩基配列解析

# (Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランскриプトーム、正規化、発現変動、統計、モデル、バイオインフォマティクス～  
 (last modified 2014/07/31, since 2014/07/31)

**What's new?**

- このウェブページはフリーソフト R と関連して、1. [Rのインストールと起動](#)および 2. [Rによる塩基配列解析](#)を紹介するものです。
- 2014年10月04日に[HPCワークショップ「バイオインフォマティクス入門講習会@仙台国際会議場](#)が開催されました。
- 門田幸二著[「Useful R 第1版」](#)が発売されました。
- 2014年9月1日～12日に「[バイオインフォマティクス入門講習会](#)」が開催されます。[受講申込](#)は6/24夕方に締め切られます。
- [参考資料\(講義、講習会、本など\)](#)
- [はじめに](#) (last modified 2014/01/30)
- [参考資料\(講義、講習会、本など\)](#)
- [過去のお知らせ](#) (last modified 2014/07/31)
- [Rのインストールと起動](#) (last modified 2014/07/31)
- [基本的な利用法](#) (last modified 2014/07/31)

## Rのインストールと起動 NEW

基本的には[こちら](#)または[こちら](#)をご覧ください。よく分からない人でWindowsユーザーの方は以下を参考にしてください。2014年7月31日にアップデートしたWindows用のインストール手順は[こちら](#)。2014年5月14日にアップデートしたMac版のインストール手順[こちら](#)(by 孫建強氏)もあります。注意点は、「Mac OS X のバージョンに関わらず R-3.1.0-snowleopard.pkg をインストールしたほうがよい」です。

**1. Windows release版のインストールの場合:**

1. [Rのインストーラ](#)を「実行」
2. 聞かれるがままに「次へ」などを押していく
3. **Windows Vista**の人は(パッケージのインストーラーに)「コントロールパネル」「ユーザー」「ユーザーアカウント制御(UAC)」を使うことを強くお勧めします。
4. インストールが無事完了したら、デスクトップに「R」というアイコンが表示される(によって異なります)または「R x64 3.1.0」
5. 以下のコードを、「Rコンソール画面上」でコピーして貼り付けてください。(どこからダウンロードするか?)

インストールができているつもりでも、実際にできていなかったという事例が散見されます。パッケージ名のスペルミスもよく見受けられます。Macintoshのヒトもこちらを参考にして、いくつかのパッケージについて適切にインストールされているか確認しておきましょう。

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

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

# (Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランскриプトーム、正規化、発現変動、統  
(last modified 2014/07/31, since 2010)

## What's new?

- このウェブページはフリーソフトRと必要なパッケージをインストール済みは、1. [Rのインストールと起動](#)および2. [基本的な利用法](#)で自習してください。
  - 2014年10月04日にHPCIワークショップ「医療とビッグデータ解析」(9:00-12:00) フォマティクス入門講習会@仙台国際センター(10:50-12:20)で話します。
  - 門田幸二著シリーズ Useful R 第7巻トランскриプトーム解析刊行(共著)
  - 2014年9月1日～12日に「バイオインフォマティクス人材育成カリキュラム」します。受講申込は6/24夕方に締め切りました。TA申込枠はあと数名あります。
  - 参考資料(講義、講習会、本など)の項目を追加しました。(2014/07/30)
- 
- [はじめに](#) (last modified 2014/01/30)
  - [参考資料\(講義、講習会、本など\)](#) (last modified 2014/07/30) NEW
  - [過去のお知らせ](#) (last modified 2014/07/31) NEW
  - [Rのインストールと起動](#) (last modified 2014/07/07) NEW
  - [基本的な利用法](#) (last modified 2014/07/20) NEW

定期的なバージョンアップや予め殆ど全てのパッケージのインストールを推奨するポリシーも記載。ウェブページでは書きづらい統計的なものの考え方、数式が苦手な人向けに重みつき平均値を算出する手順を詳述し、用いるパラメータの常識・非常識などを丁寧に解説。



金 明哲 編・門田 幸二 著

シリーズ名 シリーズ Useful R 全10巻 [7] 卷  
ISBN 978-4-320-12370-0  
判型 B5  
ページ数 240ページ  
発売予定 2014年04月10日  
本体価格 3,600円

## 新刊

今日では、インターネット上の検索エンジンでキーワード検索すれば、個別の情報は簡単に得られる。しかし、本書のメインターゲットである生命科学分野の実験系研究者やこれからバイオインフォマティクスを学ぼうとする大学院生にとって、特に統計関連の記述は難解であろう。巷に溢れている統計関連書籍の記述内容もまた、意味不明だという声をよく聞く。

本書は、トランスクリプトーム解析を行うための一連のスクリプト集である著者の2つのウェブページ「(Rで)マイクロアレイデータ解析;  
<http://www.iu.a.u-tokyo.ac.jp/~kadota/r.html>」および「(Rで)塩基配列解析;  
[http://www.iu.a.u-tokyo.ac.jp/~kadota/r\\_seq.html](http://www.iu.a.u-tokyo.ac.jp/~kadota/r_seq.html)」を体系的にまとめた初の書籍である。

まず手元にある実際のデータやその解析結果を示し、解釈の仕方を述べてから一般論に導く記述形式を採用している。主な目的は、実データの解析結果を徹底的に眺めることで、統計的な感覚や数式の感覚を身につけることである。一般に、書籍中に記載されているRパッケージや関数は、パッケージ自体がなくなってしまっていたりオプションが変更されるなど比較的早期に陳腐化していく。本書は、もちろん執筆時点で最新の解析手順やRの関数を利用しているが、それらの賞味期限は短いことが予想される。そのため、最新の利用手順は2つのウェブページを参考にされたい。本書は、トランスクリプトーム解析のための二大技術であるマイクロアレイとRNA-seqをRで自在に解析するための基本的な考え方や注意点を体系的にまとめたものである。

# その他参考書

羊土社HPより



## 次世代シーケンス解析スタンダード

NGSのポテンシャルを活かしきるWET&DRY

二階堂 愛／編

定価 5,500円+税 2014年08月 発行 B5判 404ページ  
ISBN 978-4-7581-0191-2

[SHARE] ショート シェア B! 8+1

本書概要 目次詳細 立ち読み 掲載広告・資料請求

エピゲノム研究はもとより、医療現場から非モデル生物、生物資源まで各分野の「NGSの現場」が詰まった1冊。コツや条件検討方法などWET実験のポイントが、データ解析の具体的なコマンド例が、わかる！

内容見本



PDFダウンロード

序

索引

目次詳細

次世代  
シーケンス  
解析

Standard next-generation sequencing  
Next Generation Sequencing



羊土社  
YODOBASHI

二階堂愛先生編集！現場  
目線で徹底的に丁寧な解説。



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

&gt; library(Biostrings)

要求されたパッケージ BiocGenerics をロード中です  
要求されたパッケージ parallel をロード中です

次のパッケージを付け加えます: 'BiocGenerics'

以下のオブジェクトはマスクされています (from 'package:parallel') :

```
clusterApply, clusterApplyLB,
clusterExport, clusterMap, parLapplyLB, parRapply, parSapply
```

以下のオブジェクトはマスクされています (from 'package:IRanges')

xtabs

以下のオブジェクトはマスクされています (from 'package:XVector')

```
anyDuplicated, append, as.dat,
colnames, do.call, duplicated,
intersect, is.unsorted, lapply,
paste, pmax, pmax.int, pmin, p,
Reduce, rep.int, rownames, sapply,
union, unique, unlist
```

要求されたパッケージ IRanges をロード中です  
要求されたパッケージ XVector をロード中です

&gt;

一連のインストール作業が終了すると、  
Biostrings, ShortRead, TCC,  
BSgenome.Athaliana.TAIR.TAIR9などの各  
種パッケージが利用可能になっているはず。  
左記のように打ち込んでエラーが出ていな  
ければオッケー。

R GUI (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

STOP

R Console

要求されたパッケージ IRanges をロード中です  
要求されたパッケージ XVector をロード中です

```
> library(ShortRead)
要求されたパッケージ BiocParallel をロード中です
要求されたパッケージ Rsamtools をロード中です
要求されたパッケージ GenomicRanges をロード中です
要求されたパッケージ GenomeInfoDb をロード中です
要求されたパッケージ GenomicAlignments をロード中です
要求されたパッケージ BSgenome をロード中です
```

> |

&gt; library(TCC)

要求されたパッケージ DESeq をロード中です  
 要求されたパッケージ Biobase をロード中です  
 Welcome to Bioconductor

Vignettes contain introductory material; view  
 'browseVignettes()'. To cite Bioconductor, see  
 'citation("Biobase")', and for packages  
 'citation("pkgname")'.

要求されたパッケージ locfit をロード中です  
 locfit 1.5-9.1 2013-03-22

次のパッケージを付け加えます: 'locfit'

以下のオブジェクトはマスクされています (from 'package:Geno  
 left, right

要求されたパッケージ lattice をロード中です  
 Welcome to 'DESeq'. For improved performance,  
 and functionality, please consider migrating

要求されたパッケージ DESeq2 をロード中です

要求されたパッケージ Rcpp をロード中です

要求されたパッケージ RcppArmadillo をロード中です

次のパッケージを付け加えます: 'DESeq2'

以下のオブジェクトはマスクされています (from 'package:DESe  
 estimateSizeFactorsForMatrix, getVarianceStabilizingTransfo  
 plotPCA, varianceStabilizingTransformation

一連のインストール作業が終了すると、  
 Biostrings, ShortRead, TCC,  
 BSgenome.Athaliana.TAIR.TAIR9などの各  
 種パッケージが利用可能になっているはず。  
 左記のように打ち込んでエラーが出ていな  
 ければオッケー。

R Console

要求されたパッケージ ba

次のパッケージを付け加えます: 'bayseq'

以下のオブジェクトはマスクされています (from 'package:ShortRead') \$

rbind

以下のオブジェクトはマスクされています (from 'package:GenomicRanges\$

rbind

以下のオブジェクトはマスクされています (from 'package:IRanges') :

rbind

以下のオブジェクトはマスクされています (from 'package:BiocGenerics\$

rbind

以下のオブジェクトはマスクされています (from 'package:base') :

rbind

要求されたパッケージ ROC をロード中です

次のパッケージを付け加えます: 'TCC'

以下のオブジェクトはマスクされています (from 'package:edgeR') :

calcNormFactors

> |

```
> library(BSgenome.Athaliana.TAIR.TAIR9)
要求されたパッケージ BSgenome をロード中です
要求されたパッケージ BiocGenerics をロード中です
要求されたパッケージ parallel をロード中です
```

次のパッケージを付け加えます: 'BiocGenerics'

以下のオブジェクトはマスクされています (from 'package:parallel') :

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

以下のオブジェクトはマスクされています (from 'package:stats') :

```
xtabs
```

以下のオブジェクトはマスクされています (from 'package:base') :

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

要求されたパッケージ IRanges をロード中です
要求されたパッケージ GenomicRanges をロード中です
要求されたパッケージ GenomeInfoDb をロード中です
要求されたパッケージ Biostrings をロード中です
要求されたパッケージ XVector をロード中です

```
> library(BSgenome.Athaliana.TAIR.TAIR9)
```

```
> |
```

一連のインストール作業が終了すると、  
**Biostrings, ShortRead, TCC, BSgenome.Athaliana.TAIR.TAIR9**などの各種パッケージが利用可能になっているはず。左記のように打ち込んでエラーが出ていなければオッケー。

一度、library関数を用いて読み込んだパッケージをもう一度読み込むと、表示される文章がなくなります。しかしこれもエラーなく読み込んでいるので問題なしです

R Gui (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 Vignettes

R Console

```
rbind
以下のオブジェクトはマスクされています (from 'package:base') :

rbind

要求されたパッケージ ROC をロード中です

次のパッケージを付け加えます: 'TCC'

以下のオブジェクトはマスクされています (from 'package:edgeR') :

calcNormFactors

> library(ShortRead)
> library(TCC)
> library(Biostrings)
> |
```

# エラー遭遇例とその対処法1

R R Console

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

次のパッケージを付け加えます: 'BiocGenerics'

以下のオブジェクトはマスクされています (from 'package:parallel') :

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

以下のオブジェクトはマスクされています (from 'package:stats') :

xtabs

以下のオブジェクトはマスクされています (from 'package:base') :

anyDuplicated, append, as.data.frame, as.vector, cbind,
colnames, do.call, duplicated, eval, evalq, Filter, Find, get,
intersect, is.unsorted, lapply, Map, mapply, match, mget, order,
paste, pmax, pmax.int, pm
Reduce, rep.int, rownames
union, unique, unlist

要求されたパッケージ Biobase をロード中です
Welcome to Bioconductor

Vignettes contain introductions to various analysis
'browseVignettes()'. To cite these packages:
'citation("Biobase")', and many more.

要求されたパッケージ locfit をロード中です
locfit 1.5-9.1 2013-03-22
要求されたパッケージ lattice をロード中です
Welcome to 'DESeq'. For improved performance, usability and
functionality, please consider migrating to 'DESeq2'.
要求されたパッケージ DESeq2 をロード中です
要求されたパッケージ GenomicRanges をロード中です
要求されたパッケージ IRanges をロード中です
要求されたパッケージ GenomeInfoDb をロード中です
要求されたパッケージ Rcpp をロード中です
エラー: パッケージ 'RcppArmadillo' が 'DESeq2' によって要求されましたが、見つけられませんでした
>
```

ときどき必要なパッケージのインストールに失敗していて、任意のパッケージXXXの読み込みを行うlibrary(XXX)実行後にエラーが出てしまうことがあります。この例では、TCCパッケージが要求している「RcppArmadilloパッケージがないからダメ!」と文句を言っています。

R R Console

```
> library(TCC)
要求されたパッケージ DESeq2 をロード中です
エラー: パッケージ 'RcppArmadillo' が 'DESeq2' によって要求されましたが、見つけられませんでした
>
```

# エラー遭遇例とその対処法1

基本的な対処法は、文句を言われたパッケージのみインストールすることです。RcppArmadilloパッケージを個別にインストールするためのコマンドの基本形は以下のとおりです：

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

R Console

```
要求されたパッケージ DESeq2 をロード中です
エラー: パッケージ 'RcppArmadillo' が 'DESeq2' によって要求されました but it was not found
> source("http://www.bioconductor.org/biocLite.R")
Bioconductor version 2.14 (BiocInstaller 1.14.1), ?biocLite for help
> biocLite("RcppArmadillo")
BioC_mirror: http://bioconductor.org
Using Bioconductor version 2.14 (BiocInstaller 1.14.1), R version 3.1.0.
Installing package(s) 'RcppArmadillo'
URL 'http://cran.fhcrc.org/bin/windows/contrib/3.1/RcppArmadillo_0.4.200.0.zip' を試して
Content type 'application/zip' length 1551263 bytes (1.5 Mb)
開かれた URL
downloaded 1.5 Mb

package 'RcppArmadillo' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
  C:\Users\kadota\AppData\Local\Temp\RtmpgNGQWx\downloaded_packages
Old packages: 'AnthropMMD', 'bayesQR', 'Bclim', 'care', 'clogitL1', 'freestats',
  'geomorph', 'gtools', 'investr', 'jsonlite', 'markovchain', 'meta', 'multicon',
  'mvtnorm', 'NLP', 'openNLP', 'PBD', 'pdfetch', 'poisson.glm.mix', 'QCA3', 'Rbitcoin',
  'regRSM', 'Reol', 'rgbif', 'Rmpi', 'rsm', 'RTexureMetrics', 'RWebLogo', 'sda',
  'SEERaBomb', 'segmented', 'seqDesign', 'seqminer', 'sjPlot', 'spcr', 'st', 'yuima'
Update all/some/none? [a/s/n]: n
> |
```

Update all/some/none? [a/s/n]:  
と聞かれることもありますが基本  
はnでいいです。

R Console

```

Update all/some/none? [a/s/n]: n
> library(TCC)
要求されたパッケージ DESeq2 をロード中です
要求されたパッケージ RcppArmadillo をロード中です
次のパッケージを付け加えます: 'DESeq2'
以下のオブジェクトはマスクされています (from 'package:DESeq') :

estimateSizeFactorsForMatrix, getVarianceStabilizedData, plotPCA,
varianceStabilizingTransformation

要求されたパッケージ edgeR をロード中です
要求されたパッケージ limma をロード中です
次のパッケージを付け加えます: 'limma'
以下のオブジェクトはマスクされています (from 'package:DESeq2') :

plotMA

以下のオブジェクトはマスクされています (from 'package:DESeq') :

plotMA

以下のオブジェクトはマスクされています (from 'package:BiocGenerics') :

plotMA

要求されたパッケージ baySeq をロード中です
次のパッケージを付け加えます: 'baySeq'
以下のオブジェクトはマスクされています (from 'package:GenomicRanges') :

rbind

以下のオブジェクトはマスクされています (from 'package:IRanges') :

rbind

以下のオブジェクトはマスクされています (from 'package:BiocGenerics') :

rbind

以下のオブジェクトはマスクされています (from 'package:base') :

rbind

要求されたパッケージ ROC をロード中です
次のパッケージを付け加えます: 'TCC'
以下のオブジェクトはマスクされています (from 'package:edgeR') :

calcNormFactors

```

RcppArmadilloパッケージのインストール  
後に、もう一度library(TCC)とやって、エラーが出なくなることを確認しています。

R Console

```

以下のオブジェクトはマスクされています (from 'package:$

calcNormFactors

> library(TCC)
>

```

# エラー遭遇例とその対処法2

R R Console

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

次のパッケージを付け加えます: 'BiocGenerics'

以下のオブジェクトはマスクされています (from 'package:parallel') :

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

以下のオブジェクトはマスクされています (from 'package:stats') :

xtabs

以下のオブジェクトはマスクされています (from 'pac

anyDuplicated, append, as.data.frame,
colnames, do.call, duplicated, eval,
intersect, is.unsorted, lapply, Map,
paste, pmax, pmax.int, pmin, pmin.in
Reduce, rep.int, rownames, sapply, s
union, unique, unlist

要求されたパッケージ Biobase をロード中です
Welcome to Bioconductor

Vignettes contain introductory material; view with
'browseVignettes()'. To cite Bioconductor, see
'citation("Biobase")', and for packages 'citation("pkgname")'.

要求されたパッケージ locfit をロード中です
locfit 1.5-9.1 2013-03-22
要求されたパッケージ lattice をロード中です
Error in loadNamespace(j <- i[[1L]], c(lib.loc, .libPaths()), versionCheck =
'XML' という名前のパッケージはありません
エラー: パッケージ 'DESeq' をロードできませんでした
> |
```

次の例では、TCCパッケージが要求している「XMLパッケージがないからダメ!」と文句を言っています。  
重要な点は、エラーメッセージ中に「パッケージ'DESeq'をロードできませんでした」と書いてありますが、原因はDESeqではなくXMLパッケージがないためであるということを読み解くことです。

R R Console

```
> library(TCC)
要求されたパッケージ DESeq をロード中です
Error in loadNamespace(j <- i[[1L]], c(lib.loc, .libPaths()), versionCheck =
'XML' という名前のパッケージはありません
エラー: パッケージ 'DESeq' をロードできませんでした
> |
```

# エラー遭遇例とその対処法2

基本的な対処法は、文句を言われたパッケージのみインストールすることです。XMLパッケージを個別にインストールするためのコマンドの基本形は以下のとおりです：

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

```
R Console

> library(TCC)
要求されたパッケージ DESeq をロード中です
Error in loadNamespace(j <- i[[1L]], c(lib.loc, .libPaths()), versionCheck = $
  'XML' という名前のパッケージはありません
エラー： パッケージ 'DESeq' をロードできませんでした
> source("http://www.bioconductor.org/biocLite.R")
Bioconductor version 2.14 (BiocInstaller 1.14.1), ?biocLite for help
> biocLite("XML")
BioC_mirror: http://bioconductor.org
Using Bioconductor version 2.14 (BiocInstaller 1.14.1), R version 3.1.0.
Installing package(s) 'XML'
URL 'http://cran.fhcrc.org/bin/windows/contrib/3.1/XML_3.98-1.1.zip' を試し
Content type 'application/zip' length 4288694 bytes (4.1 Mb)
開かれた URL
downloaded 4.1 Mb

package 'XML' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
  C:\Users\kadota\AppData\Local\Temp\RtmpY3lPk1\downloaded_packages
Old packages: 'AnthropMMD', 'bayesQR', 'Bclim', 'care', 'clogitL1',
  'freestats', 'geomorph', 'gtools', 'investr', 'jsonlite', 'markovchain',
  'meta', 'multicon', 'mvtnorm', 'NLP', 'openNLP', 'PBD', 'pdfetch',
  'poisson.glm.mix', 'QCA3', 'Rbitcoin', 'regRSM', 'Reol', 'rgbif', 'Rmpi',
  'rsm', 'RTtextureMetrics', 'RWebLogo', 'sda', 'SEERaBomb', 'segmented',
  'seqDesign', 'seqminer', 'sjPlot', 'spcr', 'st', 'yuima'
Update all/some/none? [a/s/n]: n
> |
```

Update all/some/none? [a/s/n]:  
と聞かれることもありますが基本  
はnでいいです。

R Console

```

Update all/some-none? [a/s/n]: n
> library(TCC)
要求されたパッケージ DESeq をロード中です
  Welcome to 'DESeq'. For improved performance, usability and
  functionality, please consider migrating to 'DESeq2'.
要求されたパッケージ DESeq2 をロード中です
要求されたパッケージ GenomicRanges をロード中です
要求されたパッケージ IRanges をロード中です
要求されたパッケージ GenomeInfoDb をロード中です
要求されたパッケージ Rcpp をロード中です
要求されたパッケージ RcppArmadillo をロード中です

次のパッケージを付け加えます: 'DESeq2'

以下のオブジェクトはマスクされています (from 'package:DESeq') :

  estimateSizeFactorsForMatrix, getVarianceStabilizedData,
  plotPCA, varianceStabilizingTransformation

要求されたパッケージ edgeR をロード中です
要求されたパッケージ limma をロード中です

次のパッケージを付け加えます: 'limma'

以下のオブジェクトはマスクされています (from 'package:DESeq2') :

  plotMA

以下のオブジェクトはマスクされています (from 'package:DESeq') :

  plotMA

以下のオブジェクトはマスクされています (from 'package:BiocGenerics') :

  plotMA

要求されたパッケージ baySeq をロード中です
次のパッケージを付け加えます: 'baySeq'

以下のオブジェクトはマスクされています (from 'package:GenomicRanges') :

  rbind

以下のオブジェクトはマスクされています (from 'package:IRanges') :

  rbind

以下のオブジェクトはマスクされています (from 'package:BiocGenerics') :

  rbind

以下のオブジェクトはマスクされています (from 'package:base') :

  rbind

要求されたパッケージ ROC をロード中です
次のパッケージを付け加えます: 'TCC'

以下のオブジェクトはマスクされています (from 'package:edgeR') :

  calcNormFactors

> library(TCC)
> |

```

## 几法2

XMLパッケージのインストール後に、もう一度library(TCC)とやって、エラーが出なくなることを確認しています。

R Console

```

要求されたパッケージ ROC をロード 中です

次のパッケージを付け加えます: 'TCC'

以下のオブジェクトはマスクされています (from 'package:edgeR') :

  calcNormFactors

> library(TCC)
> |

```

# エラー遭遇例とその対処法3

シロイヌナズナ(*A. thaliana*)ゲノム配列情報を含む  
**BSgenome.Athaliana.TAIR.TAIR9**パッケージ読み込み時にエラーが出ている例です。対処法は以下の通りです。

R R Console

```
> library(BSgenome.Athaliana.TAIR.TAIR9)
以下にエラー library(BSgenome.Athaliana.TAIR.TAIR9) :
'BSgenome.Athaliana.TAIR.TAIR9' という名前のパッケージはありません
> source("http://www.bioconductor.org/biocLite.R")
Bioconductor version 2.14 (BiocInstaller 1.14.1), ?biocLite for help
> biocLite("BSgenome.Athaliana.TAIR.TAIR9")
BioC_mirror: http://bioconductor.org
Using Bioconductor version 2.14 (BiocInstaller 1.14.1), R version 3.1.0.
Installing package(s) 'BSgenome.Athaliana.TAIR.TAIR9'
URL 'http://bioconductor.org/packages/2.14/data/annotation/bin/windows/cont$'
Content type 'application/zip' length 37765491 bytes (36.0 Mb)
開かれた URL
downloaded 36.0 Mb
```

The downloaded binary packages are in

C:\Users\kadota\AppData\Local\Temp\RtmpaoaPGj\downloaded\_packages

Old packages: 'AnthropMMD', 'bayesQR', 'Bclim', 'care', 'clogitL1',
'freestats', 'geomorph', 'gtools', 'investr', 'jsonlite', 'markovchain',
'meta', 'multicon', 'mvtnorm', 'NLP', 'openNLP', 'PBD', 'pdfetch',
'poisson.glm.mix', 'QCA3', 'Rbitcoin', 'regRSM', 'Reol', 'rgbif',
'rsm', 'RTtextureMetrics', 'RWebLogo', 'sda', 'SEERaBomb', 'segmenter',
'seqDesign', 'seqminer', 'sjPlot', 'spcr', 'st', 'yuima'

Update all/some/none? [a/s/n]: n

>

Update all/some/none? [a/s/n]:  
と聞かれることもありますが基本  
はnでいいです。

# エラー遭遇例とその対処法3

R Console

```
> library(BSgenome.Athaliana.TAIR.TAIR9)
```

要求されたパッケージ BSgenome をロード中です

要求されたパッケージ BiocGenerics をロード中です

要求されたパッケージ parallel をロード中です

次のパッケージを付け加えます: 'BiocGenerics'

以下のオブジェクトはマスクされています (from 'package:parallel') :

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

以下のオブジェクトはマスクされています (from 'package:stats') :

```
xtabs
```

以下のオブジェクトはマスクされています (from 'package:base') :

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

要求されたパッケージ IRanges をロード中です

要求されたパッケージ GenomicRanges をロード中です

要求されたパッケージ GenomeInfoDb をロード中です

要求されたパッケージ Biostrings をロード中です

要求されたパッケージ XVector をロード中です

```
> library(BSgenome.Athaliana.TAIR.TAIR9)
```

```
> |
```

パッケージのインストール後に、もう一度 library(BSgenome.Athaliana.TAIR.TAIR9) とやって、エラーが出なくなることを確認しています。

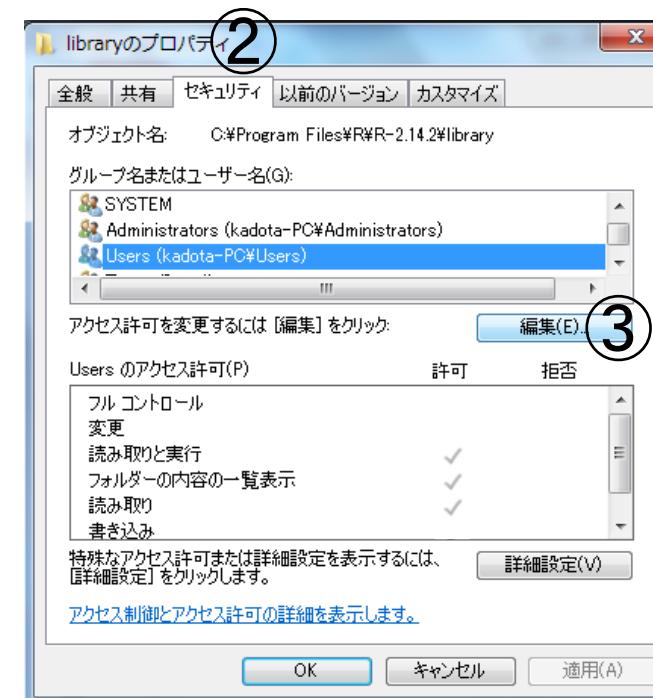
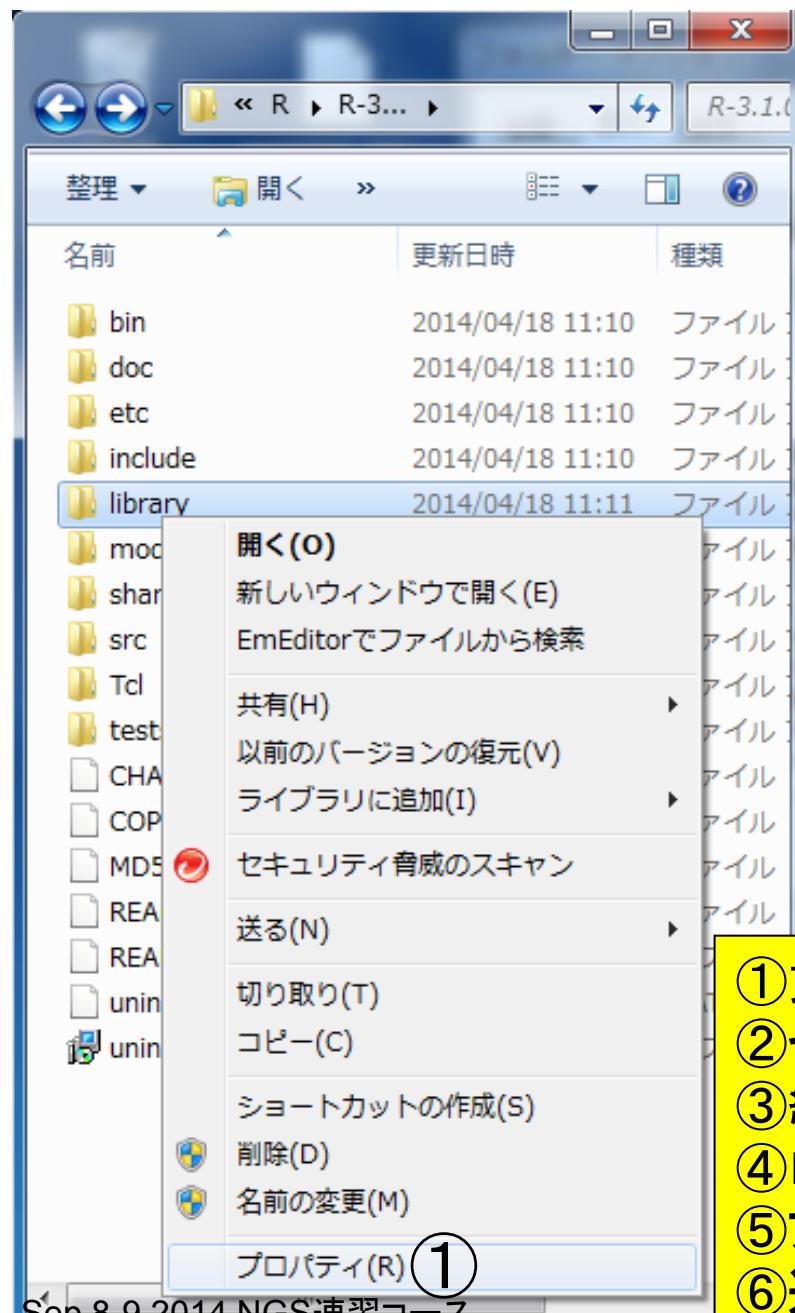
# 対処時の注意

パッケージを個別にインストールするためのコマンドの基本形は以下のとおりですが、**二重クオーテーションに注意!! 以下はXMLの左側がダメな例です**  
source("http://www.bioconductor.org/biocLite.R")  
biocLite("XML")

```
R R Console
```

```
> library(TCC)
要求されたパッケージ DESeq をロード中です
Error in loadNamespace(j <- i[[1L]], c(lib.loc, .libPaths()), versionCh$'XML' という名前のパッケージはありません
エラー: パッケージ 'DESeq' をロードできませんでした
> source("http://www.bioconductor.org/biocLite.R")
Bioconductor version 2.14 (BiocInstaller 1.14.1), ?biocLite for help
> biocLite("XML")
> library(TCC)
要求されたパッケージ DESeq をロード中です
Error in loadNamespace(j <- i[[1L]], c(lib.loc, .libPaths()), versionCh$'XML' という名前のパッケージはありません
エラー: パッケージ 'DESeq' をロードできませんでした
> |
```

「“C:/Program Files/R/R-3.1.0/library”に書き込み権限がない」的なエラーが出てインストールできなかつた人は、書き込み権限を取得してもう一度トライ



- ① 文句を言われたフォルダ上で右クリックでプロパティを選択
- ② セキュリティタブを選択
- ③ 編集をクリック
- ④ ログインしているユーザーを選択(フルコントロールにチェックなし)
- ⑤ フルコントロールにチェックを入れる
- ⑥ 適用をクリック

# Contents

- 3-1. R 基礎1、2014/09/08 10:30–12:00、初級、実習
  - Rおよびパッケージのインストール、インストール後の確認
    - 参考図書
    - パッケージが正しくインストールされているかどうかの確認
    - エラーメッセージとその対処法
  - 基本的な利用法
    - 四則演算、スクリプトファイルの作成とコピペ、改行の有無に注意
    - コメント行、上下左右の矢印キーを有効利用、エラーメッセージ
    - 関数の利用マニュアル、ベクトル計算、オプションの変更
    - 条件判定、論理値ベクトル
    - ベクトル中の任意の要素を抽出(subsetting)
    - ベクトルの要素数、組合せ(ソート後に最初の3要素を抽出)の基本



# (Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランскриプトーム、正規化、発現変動、統計、モデル、バイオインフォマティクス～  
(last modified 2014/07/31, since 2010)

## What's new?

- このウェブページはフリーソフト R と必要なパッケージをインストール済みである前提で記述しています。初心者は、1. [Rのインストールと起動](#)および2. [基本的な利用法](#)で自習してください。(2014/07/21) **NEW**
- 2014年10月04日に [HPCワークショップ「医療データ解析」](#) (9:00-9:20) に引き続いで [中級者向けバイオインフォマティクス入門講習会@仙台国際センター](#) (10:50-12:20) で話します。興味ある方はどうぞ。(2014/07/23) **NEW**
- 門田幸二著 [シリーズ Useful R 第7巻トランскриプトーム解析](#)刊行(共立出版)
- 2014年9月1日～12日に「[バイオインフォマティクス人材育成カリキュラム\(次世代シークエンサ\)速習コース](#)」を開催します。[受講申込](#)は6/24夕方に締め切りました。TA申込枠はあと数名です。(2014/07/21) **NEW**
- [参考資料\(講義、講習会、本など\)](#)の項目を追加しました。(2014/07/30) **NEW**

- [はじめに](#) (last update: 2014/07/30)
- [参考資料\(講義\)](#)
- [過去のお知らせ](#)
- [Rのインストール](#)
- [基本的な利用法](#)

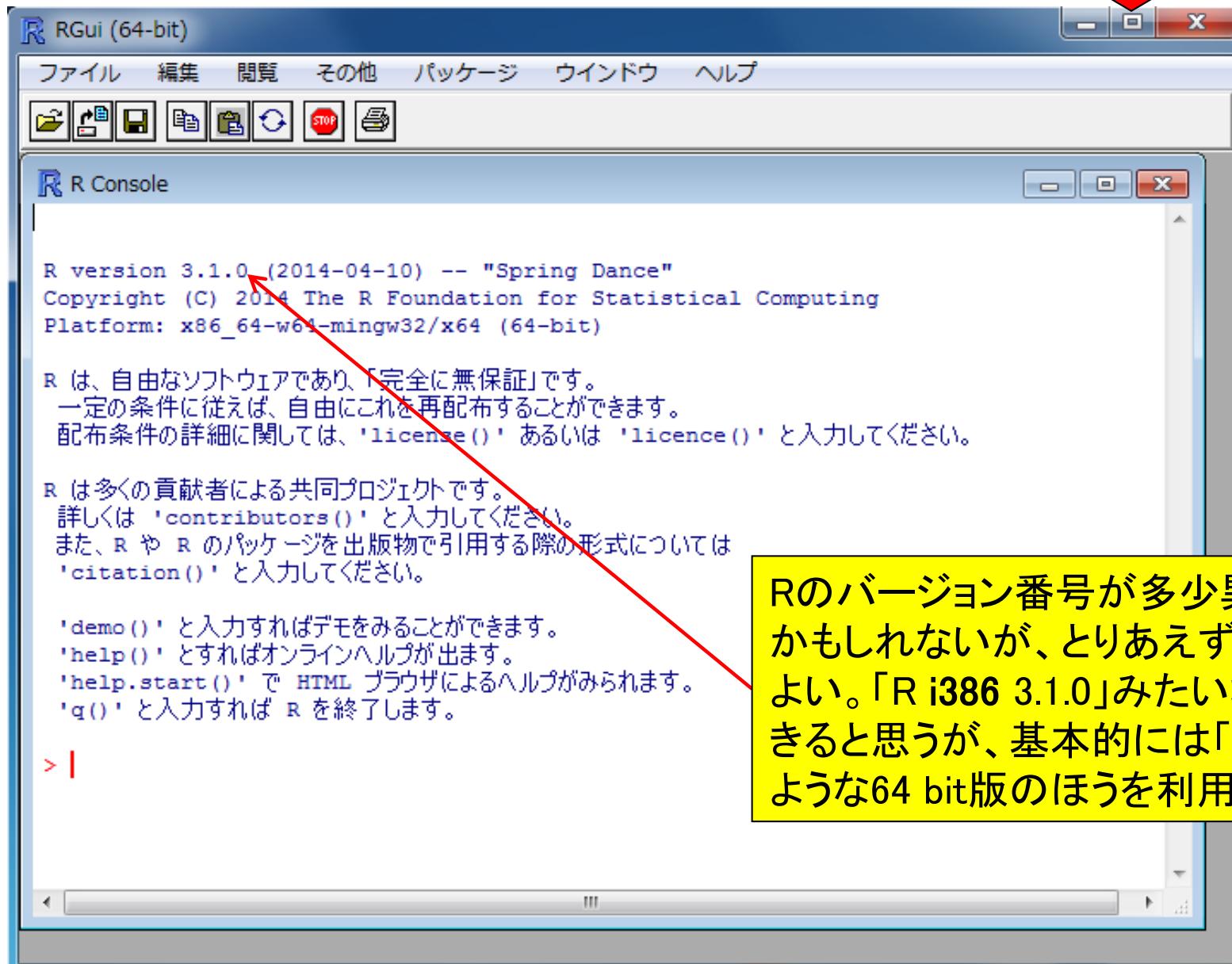
## 基本的な利用法 **NEW**

[Rのインストールと起動](#)を参考にして必要なパッケージのインストールが完了済みのヒトを対象として、[R Gui](#)画面や作業ディレクトリの変更、このウェブページの基本的な利用法を簡単に解説した[PDFファイル](#)を作成しました。2014年10月4日の[中級者向けバイオインフォマティクス入門講習会@東北大学](#) (10:50-12:20) 受講希望者は、この[PDFファイル](#)および[シリーズ Useful R 第7巻トランскриプトーム解析](#)を参考にして、このウェブページの基本的な利用法をマスターしておいてください。

# Rの起動

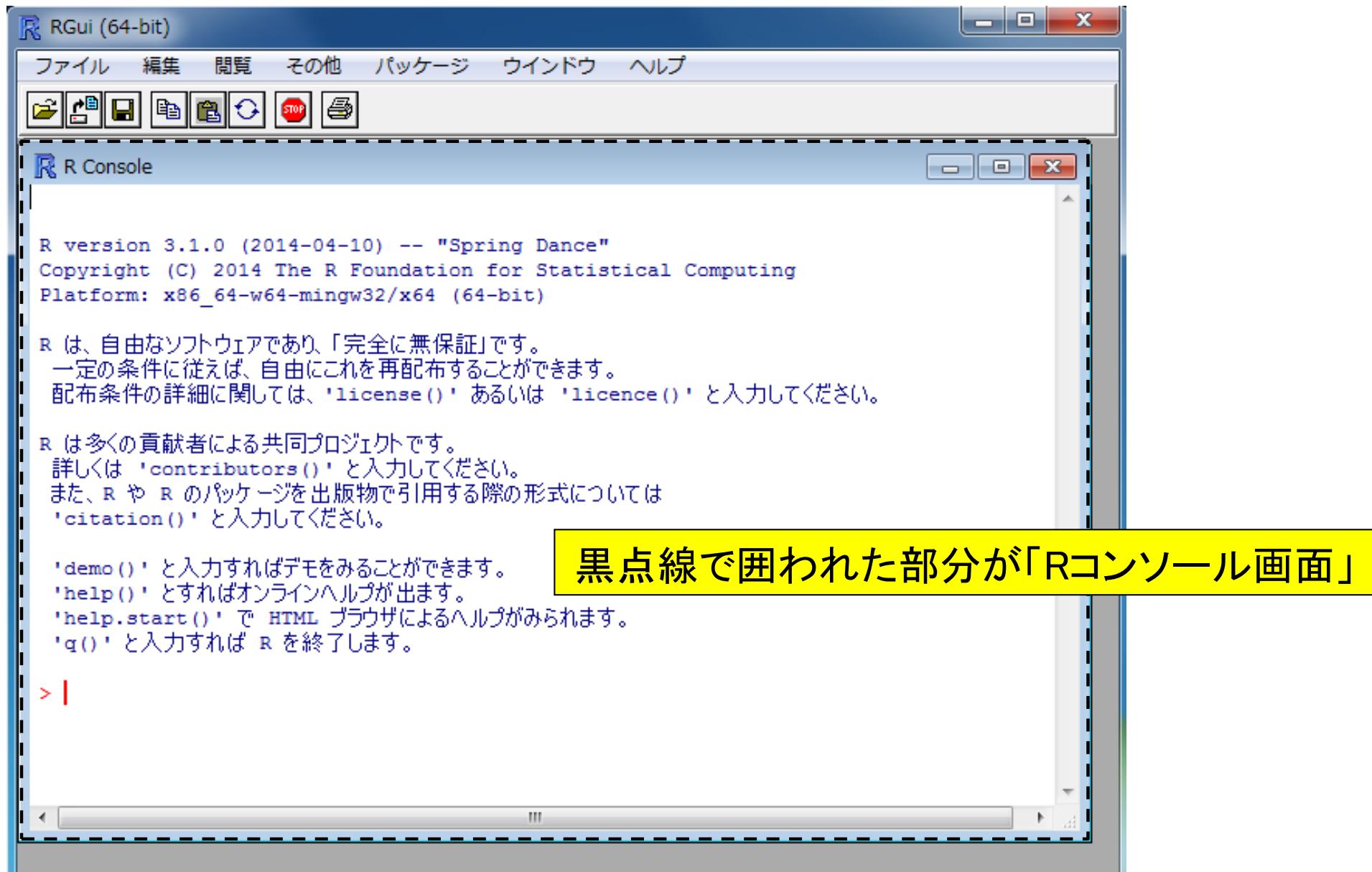


起動直後は画面いっぱいに開くので…



Rのバージョン番号が多少異なるヒトもいる  
かもしれないが、とりあえずは気にしないで  
よい。「R i386 3.1.0」みたいなアイコンもで  
きると思うが、基本的には「R x64 3.1.0」の  
ような64 bit版のほうを利用すべし。

# Rの画面説明



# 基本的な利用法

R version 3.1.0 (2014-04-10) -- "Spring Dance"  
Copyright (C) 2014 The R Foundation for Statistical Computing  
Platform: x86\_64-w64-mingw32/x64 (64-bit)

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

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

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

```
> 1+1  
[1] 2  
> 100/3  
[1] 33.33333  
> |
```

数値計算ができます

# 基本的な利用法

R Console

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

```
> 1+1  
[1] 2  
> 100/3  
[1] 33.33333  
>  
> uge <- 100  
> uge/3  
[1] 33.33333  
> |
```

ugeというものの(オブジェクト)に100という数値を代入しておいて、それを利用することもできます。

1	→
2	←
3	矢印
4	↓
5	⇒
6	↑
7	↔
8	やじるし
9	野次るし

# 基本的な利用法

The screenshot shows the R Console window. It displays the following text:

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

> 1+1
[1] 2
> 100/3
[1] 33.33333
>
> uge <- 100
> uge/3
[1] 33.33333
>
> age = 100
> age/3
[1] 33.33333
> |
```

A yellow callout box with a black border and arrow points from the text "age = 100" to the line "age/3". The text inside the box is:

<- の代わりに = を利用するのでも  
よいです。が、門田は”<-”派です。

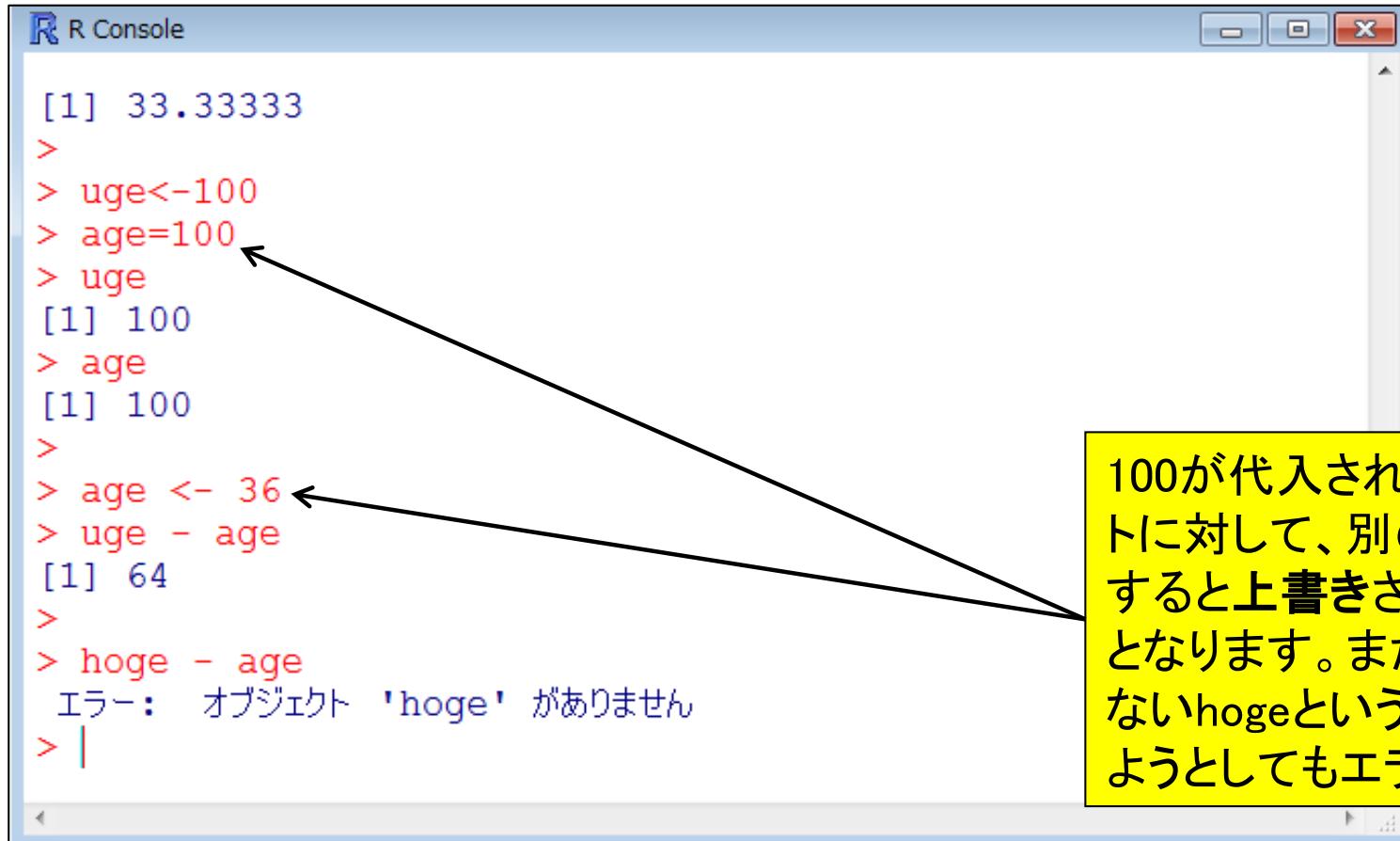
# 基本的な利用法

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

```
>
> uge <- 100
> uge/3
[1] 33.33333
>
> age = 100
> age/3
[1] 33.33333
>
> uge<-100 ←
> age=100 ←
> uge
[1] 100
> age
[1] 100
> |
```

A yellow callout box with a black border contains the Japanese text: "スペースを開けなくてもよいが、スペースを空けるのが一般的です。" (It's fine to leave spaces, but leaving them is common).

# 基本的な利用法



```
R Console
[1] 33.33333
>
> uge<-100
> age=100
> uge
[1] 100
> age
[1] 100
>
> age <- 36
> uge - age
[1] 64
>
> hoge - age
エラー: オブジェクト 'hoge' がありません
> |
```

The screenshot shows an R console window titled "R Console". It displays the following session history:

- [1] 33.33333
- >
- > uge<-100
- > age=100
- > uge
- [1] 100
- > age
- [1] 100
- >
- > age <- 36
- > uge - age
- [1] 64
- >
- > hoge - age
- エラー: オブジェクト 'hoge' がありません
- > |

Annotations with arrows point from the text "age=100" and "age <- 36" to a yellow callout box on the right.

100が代入されていたageオブジェクトに対して、別の数値を新たに代入すると上書きされます。 $100 - 36 = 64$ となります。また、何も代入されていないhogeというオブジェクトを利用しようとしてもエラーが出るだけです。

# バイオインフォ系研究者の実験ノート

- 過去のお知らせ (last modified 2014/08/03) NEW
- Rのインストールと起動 (last modified 2014/07/31) NEW
- 基本的な利用法 (last modified 2014/07/20) NEW
- サンプルデータ (last modified 2014/07/17) NEW
- バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ) | 速習コース | modified 2014/07/30) NEW
- 書籍 | ブラウザによる解析 | 2.2.1 RNA ～データセクションのファイル (last modified 2014/04/15)

## バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ) | 速習コース NEW

2014年9月にJST-NBDCと東大農アグリバイオ主催で「バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ)速習コース」が開催されます。主催機関のサイト上で情報提供したほうがいいだろうということで、受講者が各自でインストールするソフトウェアや、イメージファイルのダウンロードなど準備していただく計算機環境の情報などを示します。

### バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ)関連:

- NBDCの速習コース
- HPCの速習コース
- カリキュラムを策定
- 「NBDCで実施した」
  - 「バイオイン」
  - 「カリキュラム」
  - 「カリキュラム」

- 2014年9月5日15:00-18:15、「2-2. バイオ系データベース」
- 小野浩雅(DBCLS) 統合TV、講義資料
- 基本的な各種バイオ系データベースの理解、統合DBの利用法。
  
- 2014年9月8日10:30-12:00、「3-1. R 基礎1」、初級、実習
- 門田幸二(東京大学)、統合TV、講義資料(20140822, 18:04版)
- Rインストール自体は基本的に終了した状態を想定しているものの、最初にlibrary(Biostrings)などいくつかの利用予定パッケージのロードを行い、パッケージのインストールがうまくいっているかどうかを確認(できていなかったヒトの同定および対処)。Rの一般的な利用法。log関数などの基本的かつ挙動を完全に把握できる関数を例として、関数内部のオプション変更や「?関数名」で利用法の幅を広げる基本テクを概観。exp, mean, median, sort, length関数。
  - 9/8-9の2日間で用いるファイル群: hoge.zip (20140822, 17:24版)
  - Rコード: rcode\_20140908.txt

rcode\_20140908.txtはhogeフォルダ中あります。

### 計算機環境構築(Linux系)

8月初旬をめどにBio-Linu

- VirtualBox
  - VirtualBoxを
- Bio-Linux: Field et

- 2014年9月8日13:15-14:45、「3-2. R 基礎2」、初級、実習
- 門田幸二(東京大学)、統合TV、講義資料(20140811, 0:11版)
- 翻訳配列の取得を例に「(Rで)塩基配列解析」の基本的な利用法を紹介。塩基配列中にNを

# コピペを推奨(爆)

ファイル名 : rcode\_20140908.txt

```
#####
### 基本的な利用法
#####
1+1↓
100/3↓
↓
uge <- 100↓
uge/3↓
↓
age = 100↓
age/3↓
↓
uge<-100↓
age=100↓
uge↓
age↓
↓
age <- 36↓
uge - age↓
↓
hoge - age↓
↓
```

① コピー —

age <- 36↓  
uge - age↓

R Console

```
[1] 33.33333
>
> uge<-100
> age=100
> uge
[1] 100
> age
[1] 100
>
> age <- 36
> uge - age
[1] 64
>
> hoge - age
エラー: オブジェクト 'hoge' がありません
> |
```

②

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

スペルミスを避けるため、私は「メモ帳」などのテキストエディタに、必要なコマンド群(スクリプト、コード、Rコードなどという表現が一般的)をあらかじめ作成しておき、コピペで実行しています。このファイルがバイオインフォ系研究者の再現可能な実験ノートのようなものです。

# コピペを推奨

ファイル名 : rcode\_20140908.txt

```
#####
### 基本的な利用法
#####
1+1↓
100/3↓
↓
uge <- 100↓
uge/3↓
↓
age = 100↓
age/3↓
↓
uge<-100↓
age=100↓
uge↓
age↓
↓
age <- 36↓
uge - age↓
↓
hoge - age↓
↓
```

① コピペ

R Console

```
[1] 100
>
> age <- 36
> uge - age
[1] 64
>
> hoge - age
エラー: オブジェクト 'hoge' がありません
> age <- 36
> uge - age
[1] 64
>
> hoge - age
エラー: オブジェクト 'hoge' がありません
>
> |
```

コピペで実行したところ。コピーする際に最後のコマンドの改行まできちんと含めるところがポイントです。

# コピペを推奨

ファイル名 : rcode\_20140908.txt

```
#####
### 基本的な利用法
#####
1+1↓
100/3↓
↓
uge <- 100↓
uge/3↓
↓
age = 100↓
age/3↓
↓
uge<-100↓
age=100↓
uge↓
age↓
↓
age <- 36↓
uge - age↓
↓
hoge - age↓
↓
```

① コピペ

```
R Console
[1] 64
>
> hoge - age
エラー: オブジェクト 'hoge' がありません
> age <- 36
> uge - age
[1] 64
>
> hoge - age
エラー: オブジェクト 'hoge' がありません
>
> age <- 36
> uge - age
[1] 64
>
> hoge - age|
```

コピペで実行したところ。コピーする際に最後のコマンドの改行まできちんと含めるところがポイントです。この場合、改行が含まれていないので、最後のコマンドが実行されていないことがわかります。リターンキーを押して、コマンドを実行させましょう。

# 基本的な利用法2

ファイル名 : rcode\_20140908.txt

```
#####
### 基本的な利用法2
#####
uge <- 100
log(uge)
log(uge)      # デフォルトの底は自然対数
#
log(uge, base=10) # 底を10にしたい場合
log(uge, base = 2) # 底を2にしたい場合
#
log10(uge)      # log10という関数も存在する
log2(uge)        # log2関数もある
loge(uge)        # 底がeの関数は存在しない。
log(uge)         # log関数のデフォルトがeだ
#
log10(uge, base=2) # baseというオプションはない
```

①コピー

#から始まる行をR Console画面に  
コピペしてもかまいません。また、コ  
マンドの右側に#から始まるコメント  
を書いてもかまいません。#のあと  
に関数などを書いても無視されます

R Console

```
> #####
> ### 基本的な利用法2
> #####
> uge <- 100
> log(uge)
[1] 4.60517
> log(uge)      # デフォルトの底は自然対数
[1] 4.60517
> |
```

# 基本的な利用法2

ファイル名 : rcode\_20140908.txt

R R Console

```
> #####  
> ### 基本的な利用法2  
> #####  
> uge <- 100  
> log(uge)  
[1] 4.60517  
> log(uge)  
[1] 4.60517  
> uge <- 36 | # デフォルトの底は自然対数
```

100が代入されていたugeに別の数値をいれて挙動確認したい場合は、最初から全部打ち込み直すではなく、上矢印キーで以前に打ち込んだコマンドを有効利用すべし！

上下左右の矢印キーを有効に利用し最小限の労力で打つべし！

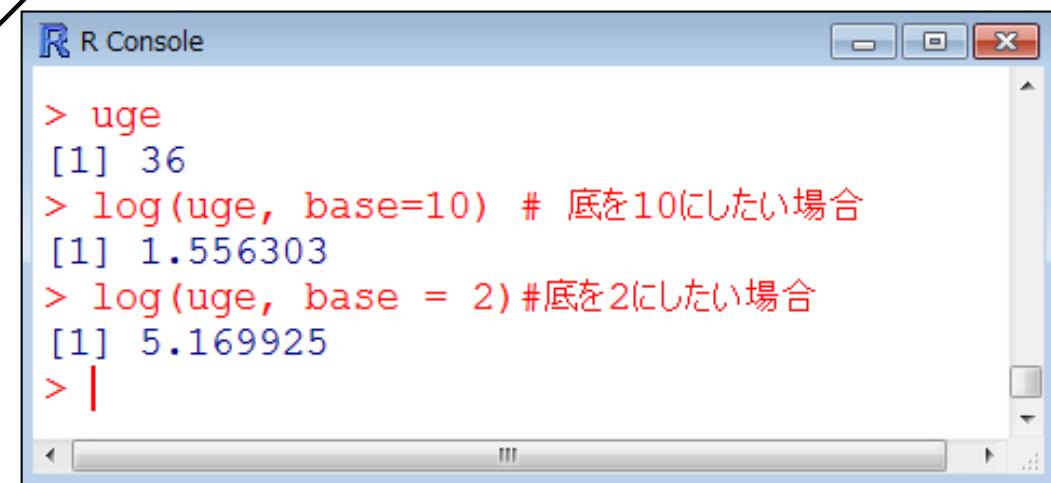


# 基本的な利用法2

ファイル名 : rcode\_20140908.txt

```
#####
### 基本的な利用法2
#####
uge <- 100
log(uge)
log(uge)      # デフォルトの底は自然対数
#
uge
log(uge, base=10) # 底を10にしたい場合
log(uge, base = 2) # 底を2にしたい場合
#
log10(uge)    # log10という関数も存在する
log2(uge)      # log2関数もある
loge(uge)      # 底がeの関数は存在しない...
log(uge)      # log関数のデフォルトがeだから
#
log10(uge, base=2) # baseというオプションはない
```

log関数のデフォルトの底はe (=2.718282)。しかし、baseというオプションを利用することで、任意の数値を底とすることができます。



R Console window showing R code and its output:

```
> uge
[1] 36
> log(uge, base=10) # 底を10にしたい場合
[1] 1.556303
> log(uge, base = 2) # 底を2にしたい場合
[1] 5.169925
> |
```

# 基本的な利用法2

おさらいです

ファイル名 : rcode\_20140908.txt

```
#####↓  
### 基本的な利用法2↓  
#####↓  
uge <- 100↓  
log(uge)↓  
log(uge)      # デフォルトの底は自然対数↓  
↓  
uge↓  
log(uge, base=10) # 底を10にしたい場合↓  
log(uge, base = 2) # 底を2にしたい場合↓  
↓  
log10(uge)      # log10という関数も存在する↓  
log2(uge)        # log2関数もある↓  
loge(uge)        # 底がeの関数は存在しない...↓  
log(uge)         # log関数のデフォルトがeだから↓  
↓  
log10(uge, base=2) # baseというオプションはない↓  
↓
```

```
R Console  
> uge  
[1] 36  
> log(uge, base=10) # 底を10にしたい場合  
[1] 1.556303  
> log(uge, base = 2) # 底を2にしたい場合  
[1] 5.169925  
> exp(1)  
[1] 2.718282  
> 10^1.556303  
[1] 36.00004  
> 2^5.169925  
[1] 36  
> log(1)  
[1] 0  
> 10^0  
[1] 1  
> 2^0  
[1] 1  
> |
```

# 基本的な利用法2

ファイル名 : rcode\_20140908.txt

```
#####
### 基本的な利用法2
#####
uge <- 100
log(uge)
log(uge)      # デフォルトの底は自然対数
#
uge
log(uge, base=10) # 底を10にしたい場合
log(uge, base = 2) # 底を2にしたい場合
#
log10(uge)    # log10という関数も存在する
log2(uge)    # log2関数もある
loge(uge)    # 底がeの関数は存在しない...
log(uge)      # log関数のデフォルトがeだから
#
log10(uge, base=2) # baseというオプションはない
```

R Console

```
> uge
[1] 36
> log(uge, base=10) # 底を10にしたい場合
[1] 1.556303
> log(uge, base = 2) # 底を2にしたい場合
[1] 5.169925
> log10(uge)          # log10という関数も存在する
[1] 1.556303
> log2(uge)           # log2関数もある
[1] 5.169925
> loge(uge)           # 底がeの関数は存在しない...
エラー: 関数 "loge" を見つけることができませんでした
> log(uge)             # log関数のデフォルトがeだから
[1] 3.583519
>
```

一つの目的を達成するうえでも多くのやり方が存在します。例えば、**log**関数利用時に**base=10**を利用するのと**log10**関数を利用するのと同じです。また存在しない関数を実行すればエラーが出ます。重要なのはエラーにうろたえるのではなく、出たメッセージとの関連を適切に把握し、対処すること。

# 基本的な利用法2

ファイル名 : rcode\_20140908.txt

```
#####
### 基本的な利用法2
#####
uge <- 100
log(uge)
log(uge)      # デフォルトの底は自然対数
#
uge
log(uge, base=10) # 底を10にしたい場合
log(uge, base = 2) # 底を2にしたい場合
#
log10(uge)     # log10という関数も存在する
log2(uge)       # log2関数もある
loge(uge)       # 底がeの関数は存在しない...
log(uge)        # log関数のデフォルトがeだから
#
log10(uge, base=2) # baseというオプションはない
```

log10という関数は底が10の対数をとるものなので、baseオプションで底を指定するという行為 자체が非論理的。数値のみを入力とするのが妥当。これは1個の引数(argument)のみをlog10関数に与えるべきということに相当する。

R Console

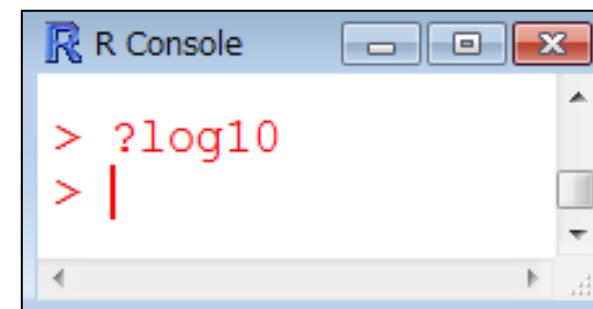
```
> log10(uge, base=2) # baseというオプションはない
   以下にエラー log10(uge, base = 2) :
   2 個の引数が 'log10' に渡されました but 1 が必要とされています
> |
```

# 基本的な利用法3

ファイル名 : rcode\_20140908.txt

```
#####  
### 基本的な利用法3  
#####  
?log10          # log10関数の使用法を調べる  
#  
help(log10)      # log10関数の使用法を調べる  
#  
x <- c(3, 2, 9, 2) # 数値ベクトルxを作成  
x                  # xの中身を表示  
mean(x)           # 平均値  
(3+2+9+2)/4     # 平均値  
average(x)        # エクセルと違ってaverageはない  
median(x)         # 中央値  
log2(x)           # ベクトルの要素ごとに計算  
#  
sort(x)  
sort(x, decreasing = FALSE)  
sort(x, decreasing=FALSE)  
sort(x, decreasing=F)  
sort(x, decreasing=TRUE)  
sort(x, decreasing=T)  
#
```

関数ごとに利用可能なオプションや、どのようなデータを入力として受け付けるのかなどを記したhtmlマニュアルが存在します。それを見る手段は「?関数名」または「help(関数名)」です。



## Logarithms and Exponentials

### Description

`log` computes logarithms, by default natural logarithms, `log10` computes common (i.e., base 10) logarithms, and `log2` computes binary (i.e., base 2) logarithms. The general form `log(x, base)` computes logarithms with base `base`.

`log1p(x)` computes  $\log(1+x)$  accurately also for  $|x| \ll 1$ .

`exp` computes the exponential function.

`expm1(x)` computes  $\exp(x) - 1$  accurately also for  $|x| \ll 1$ .

### Usage

```
log(x, base = exp(1))
logb(x, base = exp(1))
log10(x)
log2(x)

log1p(x)

exp(x)
expm1(x)
```

### Arguments

`x` a numeric or complex vector.

`base` a positive or complex number: the base with respect to which logarithms are computed. Defaults to `e=exp(1)`.

Descriptionには関数の概要が書かれている。

Usageには基本的な利用法が書かれている。`log`関数には引数が2つ、`log10`関数には引数が1つしかないことはここでわかる。

Argumentsには引数に関する説明が書かれている。`base`オプションのデフォルトが`e`であることはここでもわかる。

全く知らない関数のマニュアルを見てもさっぱりわからない(個人の感想です)ため、この例のようにある程度挙動既知の関数を眺め、記述形式に慣れておくとよい。

## Details

続き…

All except `logb` are generic functions: methods can be defined for them individually or via the [Math](#) group generic.

詳細な説明。関数によってあたりなかつたりします。

`log10` and `log2` are only convenience wrappers, but logs to bases 10 and 2 (whether computed via `log` or the wrappers) will be computed more efficiently and accurately where supported by the OS. Methods can be set for them individually (and otherwise methods for `log` will be used).

`logb` is a wrapper for `log` for compatibility with S. If (S3 or S4) methods are set for `log` they will be dispatched. Do not set S4 methods on `logb` itself.

All except `log` are [primitive](#) functions.

関数の返り値に関する説明。  
この関数を実行するとどんな  
結果が返されるかという説明  
書き。

## Value

A vector of the same length as `x` containing the transformed values. `log(0)` gives `-Inf`, and `log(x)` for negative values of `x` is `NaN`. `exp(-Inf)` is 0.

...

For complex inputs to the log functions, the value is a complex number with imaginary part in the range  $[-\pi, \pi]$ : which end of the range is used might be platform-specific.

## S4 methods

`exp`, `expm1`, `log`, `log10`, `log2` and `log1p` are S4 generic and are members of the [Math](#) group generic.

Note that this means that the S4 generic for `log` has a signature with only one argument, `x`, but that `base` can be passed to methods (but will not be used for method selection). On the other hand, if you only set a method for the `Math` group generic then `base` argument of `log` will be ignored for your class.

## Source

続き…

`log1p` and `expm1` may be taken from the operating system, but if not available there then they are based on the Fortran subroutine `dlnrel` by W. Fullerton of Los Alamos Scientific Laboratory (see <http://www.netlib.org/slatec/fnlib/dlnrel.f> and (for small  $x$ ) a single Newton step for the solution of  $\log1p(y) = x$  respectively.

## References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole. (for `log`, `log10` and `exp`.)

Chambers, J. M. (1998) *Programming with Data. A Guide to the S Language*. Springer. (for `logb`.)

## See Also

[Trig](#), [sqrt](#), [Arithmetic](#).

## Examples

```
log(exp(3))  
log10(1e7) # = 7  
  
x <- 10^{-(1+2*1:9)}  
cbind(x, log(1+x), log1p(x), exp(x)-1, expm1(x))
```

[Package *base* version 3.1.0 [Index](#)]

The screenshot shows an R console window with the title "R Console". It contains the following R code and output:

```
> log(exp(3))
[1] 3
> log10(1e7) # = 7
[1] 7
>
```

利用例。こここの記述をコピペして挙動確認したりします。

# 基本的な利用法3

ベクトル演算ができます

ファイル名 : rcode\_20140908.txt

```
#####↓  
### 基本的な利用法3↓  
#####↓  
?log10           # log10関数の使用法を調べる↓  
↓  
help(log10)      # log10関数の使用法を調べる↓  
↓  
x <- c(3, 2, 9, 2) # 数値ベクトルxを作成↓  
x                  # xの中身を表示↓  
mean(x)           # 平均値↓  
(3+2+9+2)/4     # 平均値↓  
average(x)         # エクセルと違ってaverageはない  
median(x)          # 中央値↓  
log2(x)           # ベクトルの要素ごとに計算↓  
↓  
sort(x)↓  
sort(x, decreasing = FALSE)↓  
sort(x, decreasing=FALSE)↓  
sort(x, decreasing=F)↓  
sort(x, decreasing=TRUE)↓  
sort(x, decreasing=T)↓  
↓
```

R R Console

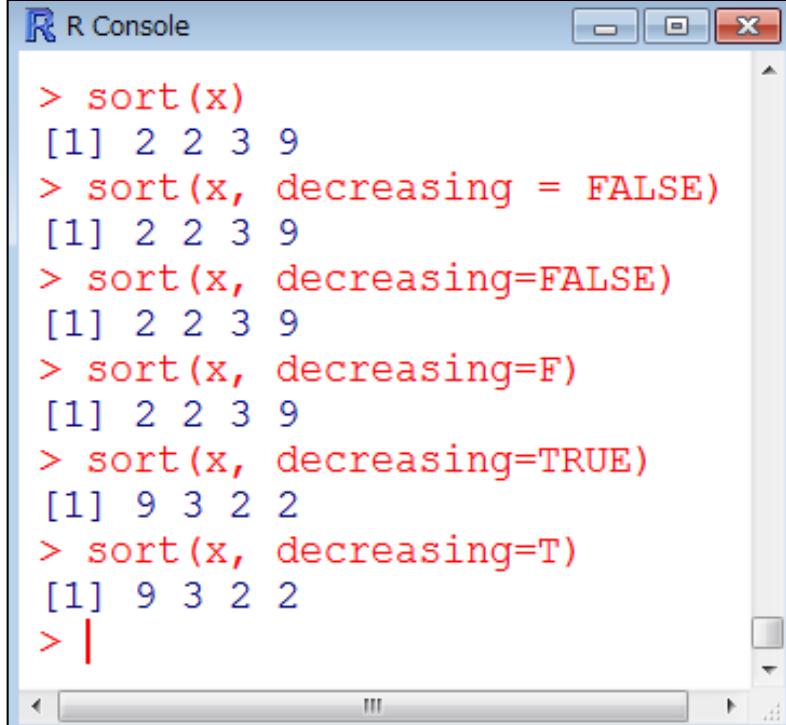
```
> x <- c(3, 2, 9, 2) # 数値ベクトルxを作成  
> x                      # xの中身を表示  
[1] 3 2 9 2  
> mean(x)                # 平均値  
[1] 4  
> (3+2+9+2)/4           # 平均値  
[1] 4  
> average(x)              # エクセルと違ってaverageはない  
エラー: 関数 "average" を見つけることができませんで$  
> median(x)               # 中央値  
[1] 2.5  
> log2(x)                 # ベクトルの要素ごとに計算  
[1] 1.584963 1.000000 3.169925 1.000000  
> |
```

# 基本的な利用法3

ファイル名 : rcode\_20140908.txt

```
#####↓
### 基本的な利用法3↓
#####↓
?log10           # log10関数の使用法を調べる↓
↓
help(log10)      # log10関数の使用法を調べる↓
↓
x <- c(3, 2, 9, 2) # 数値ベクトルxを作成↓
x                  # xの中身を表示↓
mean(x)           # 平均値↓
(3+2+9+2)/4      # 平均値↓
average(x)        # エクセルと違ってaverageはない
median(x)         # 中央値↓
log2(x)           # ベクトルの要素ごとに計算↓
↓
sort(x)           ↓
sort(x, decreasing = FALSE) ↓
sort(x, decreasing=FALSE) ↓
sort(x, decreasing=F) ↓
sort(x, decreasing=TRUE) ↓
sort(x, decreasing=T) ↓
↓
```

最初の2行分で、sort関数のデフォルトはdecreasing = FALSEであることが分かります。decreasingオプションは、「ソートの際降順にするか否か」なので、FALSEを指定することは「降順にしない」つまり「昇順でソート」を意味します。



R Console window showing R code and its output. The code demonstrates the use of the sort function with different arguments.

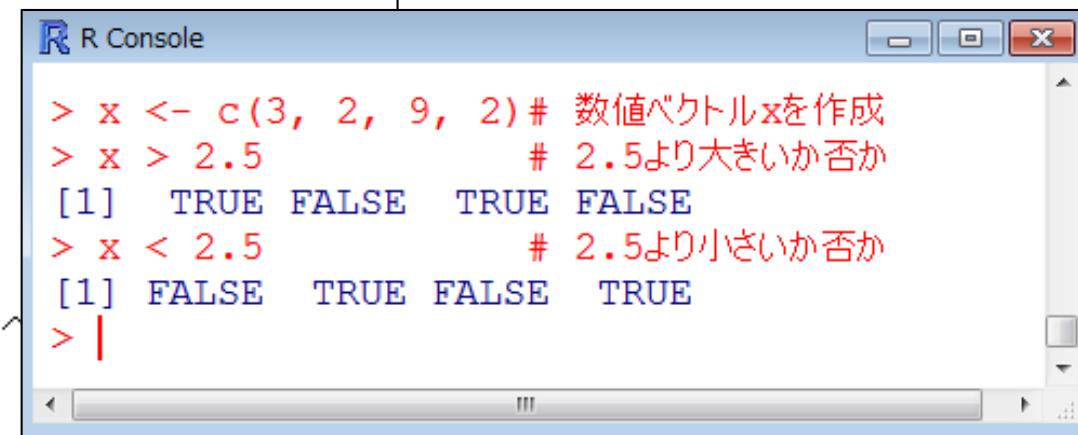
```
> sort(x)
[1] 2 2 3 9
> sort(x, decreasing = FALSE)
[1] 2 2 3 9
> sort(x, decreasing=FALSE)
[1] 2 2 3 9
> sort(x, decreasing=F)
[1] 2 2 3 9
> sort(x, decreasing=TRUE)
[1] 9 3 2 2
> sort(x, decreasing=T)
[1] 9 3 2 2
> |
```

# 基本的な利用法4

ファイル名 : rcode\_20140908.txt

```
#####  
### 基本的な利用法4  
#####  
  
x <- c(3, 2, 9, 2) # 数値ベクトルxを作成  
x > 2.5           # 2.5より大きいか否か  
x < 2.5           # 2.5より小さいか否か  
  
↓  
k <- c(-3, 2, 9, 2) # 数値ベクトルxを作成  
k > -2.5          # -2.5より大きいか否か  
k < -2.5          # -2.5より小さいか否か  
  
↓  
obj <- k < -2.5   # -2.5より小さいか否かという論理値へ  
obj               # objの中身を表示  
obj <- (k < -2.5) # ()で囲って見やすくしてもよい  
obj               # objの中身を表示  
↓
```

指定した条件を満たす要素の位置を TRUEまたはFALSEからなるベクトル(論理値ベクトル)で表すことができます。「～より大きい」の">”や「～より小さい」の”<“以外にも、「～以上」の”>=“とか「～以下」の”<=“なども指定可能です。



R Console

```
> x <- c(3, 2, 9, 2) # 数値ベクトルxを作成  
> x > 2.5           # 2.5より大きいか否か  
[1] TRUE FALSE TRUE FALSE  
> x < 2.5           # 2.5より小さいか否か  
[1] FALSE TRUE FALSE TRUE  
> |
```

# 基本的な利用法4

ファイル名 : rcode\_20140908.txt

```
#####  
### 基本的な利用法4  
#####  
x <- c(3, 2, 9, 2) # 数値ベクトルxを作成  
x > 2.5 # 2.5より大きいか否か  
x < 2.5 # 2.5より小さいか否か  
  
#  
k <- c(-3, 2, 9, 2) # 数値ベクトルxを作成  
k > -2.5 # -2.5より大きいか否か  
k < -2.5 # -2.5より小さいか否か  
  
#  
obj <- k < -2.5 # -2.5より小さいか否かという論理値  
obj # objの中身を表示  
obj <- (k < -2.5) # ()で囲って見やすくしてもよい  
obj # objの中身を表示  
#
```

もちろんマイナスの要素を含むベクトル  
kでも問題なく条件を満たすかどうかを  
判定可能です。

R R Console

```
> k <- c(-3, 2, 9, 2) # 数値ベクトルxを作成  
> k > -2.5 # -2.5より大きいか否か  
[1] FALSE TRUE TRUE TRUE  
> k < -2.5 # -2.5より小さいか否か  
[1] TRUE FALSE FALSE FALSE  
> |
```

# 基本的な利用法4

ファイル名 : rcode\_20140908.txt

```
#####
### 基本的な利用法4
#####
x <- c(3, 2, 9, 2) # 数値ベクトルxを作成
x > 2.5           # 2.5より大きいか否か
x < 2.5           # 2.5より小さいか否か
#
k <- c(-3, 2, 9, 2) # 数値ベクトルxを作成
k > -2.5          # -2.5より大きいか否か
k < -2.5          # -2.5より小さいか否か
#
obj <- k < -2.5 # -2.5より小さいか否かという論理値ベクトルをobjに格納
obj             # objの中身を表示
obj <- (k < -2.5) # ()で囲って見やすくしてもよい
obj             # objの中身を表示
#
```

赤色の下線部分が論理値ベクトルの部分。これをobjに格納することもできる。ただし、条件判定を意味する”<“か代入を意味する”<-”がわかりづらいため、見やすくすることを目的として括弧をつけたりすることもあります。

R Console

```
> obj <- k < -2.5    # -2.5より小さいか否か
> obj                 # objの中身を表示
[1] TRUE FALSE FALSE FALSE
> obj <- (k < -2.5) # ()で囲って見やすく
> obj                 # objの中身を表示
[1] TRUE FALSE FALSE FALSE
> |
```

# 基本的な利用法5

ファイル名 : rcode\_20140908.txt

```
#####↓  
### 基本的な利用法5↓  
#####↓  
x <- c(3, 2, 9, 2) # 数値ベクトルxを作成↓  
obj <- (x > 2.5) # 2.5より大きいか否かの結果をobjに格納↓  
obj # objの中身を表示↓  
x[obj] # objがTRUEとなる要素のみを表示↓
```

```
↓  
posi <- c(1,3) # 数値ベクトルposiを作成↓  
posi # posiの中身を表示↓  
x[posi] # posiで指定した要素番号を表示↓  
↓  
posi <- 2:4 # 数値ベクトルposiを作成↓  
posi # posiの中身を表示↓  
x[posi] # posiで指定した要素番号を表示↓
```

論理値ベクトルobjの実際の利用例です。  
「4つの要素からなるベクトル」を「4つの遺伝子からなる、あるサンプルの発現ベクトル」と考えて、発現レベルが一定以上を満たすもののみを抽出するようなイメージです。

R R Console

```
> x <- c(3, 2, 9, 2) # 数値ベクトルxを作成  
> obj <- (x > 2.5) # 2.5より大きいか否か$  
> obj # objの中身を表示  
[1] TRUE FALSE TRUE FALSE  
> x[obj] # objがTRUEとなる要素$  
[1] 3 9  
> |
```

# 基本的な利用法5

ファイル名 : rcode\_20140908.txt

```
#####  
### 基本的な利用法5  
#####  
x <- c(3, 2, 9, 2) # 数値ベクトルxを作成  
obj <- (x > 2.5) # 2.5より大きいか否かの結果をobjに格納  
obj # objの中身を表示  
x[obj] # objがTRUEとなる要素のみを表示  
  
posi <- c(1,3) # 数値ベクトルposiを作成  
posi # posiの中身を表示  
x[posi] # posiで指定した要素番号を表示  
  
posi <- 2:4 # 数値ベクトルposiを作成  
posi # posiの中身を表示  
x[posi] # posiで指定した要素番号を表示
```

posiで指定した特定の位置の要素のみ抽出する基本形です。

R R Console

```
> posi <- c(1,3)      # 数値ベクトルposiを作成  
> posi                # posiの中身を表示  
[1] 1 3  
> x[posi]             # posiで指定した要素を表示  
[1] 3 9  
> |
```

# 基本的な利用法5

ファイル名 : rcode\_20140908.txt

```
#####  
### 基本的な利用法5  
#####  
x <- c(3, 2, 9, 2) # 数値ベクトルxを作成  
obj <- (x > 2.5) # 2.5より大きいか否かの結果をobjに格納  
obj # objの中身を表示  
x[obj] # objがTRUEとなる要素のみを表示  
#  
posi <- c(1,3) # 数値ベクトルposiを作成  
posi # posiの中身を表示  
x[posi] # posiで指定した要素番号を表示  
#  
posi <- 2:4 # 数値ベクトルposiを作成  
posi # posiの中身を表示  
x[posi] # posiで指定した要素番号を表示  
#
```

「2:4」は「c(2,3,4)」と同じ意味です。「1:1000」のような数値範囲が大きくなる場合に便利な表記法です。

```
R R Console  
> posi <- 2:4      # 数値ベクトルposiを作成  
> posi            # posiの中身を表示  
[1] 2 3 4  
> x[posi]          # posiで指定した要素を表示  
[1] 2 9 2  
> |
```

# 基本的な利用法6

ファイル名 : rcode\_20140908.txt

```
#####  
### 基本的な利用法6  
#####  
x <- c(3, 2, 9, 2) # 数値ベクトルxを作成  
length(x)           # ベクトルxの要素数を表示
```

```
posi <- 1:3          # 数値ベクトルposiを作成  
sort(x)[posi]        # xを昇順にソートし、最初の3つを表示  
sort(x, decreasing=T)[posi] # xを降順にソートし、最初の3つを表示
```

length関数はベクトルの要素数を返します。ベクトルの要素数分だけ何かをしたいときにlength関数を利用することがあります。

R R Console

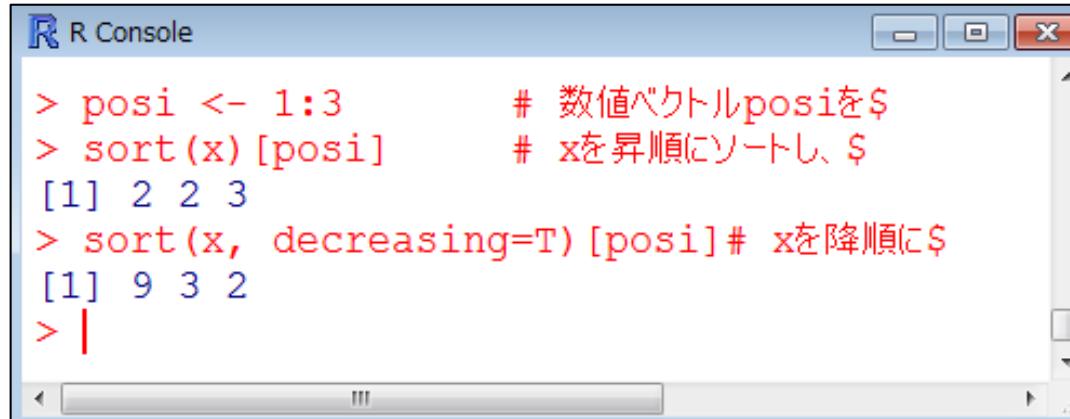
```
> x <- c(3, 2, 9, 2) # 数値ベクトルxを作成  
> length(x)           # ベクトルxの要素数を表示  
[1] 4  
> |
```

# 基本的な利用法6

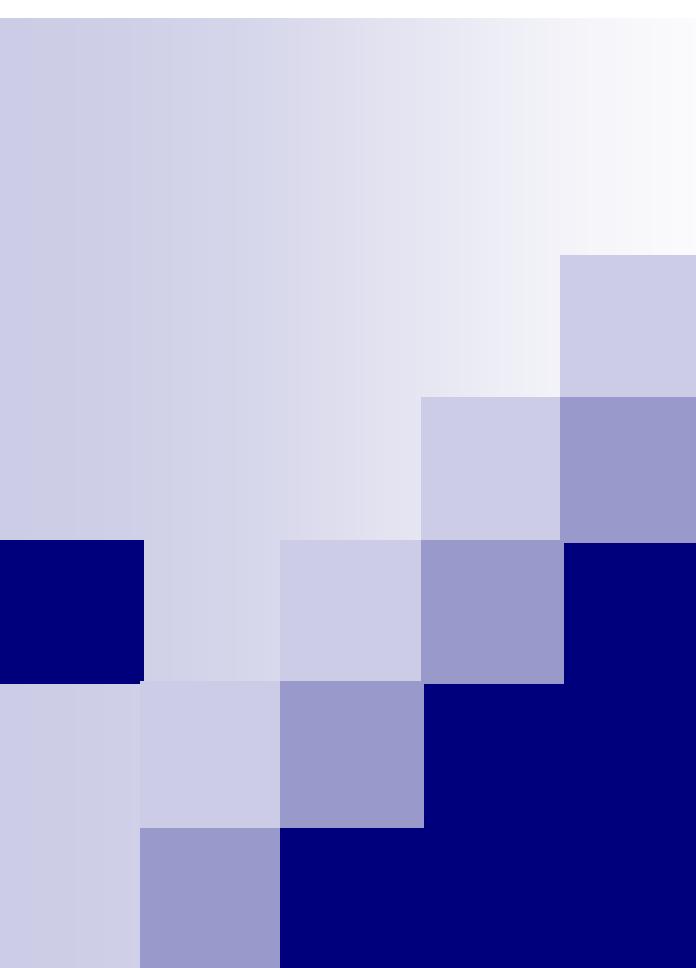
ファイル名 : rcode\_20140908.txt

```
#####↓  
### 基本的な利用法6↓  
#####↓  
x <- c(3, 2, 9, 2) # 数値ベクトルxを作成↓  
length(x)           # ベクトルxの要素数を表示↓  
↓  
posi <- 1:3         # 数値ベクトルposiを作成↓  
sort(x)[posi]       # xを昇順にソートし、最初の3つを表示↓  
sort(x, decreasing=T)[posi] # xを降順にソートし、最初の3つを表示↓  
↓
```

sort関数で昇順にソートし、最初の3つの要素を表示する組合せ(発展形)です。発現レベルの低い順や高い順に並べ、上位3つを表示させることと同義。



```
R Console  
> posi <- 1:3          # 数値ベクトルposiを作成  
> sort(x)[posi]        # xを昇順にソートし、最初の3つを表示  
[1] 2 2 3  
> sort(x, decreasing=T)[posi] # xを降順にソートし、最初の3つを表示  
[1] 9 3 2  
> |
```



# バイオインフォマティクス人材育成カリ キュラム(次世代シークエンサ)速習 コース

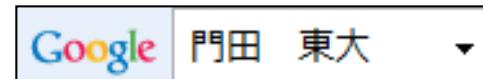
## 3. データ解析基礎 | 3-2. R 基礎2

東京大学・大学院農学生命科学研究科  
アグリバイオインフォマティクス教育研究ユニット

門田幸二(かどた こうじ)

[kadota@iu.a.u-tokyo.ac.jp](mailto:kadota@iu.a.u-tokyo.ac.jp)

<http://www.iu.a.u-tokyo.ac.jp/~kadota/>



# Contents

## ■ 3-2. R 基礎2、2014/09/08 13:15–14:45、初級、実習

- (Rで)塩基配列解析の基本的な利用法(翻訳配列の取得を例に)
  - 入力ファイル取得、作業ディレクトリの変更、本番(基本はコピペ)、出力結果の確認
  - 1つの項目に多数の例題(異なる入力ファイル、エラーへの対処例)
- 行列形式ファイルの解析基礎(アノテーションファイルを例に)
  - 例題をテンプレートとして任意の解析を行う基本手順
  - ありがちなミスとエラーメッセージ
  - 入力ファイルの最後の改行の有無
  - プログラム内部の説明(行列演算の基礎)
- Tips
  - 集合演算:union, intersect, setdiff
  - その他:sort, table, is.element, toupper, tolower



# (Rで)塩基配列解析

~NGS、RNA-seq、ゲノム、トランскриプトーム、正規化、発現変動、統計、モデル、バイオイ  
(last modified 2014/08/22, since 2010)

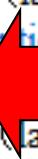
塩基配列を入力として、その翻訳された  
アミノ酸配列を取得することができます

## What's new?

- このウェブへ  
1. Rのインストール
- 2014年10月  
フォマティクス
- 門田幸二著
- バイオインフ  
一通りの手順
- 日本乳酸菌
- 参考資料(語)

- はじめに (last modified 2014/08/22)
- 参考資料 (last modified 2014/08/22)
- 過去のお知らせ
- Rのインストール
- 基本的な利用法
- サンプルデータ
- バイオインフ
- 書籍 | トラン
- 書籍 | トラン
- 書籍 | トラン

- ・ イントロ | 一般 | 任意の長さの可能な全ての塩基配列を作成 (last modified 2013/06/14)
- ・ イントロ | 一般 | 任意の位置の塩基を置換 (last modified 2013/09/12)
- ・ イントロ | 一般 | 指定した範囲の配列を取得 (last modified 2014/03/08)
- ・ イントロ | 一般 | 指定したID(染色体や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 (last modified 2013/06/14)
- ・ イントロ | 一般 | 相補鎖(complement)を取得 (last modified 2013/06/14)
- ・ イントロ | 一般 | 逆相補鎖(reverse complement)を取得 (last modified 2013/06/14)
- ・ イントロ | 一般 | 逆鎖(reverse)を取得 (last modified 2013/06/14)
- ・ イントロ | 一般 | 2連続塩基の出現頻度情報を取得 (last modified 2014/07/18) NEW
- ・ イントロ | 一般 | 3連続塩基の出現頻度情報を取得 (last modified 2013/06/14)
- ・ イントロ | 一般 | 任意の長さの連続塩基の出現頻度情報を取得 (last modified 2013/06/14)
- ・ イントロ | 一般 | Tips | 任意の拡張子でファイルを保存 (last modified 2013/09/26)
- ・ イントロ | 一般 | Tips | 拡張子は同じで任意の文字を追加して保存 (last modified 2013/09/26)
- ・ イントロ | 一般 | 配列取得 | ゲノム配列 | 公共DBから (last modified 2014/05/28)
- ・ イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome (last modified 2014/06/28) NEW
- ・ イントロ | 一般 | 配列取得 | プロモーター配列 | 公共DBから (last modified 2014/04/02)
- ・ イントロ | 一般 | 配列取得 | プロモーター配列 | BSgenome (last modified 2014/04/25)
- ・ イントロ | 一般 | 配列取得 | プロモーター配列 | GenomicFeatures(Lawrence 2013) (last modified 2014/04/23)
- ・ イントロ | 一般 | 配列取得 | トランскриプトーム配列 | 公共DBから (last modified 2014/04/02)
- ・ イントロ | 一般 | 配列取得 | トランскриプトーム配列 | biomaRt(Durinck 2009) (last modified 2013/09/25)
- ・ イントロ | NGS | 様々なプラットフォーム (last modified 2014/06/10)
- ・ イントロ | NGS | qPCRやmicroarrayなどの比較 (last modified 2014/07/11) NEW
- ・ イントロ | NGS | 可視化(ゲノムブラウザやViewer) (last modified 2014/06/25) NEW
- ・ イントロ | NGS | 配列取得 | FASTQ or SRALite | 公共DBから (last modified 2014/06/28) NEW
- ・ イントロ | NGS | 配列取得 | FASTQ or SRALite | SRAdb(Zhu 2013) (last modified 2014/06/26) NEW



- ・ イントロ | 一般 | 任意の長さの可能な全ての塩基配列を作成 (last modified 2013/06/14)
- ・ イントロ | 一般 | 任意の位置の塩基を置換 (last modified 2013/09/12)
- ・ イントロ | 一般 | 指定した範囲の配列を取得 (last modified 2014/03/08)
- ・ イントロ | 一般 | 指定したID(染色体やdescriptor)の配列を取得 (last modified 2014/03/10)
- ・ イントロ | 一般 | 翻訳配列(translate)を取得 (last modified 2013/06/14)
- ・ イントロ | 一般 | 相補鎖(complement)を取得 (last modified 2013/06/14)
- ・ イントロ | 一般 | 逆相補鎖(reverse complement)を取得 (last modified 2013/06/14)
- ・ イントロ | 一般 | 逆鎖(reverse)を取得 (last modified 2013/06/14)
- ・ イントロ | 一般 | 2連続塩基の出現頻度情報を取得 (last modified 2014/07/18) NEW
- ・ イントロ | 一般 | 3連続塩基の出現頻度情報を取得 (last modified 2013/06/14)

- ・ イントロ | 一般 | 翻訳配列(translate)を取得

塩基配列を入力として、その翻訳されたアミノ酸配列を取得することができます

## イントロ | 一般 | 翻訳配列(translate)を取得 NEW

塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。

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

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

```

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

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

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

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

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

```



# hogeフォルダの作成

デスクトップにあるhogeフォルダ中のファイルを解析するやり方として説明します

R Console

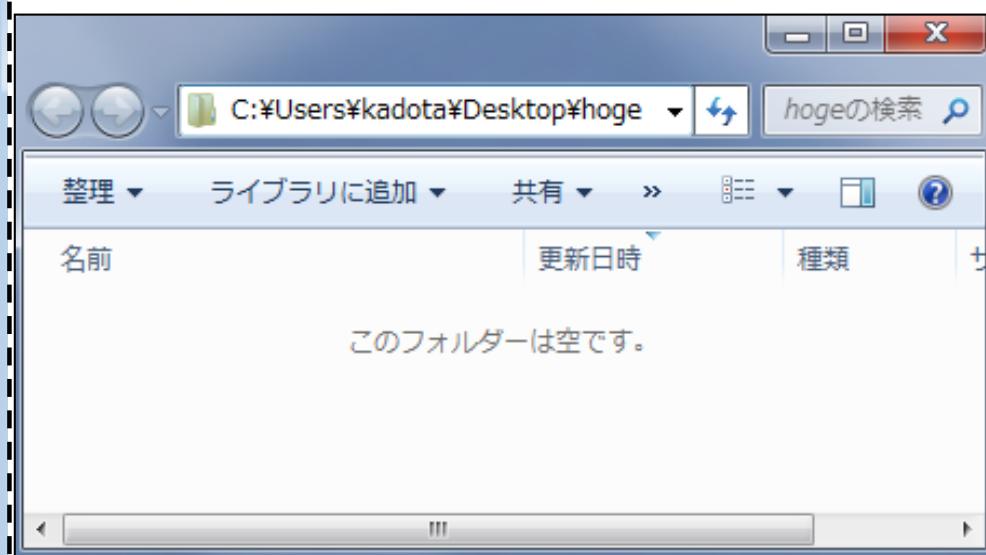
```
R version 3.1.0 (2014-04-10) -- "Spring Dance"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

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

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

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

> 1+1
[1] 2
> 100/3
[1] 33.33333
> |
```



塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。

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

### 1. FASTA形式ファイル(sample1.fasta)の場合は

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

#必要なパッケージをロード
library(Biostrings)

#入力ファイルの読み込み
fasta <- readDNAStringSet()

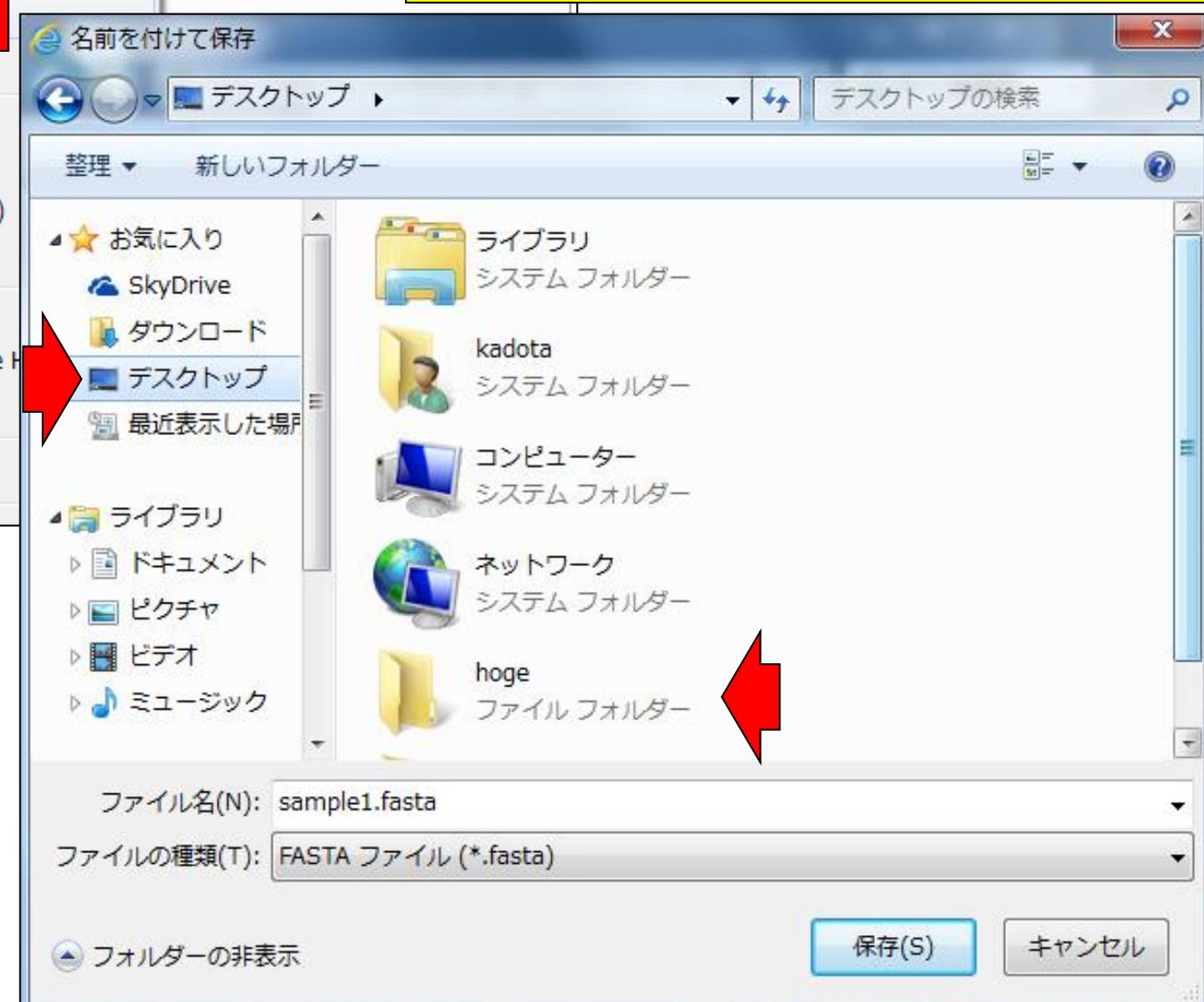
#本番
hoge <- translate(fasta)
names(hoge) <- names(fasta)
fasta <- hoge
fasta

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

開く(O)  
新しいタブで開く(W)  
新しいウィンドウで開く(N)  
**対象をファイルに保存(S)**  
対象を印刷(P)

してin\_fに格納  
してout\_fに格納

解析したいファイルsample1.fastaをhoge  
フォルダ中に保存(本当は既にhogeフォ  
ルダ中に存在するが一般論として説明)



名前を付けて保存

C:\Users\kadota\Desktop\hoge hogeの検索

整理 新しいフォルダー

お気に入り SkyDrive ダウンロード デスクトップ 最近表示した場所

ライブラリ ドキュメント ピクチャ ビデオ ミュージック

ファイル名(N): sample1.fasta

ファイルの種類(T): FASTA ファイル (\*.fasta)

保存(S) キャンセル

検索条件に一致する項目はありません。

ときどき拡張子が\*.txtなどと勝手に変わっていることがあるのでファイルの種類欄に注意すべし

解析したいファイルsample1.fastaをhoge  
フォルダ中に保存(本当は既にhoge  
フォルダ中に存在するが一般論として説明)

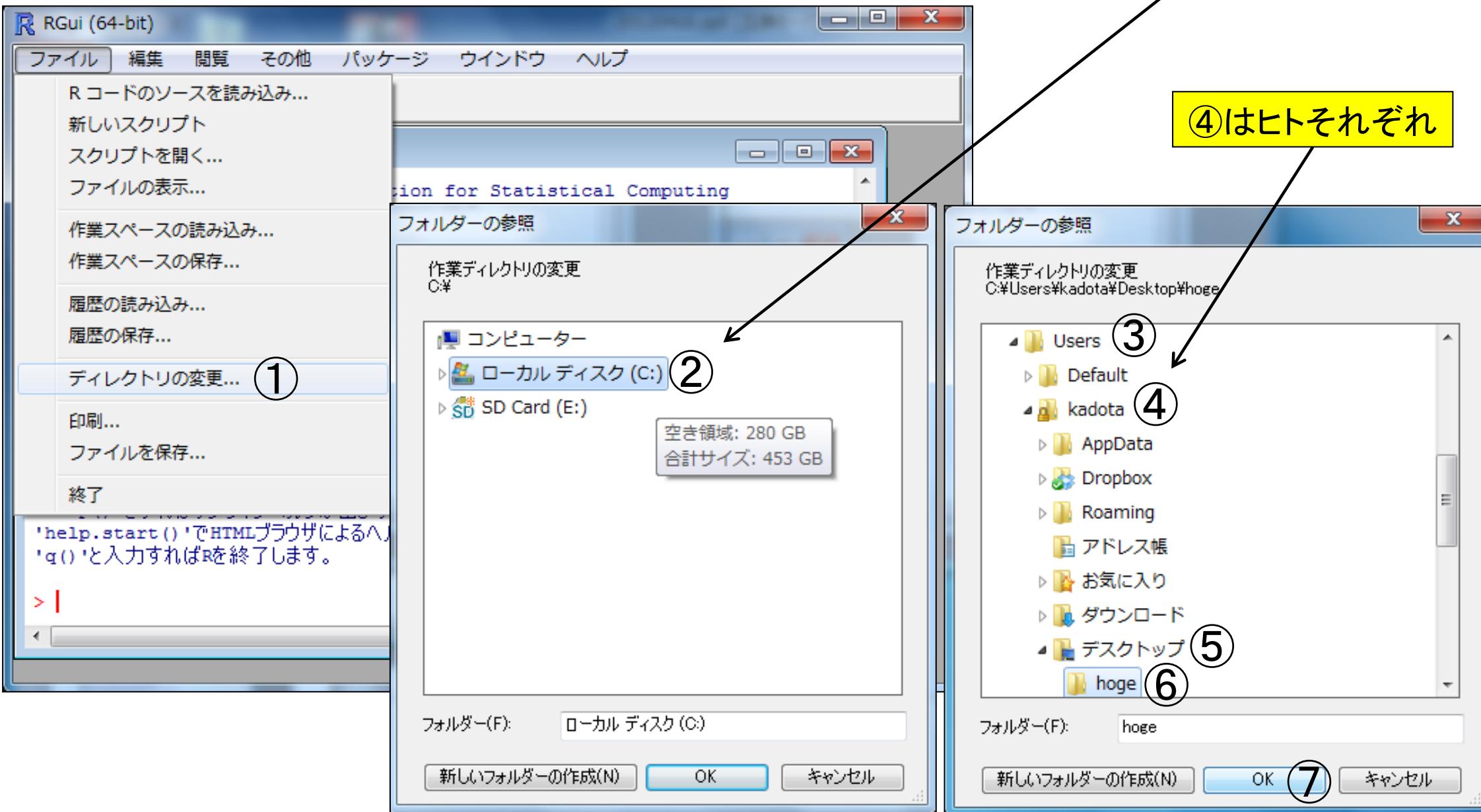
sample1.fasta - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V)  
ヘルプ(H)

>kadota  
AGTGACGGTCTT

# 作業ディレクトリの変更

「Windows(C:)」となっている場合もあるが、気にしない



# getwd()と打ち込んで確認

The screenshot shows the RGui (64-bit) application window. The main title bar is "R Gui (64-bit)". Below it is a menu bar with Japanese labels: ファイル (File), 編集 (Edit), 閲覧 (View), その他 (Other), パッケージ (Package), ウィンドウ (Window), and ヘルプ (Help). Under the "View" menu, there are several icons: a folder, a file, a document, a magnifying glass, a search icon, a refresh, and a stop sign.

The central window is titled "R Console". It displays the following text:

```
Platform: x86_64-pc-mingw32/x64 (64-bit)

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

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

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

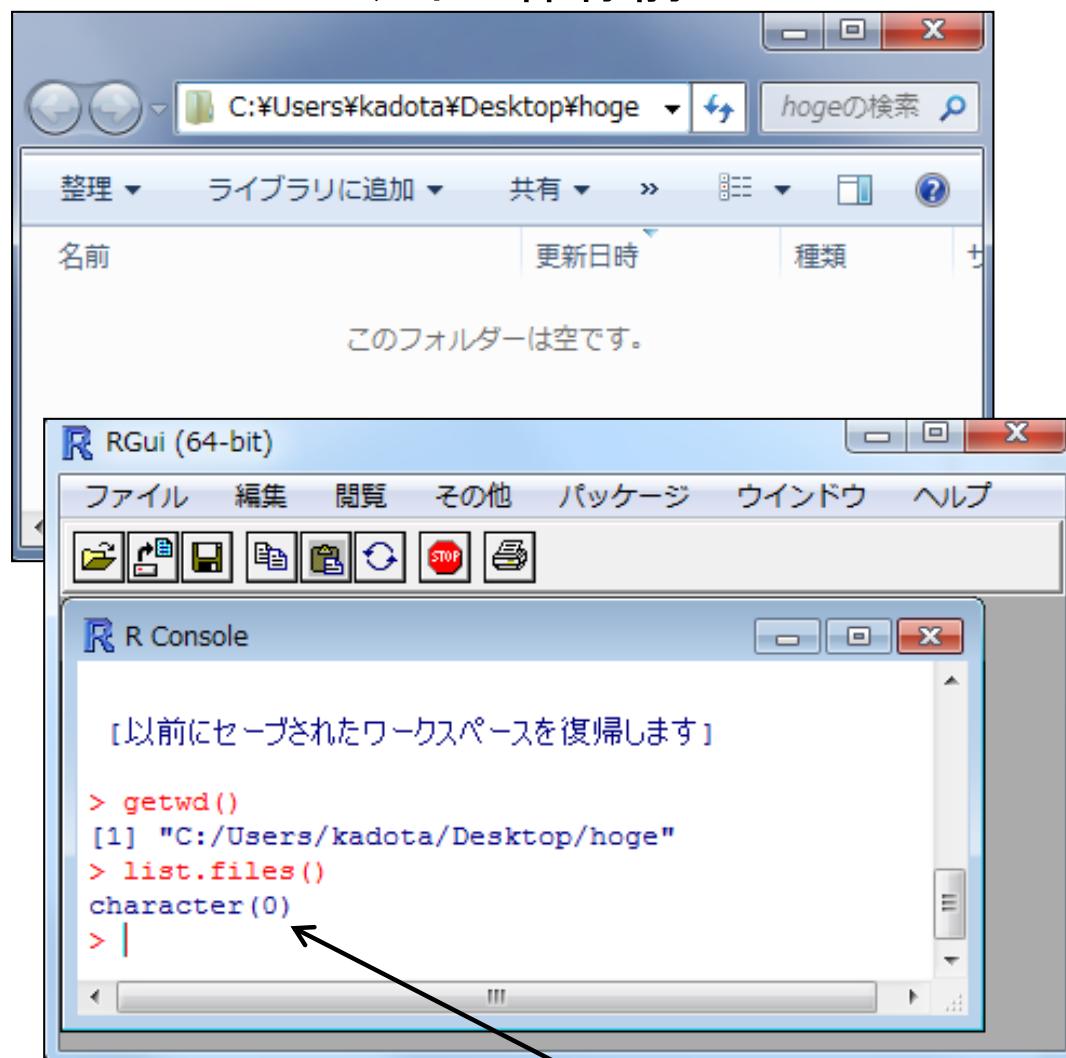
At the bottom of the console window, there is a command prompt:  
[1] > getwd()  
[1] > [REDACTED]

A yellow callout box highlights the text: "当たり前ですが、解析したいディレクトリ(またはフォルダ)を正しく指定できていなければエラーに遭遇します。また、解析したいファイルが存在しない状態でもエラーが出ます" (It is common sense that if you do not specify the correct directory (or folder), an error will occur. Additionally, if the file you want to parse does not exist, an error will occur).

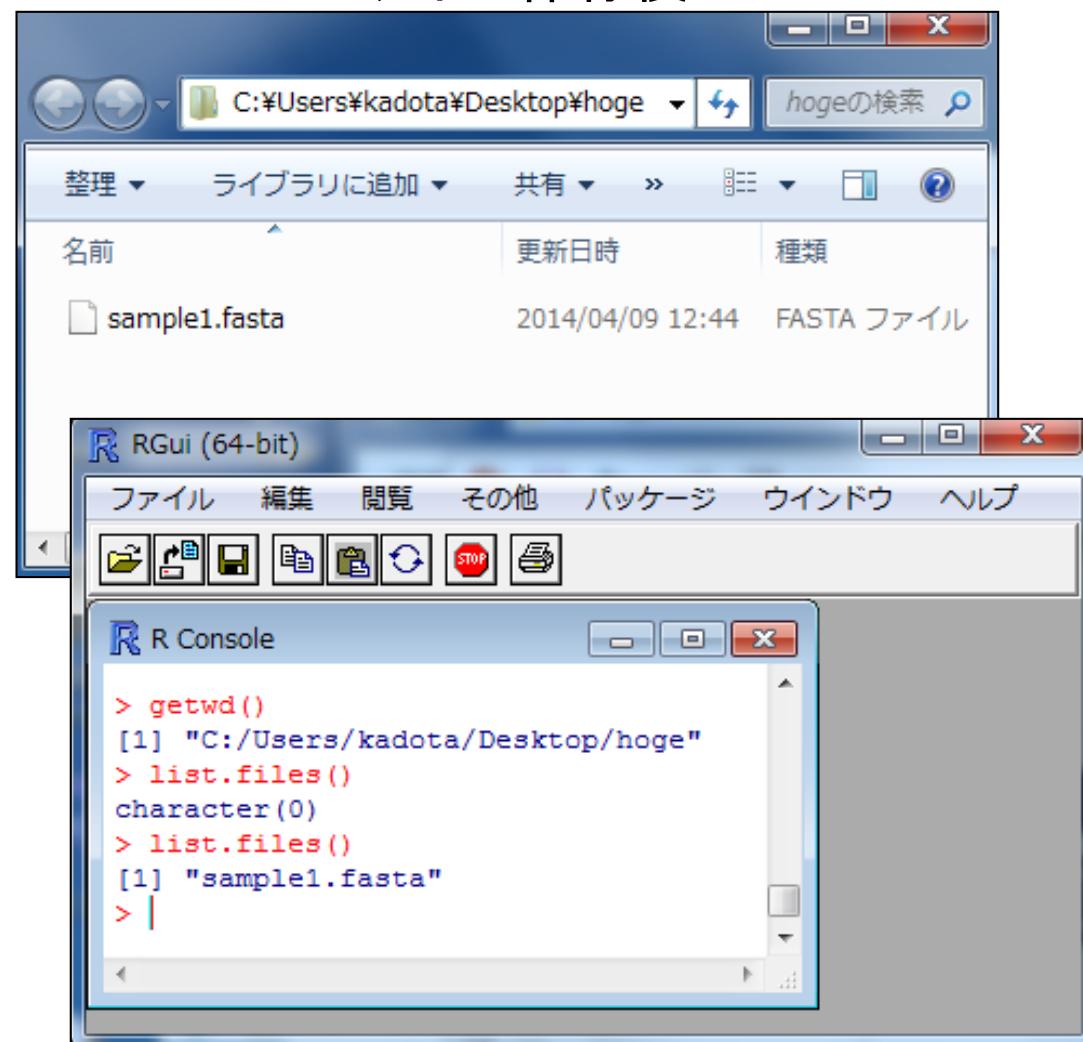


# 実際のhogeフォルダとR操作画面の関係

ファイル保存前



ファイル保存後



character(0)は何もないという意味です



# 基本はコピペ

イントロ | 一般 | 翻訳配列(translate)を取得 NEW

Windowsのヒトは、CTRLとALTキーを押しながらコードの枠内で左クリックすると、全選択できます。Macintoshはよくわかりません。

塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。

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

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

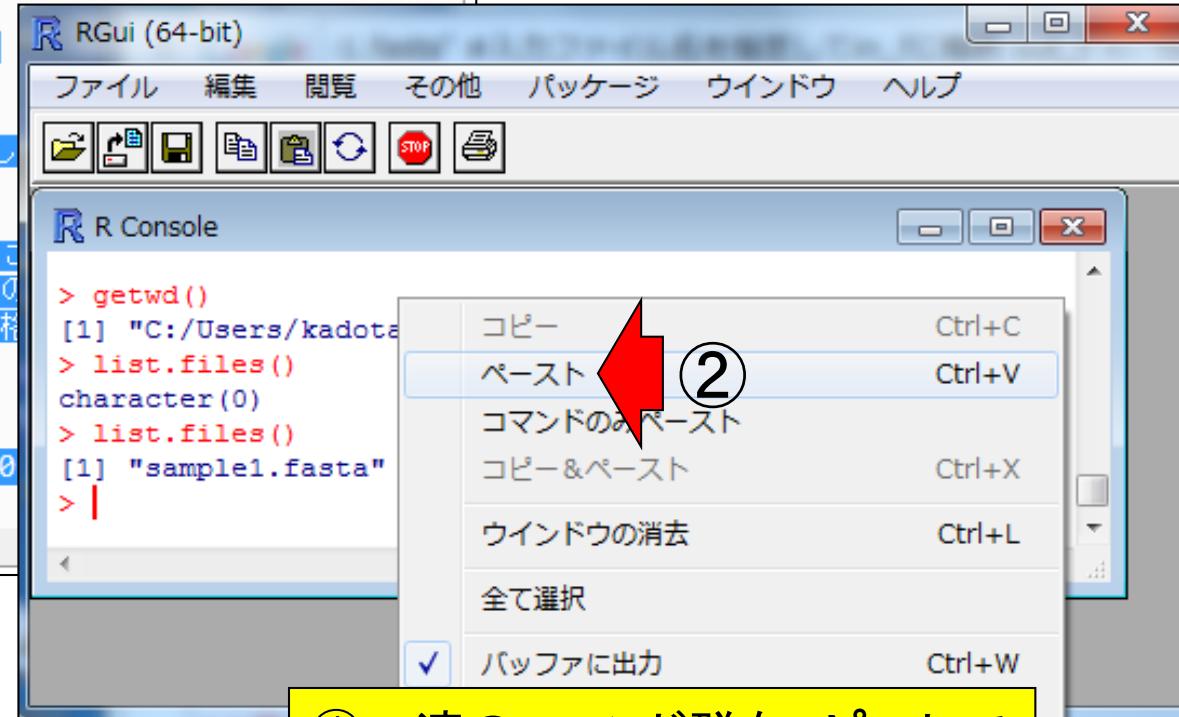
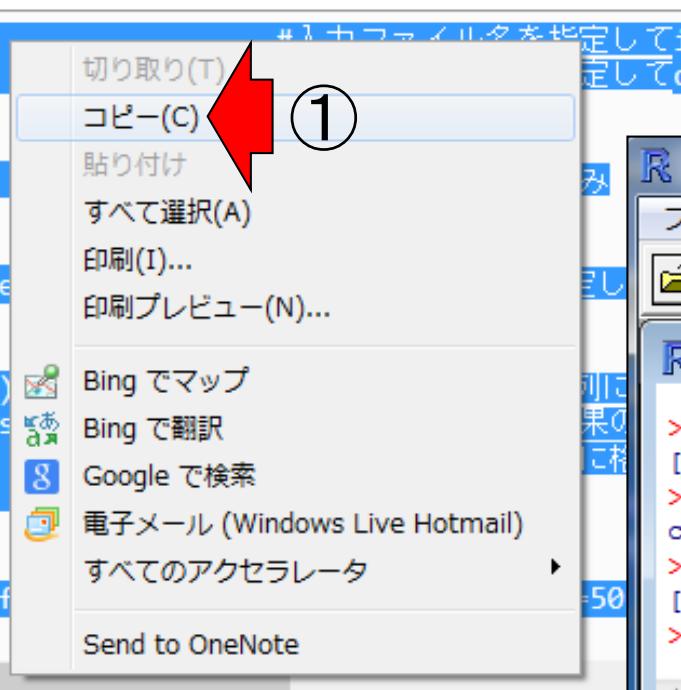
```
in_f <- "sample1.fasta"
out_f <- "hoge1.fasta"

#必要なパッケージをロード
library(Biostrings)

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

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

#ファイルに保存
writeXStringSet(fasta, "hoge1.fasta")
```



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

```

> in_f <- "sample1.fasta"          #入力ファイル名を指定してin_fに格納
> out_f <- "hogel.fasta"         #出力ファイル名を指定してout_fに格納
>
> #必要なパッケージをロード
> library(Biostrings)           #パッケージの読み込み
要求されたパッケージ BiocGenerics をロード中です
要求されたパッケージ parallel をロード中です

次のパッケージを付け加えます: 'BiocGenerics'

以下のオブジェクトはマスクされています (from 'package:parallel') :

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

以下のオブジェクトはマスクされています (from 'package:stats') :

  xtabs

以下のオブジェクトはマスクされています (from 'package:base') :

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

要求されたパッケージ IRanges をロード中です
要求されたパッケージ XVector をロード中です
>
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fasta")#in_fで指定したファイルの読み込み
>
> #本番
> hoge <- translate(fasta)           #fastaをアミノ酸配列に翻訳した結果をhogeに格納
> names(hoge) <- names(fasta)        #現状では翻訳した結果のオブジェクトhogeのdescription行が消す
> fasta <- hoge                     #hogeの中身をfastaに格納
> fasta
A AAStringSet instance of length 1
  width seq
[1]      4 SDGL
                           names
                           kadota
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=50)#fastaの中身を指定したファイル名で保存
> |

```

## エラーなく実行できた場合の全貌

# 実行結果

R Console

```

> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fast$"
>
> #本番
> hoge <- translate(fasta)                      #fast$
> names(hoge) <- names(fasta)                   #現状$
> fasta <- hoge                                 #hoge$
> fasta                                         #確認$

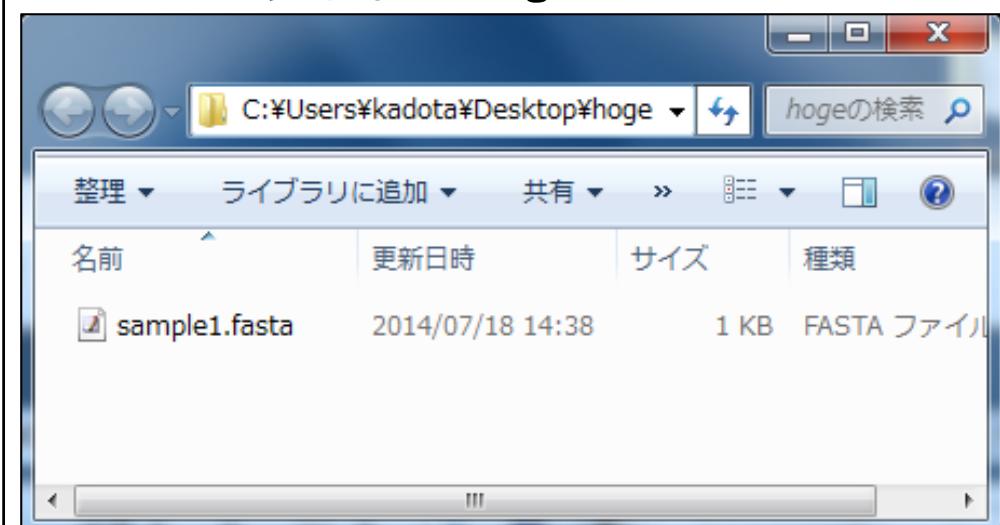
A AAStringSet instance of length 1
  width seq
[1]      4 SDGL
  names   $ 
  kadota

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

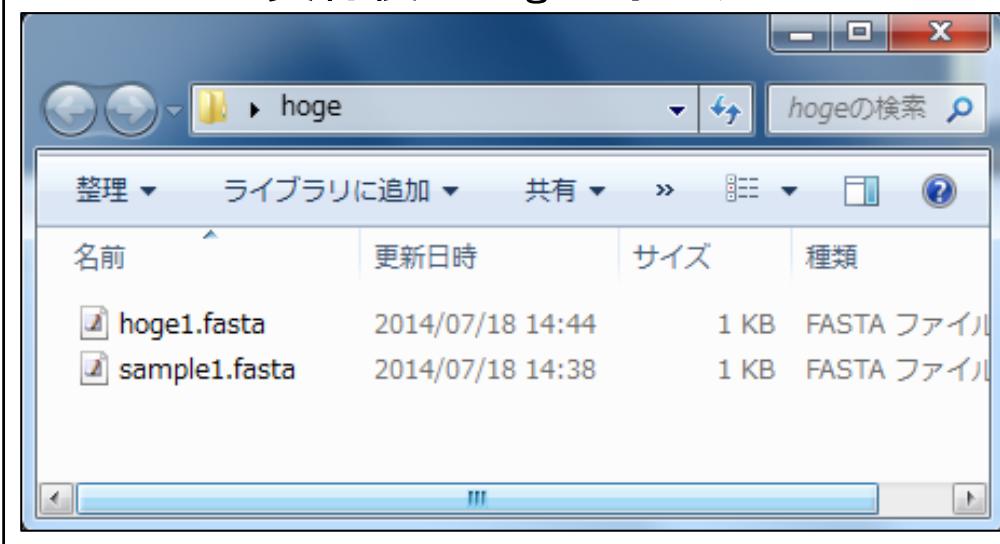
```

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

実行前のhogeフォルダ

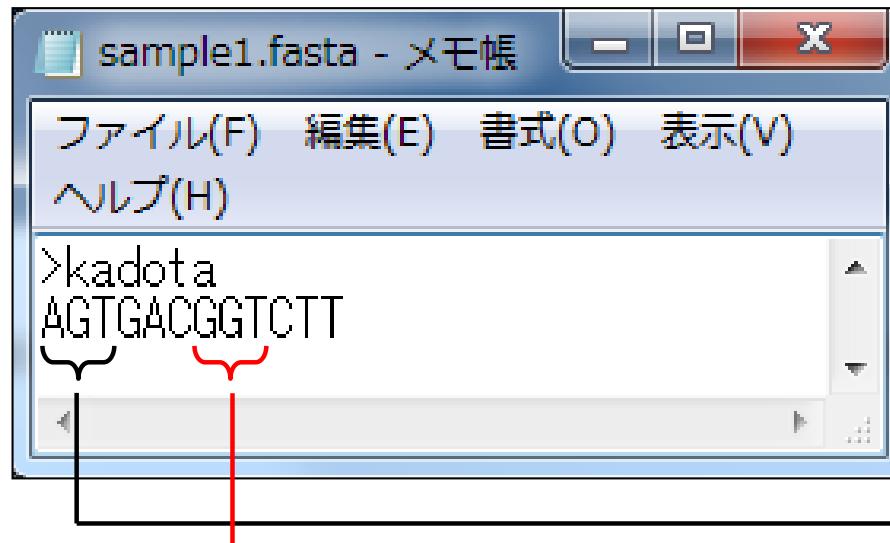


実行後のhogeフォルダ



# 実行結果

入力: 塩基配列ファイル(sample1.fasta)

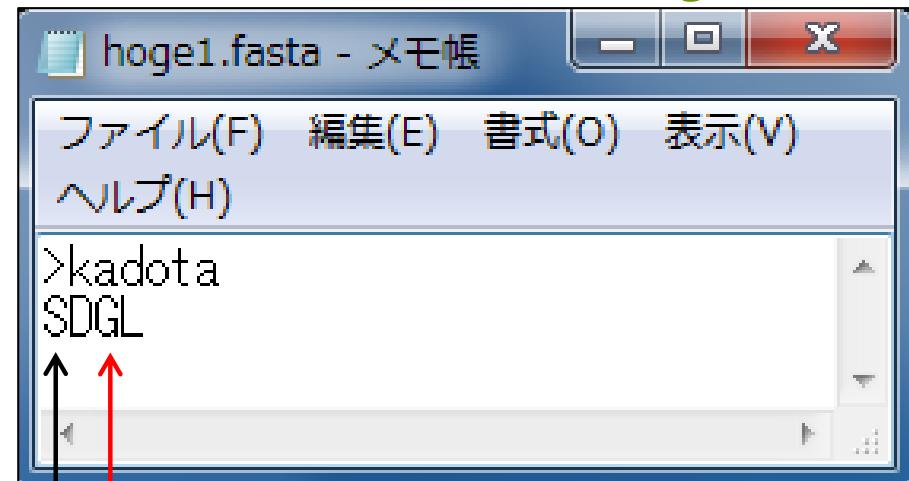


```
sample1.fasta - メモ帳
```

ファイル(F) 編集(E) 書式(O) 表示(V)  
ヘルプ(H)

```
>kadota
AGTGACGGTCTT
```

出力: アミノ酸配列ファイル(hoge1.fasta)



```
hoge1.fasta - メモ帳
```

ファイル(F) 編集(E) 書式(O) 表示(V)  
ヘルプ(H)

```
>kadota
SDGL
```

3の倍数の12塩基長、ACGTのみ  
からなるので何のエラーも出ない

# コドン表

<http://ja.wikipedia.org/wiki/%E3%82%B3%E3%83%89%E3%83%B3>

表1. 64コドンと各々に対応するアミノ酸を示したもの。mRNAの方向は5'から3'である。

		2nd base			
		U	C	A	G
1st base	U	UUU (Phe/F) <b>フェニルアラニン</b> UUC (Phe/F) <b>フェニルアラニン</b> UUA (Leu/L) <b>ロイシン</b> UUG (Leu/L) <b>ロイシン</b>	UCU (Ser/S) <b>セリン</b> UCC (Ser/S) <b>セリン</b> UCA (Ser/S) <b>セリン</b> UCG (Ser/S) <b>セリン</b>	UAU (Tyr/Y) <b>チロシン</b> UAC (Tyr/Y) <b>チロシン</b> UAA Ochre ( <b>終止</b> ) UAG Amber ( <b>終止</b> )	UGU (Cys/C) <b>システイン</b> UGC (Cys/C) <b>システイン</b> UGA Opal ( <b>終止</b> ) UGG (Trp/W) <b>トリプトファン</b>
	C	CUU (Leu/L) <b>ロイシン</b> CUC (Leu/L) <b>ロイシン</b> CUA (Leu/L) <b>ロイシン</b> CUG (Leu/L) <b>ロイシン</b>	CCU (Pro/P) <b>プロリン</b> CCC (Pro/P) <b>プロリン</b> CCA (Pro/P) <b>プロリン</b> CCG (Pro/P) <b>プロリン</b>	CAU (His/H) <b>ヒスチジン</b> CAC (His/H) <b>ヒスチジン</b> CAA (Gln/Q) <b>グルタミン</b> CAG (Gln/Q) <b>グルタミン</b>	CGU (Arg/R) <b>アルギニン</b> CGC (Arg/R) <b>アルギニン</b> CGA (Arg/R) <b>アルギニン</b> GGG (Arg/R) <b>アルギニン</b>
	A	AUU (Ile/I) <b>イソロイシン</b> AUC (Ile/I) <b>イソロイシン</b> AUA (Ile/I) <b>イソロイシン, (開始)</b> AUG (Met/M) <b>メチオニン, 開始</b> <sup>[3]</sup>	ACU (Thr/T) <b>スレオニン</b> ACC (Thr/T) <b>スレオニン</b> ACA (Thr/T) <b>スレオニン</b> ACG (Thr/T) <b>スレオニン</b>	AAU (Asn/N) <b>アスパラギン</b> AAC (Asn/N) <b>アスパラギン</b> AAA (Lys/K) <b>リシン</b> AAG (Lys/K) <b>リシン</b>	AGU (Ser/S) <b>セリン</b> AGC (Ser/S) <b>セリン</b> AGA (Arg/R) <b>アルギニン</b> AGG (Arg/R) <b>アルギニン</b>
	G	GUU (Val/V) <b>バリン</b> GUC (Val/V) <b>バリン</b> GUA (Val/V) <b>バリン</b> GUG (Val/V) <b>バリン, (開始)</b>	GCU (Ala/A) <b>アラニン</b> GCC (Ala/A) <b>アラニン</b> GCA (Ala/A) <b>アラニン</b> GCG (Ala/A) <b>アラニン</b>	GAU (Asp/D) <b>アスパラギン酸</b> GAC (Asp/D) <b>アスパラギン酸</b> GAA (Glu/E) <b>グルタミン酸</b> GAG (Glu/E) <b>グルタミン酸</b>	GGU (Gly/G) <b>グリシン</b> GGC (Gly/G) <b>グリシン</b> GGA (Gly/G) <b>グリシン</b> GGG (Gly/G) <b>グリシン</b>

# Contents

## ■ 3-2. R 基礎2、2014/09/08 13:15–14:45、初級、実習

### □ (Rで)塩基配列解析の基本的な利用法(翻訳配列の取得を例に)

- 入力ファイル取得、作業ディレクトリの変更、本番(基本はコピペ)、出力結果の確認
- 1つの項目に多数の例題(異なる入力ファイル、エラーへの対処例)

### □ 行列形式ファイルの解析基礎(アノテーションファイルを例に)

- 例題をテンプレートとして任意の解析を行う基本手順
- ありがちなミスとエラーメッセージ
- 入力ファイルの最後の改行の有無
- プログラム内部の説明(行列演算の基礎)

### □ Tips

- 集合演算:union, intersect, setdiff
- その他:sort, table, is.element, toupper, tolower



- ・ イントロ | 一般 | 指定した範囲の配列を取得 (last modified 2014/03/08)
- ・ イントロ | 一般 | 指定したID(染色体やdescriptor)の配列を取得 (last modified 2014/03/10)
- ・ イントロ | 一般 | 翻訳配列(translate)を取得 (last modified 2013/06/14)
- ・ イントロ | 一般 | 相補鎖(complement)を取得 (last modified 2013/06/14)
- ・ イントロ | 一般 | 逆相補鎖(reverse complement)を取得 (last modified 2013/06/14)

- ・ イントロ | 一般 | 翻訳配列(translate)を取得

## イントロ | 一般 | 翻訳配列(translate)を取得 NEW

塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。  
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し

### 1. FASTA形式ファイル(sample1.fasta)の場合:

in  
out

#必要な  
library

#入力フ  
asta

#本番  
hoge <  
names(  
fasta  
fasta

#ファイ  
writeX

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

配列中にACGT以外のものが存在するためエラーが出る例です。#電卓の電源を切ると電卓が壊れる

```

in_f <- "sample4.fasta"          #入力ファイル名を指定してin_fに格納
out_f <- "hoge4.fasta"           #出力ファイル名を指定してout_fに格納

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

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

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

```

各項目の例題は1つとは限りません。例えば4.はエラーが出る例。出力ファイルを盲目的に信じてはいけない、という典型例でもあります。入力ファイル(sample4.fasta)をhogeフォルダにダウンロードしてRを再起動して実行してみましょう。

#### 4. (multi-)FASTA形式ファイル([sample4.fasta](#))の場合:

配列中にACGT以外のものが存在するためエラーが出る例です。4番目の配列(つまりgene\_4)の17番目のポジションがNなので妥当です。

般 | 翻訳配列(translate)を取得

```
in_f <- "sample4.fasta"
out_f <- "hoge4.fasta"

#必要なパッケージをロード
library(Biostrings)

#入力ファイルの読み込み
fasta <- readDNAStringSet(fasta)

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

#ファイルに保存
writeXStringSet(fasta, f
```

#入力ファイル名を指定してin\_fに格  
#出力ファイル名を指定してout\_fに格

#パッケージの読み込み

- ①入力ファイルのダウンロード
- ②コピペで実行。実行時にエラーメッセージの有無を確認
- ③出力ファイルを眺める

```
R Console
> #本番
> hoge <- translate(fasta)          #fastaをアミノ酸配列に翻訳
                                     以下にエラー .Call2("DNAStringSet_translate", x, skip_code, dna_
                                     in 'x[[4]]': not a base at pos 17
> names(hoge) <- names(fasta)      #現状では翻訳した結果の$オブジェクト 'hoge' $をhogeの中身をfastaに格納
                                     以下にエラー names(hoge) <- names(fasta) : エラー: オブジェクト 'hoge' がありません
> fasta <- hoge                   #確認してるだけです
                                     A DNAStringSet instance of length 5
                                     width seq
[1] 21 CGACAGCTCCTCGGCATCCGA
[2] 27 GTCTGCCTCAAGCGCCCCAAGTGGGTT
[3] 21 TGTAGGAGAACGGCGTAATCT
[4] 30 CGTGCTGATTCCACACNGCAGTAAACGCGG
[5] 30 CGTGCTGATTCCACACAGCAGTAAACGCGG
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=50) #fa$
```

# 実行結果

入力 : sample4.fasta

```
>gene_1↓  
CGACAGCTCCTCGGCATCCGA↓  
>gene_2↓  
GTCTGCCTCAAGCGCCCCAAGTGGGTT↓  
>gene_3↓  
TGTAGGAGAACGGCGTAATCT↓  
>gene_4↓  
CGTGCTGATTCCACACNGCAGTAAACCGGGG↓  
>gene_5↓  
CGTGCTGATTCCACACAGCAGTAAACCGGGG↓  
←
```

出力 : hoge4.fasta

```
>gene_1↓  
CGACAGCTCCTCGGCATCCGA↓  
>gene_2↓  
GTCTGCCTCAAGCGCCCCAAGTGGGTT↓  
>gene_3↓  
TGTAGGAGAACGGCGTAATCT↓  
>gene_4↓  
CGTGCTGATTCCACACNGCAGTAAACCGGGG↓  
>gene_5↓  
CGTGCTGATTCCACACAGCAGTAAACCGGGG↓  
←
```

この結果がおかしいと思えるようになります。目的は翻訳配列の取得で得られた結果は塩基配列。これが、「得られた結果の合理的な解釈ができるようになる」の意味です。

> gene\_1  
 CGACAGCTCCTCGGCATCCGA  
 > gene\_2  
 GTCTGCCTCAAGCGCCCCAAGTGGGTT  
 > gene\_3  
 TGTAGGAGAAGGGCGTAATCT  
 > gene\_4  
 CGTGCTGATTCCACACACNGCAGTAAACGCGG  
 > gene\_5  
 CGTGCTGATTCCACACAGCAGTAAACGCGG  
 ←

エラーの原因は、gene\_4の17番目のポジションがNであることに由来。

```

> hoge <- translate(fasta)          #fastaをアミノ酸配列に翻訳
以下にエラー Call2("DNAStringSet_translate", x, skip_code, dna_
in 'x[[4]]': not a base at pos 17
> names(hoge) <- names(fasta)      #現状では翻訳した結果の$オブジェクト 'hoge' $を確認するだけです
以下にエラー names(hoge) <- names(fasta) : #hogeの中身をfastaに格納
> fasta <- hoge
エラー: オブジェクト 'hoge' がありません
> fasta
A DNAStringSet instance of length 5
  width seq
[1]   21 CGACAGCTCCTCGGCATCCGA
[2]   27 GTCTGCCTCAAGCGCCCCAAGTGGGTT
[3]   21 TGTAGGAGAAGGGCGTAATCT
[4]   30 CGTGCTGATTCCACACNGCAGTAAACGCGG
[5]   30 CGTGCTGATTCCACACAGCAGTAAACGCGG
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=50) #fastaをファイルに保存
> |
  
```

- ・ イントロ | 一般 | 指定した範囲の配列を取得 (last modified 2014/03/08)
- ・ イントロ | 一般 | 指定したID(染色体やdescriptor)の配列を取得 (last modified 2014/03/10)
- ・ イントロ | 一般 | 翻訳配列(translate)を取得 (last modified 2013/06/14)
- ・ イントロ | 一般 | 相補鎖(complement)を取得 (last modified 2013/06/14)
- ・ イントロ | 一般 | 逆相補鎖(reverse complement)を取得 (last modified 2013/06/14)

- ・ イントロ | 一般 | 翻訳配列(translate)を取得

## イントロ | 一般 | 翻訳配列(translate)を取得 NEW

塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。

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

### 1. FASTA

```
in_f <- "sample4.fasta"
out_f <- "hoge5.fasta"

#必要なパッケージをロード
library(Biostrings)

#本番
hoge <- names(fasta)
names(fasta) <- NULL
writeX
```

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

エラーへの対策として、ACGTのみからなる配列を抽出したサブセットを抽出しています。翻訳はそれらのサブセットのみに対して行っているので「文字が塩基ではない」という類のエラーがなっていることがわかります。出力ファイル中の\*は終始コドン(stop codon)を表すようですね。

```
in_f <- "sample4.fasta"          #入力ファイル名を指定してin_fに格納
out_f <- "hoge5.fasta"          #出力ファイル名を指定してout_fに格納

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

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

#前処理(ACGTのみからなる配列を抽出)
hoge <- rowSums(alphabetFrequency(DNAStringSet(fasta))[,1:4]) #A,C,G,T,...の数を配列ご
obj <- (width(fasta) == hoge)                                     #条件を満たすかどうかを判定した結果をobjに格納
fastal <- fasta[obj]                                              #objがTRUEとなる要素のみ抽出した結果をfastalに格
fastal                                         #確認してるだけです

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

4.のエラー対策として考えられるのは、Nを含む配列を除くこと。5. はそれを実現したコードです。

## 5. (multi-)FASTA形式ファイル([sample4.fasta](#))の場合:

エラーへの対策として、ACGTのみからなる配列を抽出したサブセットを抽出セットのみに対して行っているので「文字が塩基ではない」という類のエラーが発生する。FASTAファイル中の\*は終始コドン(アミノ酸を表す記号)を意味します。

[般 | 翻訳配列\(translate\)を取得](#)

```
in_f <- "sample4.fasta"
out_f <- "hoge5.fasta"

#必要なパッケージをロード
library(Biostrings)

#入力ファイルの読み込み
fasta <- readDNAStringSet(fasta)

#前処理(ACGTのみからなる)
hoge <- rowSums(alphabetFrequency(DNAStringSet(fasta))[,1:4]) #ACGTのみを抽出
obj <- (width(fasta) == hoge)
fasta <- fasta[obj]
fasta

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

R Console

```
> #前処理(ACGTのみからなる配列を抽出)
> hoge <- rowSums(alphabetFrequency(DNAStringSet(fasta))[,1:4]) #ACGTのみを抽出
> obj <- (width(fasta) == hoge) #条件を満たすかどうかを$確認
> fasta <- fasta[obj] #objがTRUEとなる要素のみ$確認
> fasta
A DNAStringSet instance of length 4
  width seq
[1] 21 CGACAGCTCCTCGGCATCCGA
[2] 27 GTCTGCCTCAAGCGCCCCAAGTGGGTT
[3] 21 TGTAGGGAGAAGGGCGTAATCT
[4] 30 CGTGCTGATTCCACACAGCAGTAAACGCGG
>
> #本番
> hoge <- translate(fasta)
> names(hoge) <- names(fasta)
> fasta <- hoge
> fasta
A AAStringSet instance of length 4
  width seq
[1] 7 RQLLGIR
[2] 9 VCLKRPKWV
[3] 7 CRRRA*S
[4] 10 RADSTQQ*TR
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=50) #fastaをファイルに保存
> |
```

<pre>names gene_1 gene_2 gene_3 gene_5</pre>	<pre>#fastaをアミノ酸配列に翻訳 #現状では翻訳した結果の\$確認 #hogeの中身をfastaに格納 #確認してます</pre>
<pre>names gene_1 gene_2 gene_3 gene_5</pre>	<pre>#fastaをアミノ酸配列に翻訳 #現状では翻訳した結果の\$確認 #hogeの中身をfastaに格納 #確認してます</pre>

Windowsのヒトは、CTRLとALTキーを押しながらコードの枠内で左クリックすると、全選択できます。Macintoshはよくわかりません。

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

配列中にACGT以外のものが存在するためエラーが出る例です。4番目の配列(つまりgene\_4)の17番目のポジションがNなので妥当です。

```

in_f <- "sample4.fasta"          #入力ファイル名を指定してin_fに格納
out_f <- "hoge4.fasta"           #出力ファイル名を指定してout_fに格納

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

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

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

```

・ イントロ | 一般 | 翻訳配列(translate)を取得

4と5の違いは、5のコード中の黒枠部分が追加されただけ。

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

エラーへの対策として、ACGTのみからなる配列を抽出したサブセットを抽出しています。翻訳はそれらのサブセットのみに対して行っているので「文字が塩基ではない」という類のエラーがなっていることがわかります。出力ファイル中の\*は終始コドン(stop codon)を表すようですね。

```

in_f <- "sample4.fasta"          #入力ファイル名を指定してin_fに格納
out_f <- "hoge5.fasta"           #出力ファイル名を指定してout_fに格納

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

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

#前処理(ACGTのみからなる配列を抽出)
hoge <- rowSums(alphabetFrequency(DNAStringSet(fasta))[,1:4]) #A,C,G,T,...の数を配列ご
obj <- (width(fasta) == hoge)                                     #条件を満たすかどうかを判定した結果をobjに格納
fastal <- fasta[obj]                                              #objがTRUEとなる要素のみ抽出した結果をfastalに格
fasta                                                       #確認してるだけです

```

```

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

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

```

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

・ イントロ | 一般 | 翻訳配列(translate)を取得

エラーへの対策として、ACGTのみからなる配列を抽出したサブセットを抽出しています。翻訳はそれらのサブセットのみに対して行っているので「文字が塩基ではない」という類のエラーがなっていることがわかります。出力ファイル中の\*は終始コドン(\*を意味します)を意味します。

```
in_f <- "sample4.fasta"
out_f <- "hoge5.fasta"

#必要なパッケージをロード
library(Biostrings)

#入力ファイルの読み込み
fasta <- readDNAStringSet(fasta)

#前処理(ACGTのみからなる)
hoge <- rowSums(alphabetFrequency(DNAStringSet(fasta))[,1:4]) #ACGTのみ
obj <- (width(fasta) == hoge)
fasta <- fasta[obj]
fasta

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

R Console

```
> #前処理(ACGTのみからなる配列を抽出)
> hoge <- rowSums(alphabetFrequency(DNAStringSet(fasta))[,1:4]) #ACGTのみ
> obj <- (width(fasta) == hoge) #条件を満たすかどうかを$で確認
> fasta <- fasta[obj] #objがTRUEとなる要素のみ$で確認
> fasta
A DNAStringSet instance of length 4
  width seq
[1] 21 CGACAGCTCCTCGGCATCCGA
[2] 27 GTCTGCCTCAAGCGCCCCAAGTGGGTT
[3] 21 TGTAGGGAGAAGGGCGTAATCT
[4] 30 CGTGCTGATTCCACACAGCAGTAAACGCGG
  names
  gene_1
  gene_2
  gene_3
  gene_5

>
> #本番
> hoge <- translate(fasta)
> names(hoge) <- names(fasta)
> fasta <- hoge
> fasta
A AAStringSet instance of length 4
  width seq
[1] 7 RQLLGIR
[2] 9 VCLKRPKWV
[3] 7 CRRRA*S
[4] 10 RADSTQQ*TR
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=50) #fasta形式で保存
>
```

「基本的な利用法4と5」の中で示した条件判定を内部的に利用していることが分かる。(Rで)塩基配列解析中では、条件判定を行って得られた論理値ベクトルをobjというオブジェクト名で統一的に取り扱っています。

# 実行結果

入力 : sample4.fasta

```
>gene_1↓
CGACAGCTCCTCGGCATCCGA↓
>gene_2↓
GTCTGCTCAAGCGCCCCAAGTGGGTT↓
>gene_3↓
TGTAGGAGAAGGGCGTAATCT↓
>gene_4↓
CGTGCTGATTCCACACAGCAGTAAACCGCGG↓
>gene_5↓
CGTGCTGATTCCACACAGCAGTAAACCGCGG↓
←
```

出力 : hoge5.fasta

```
>gene_1↓
RQLLGIR↓
>gene_2↓
VCLKRPKWW↓
>gene_3↓
CRRRA*S↓
>gene_5↓
RADSTQQ*TR↓
←
```

UAU (Tyr/Y) チロシン
UAC (Tyr/Y) チロシン
<b>UAA Ochre (終止)</b>
UAG Amber (終止)

CAU (His/H) ヒスチジン
CAC (His/H) ヒスチジン
CAA (Gln/Q) グルタミン
CAG (Gln/Q) グルタミン

エラーの原因であった17番目のポジションにNを含むgene\_4が除かれて、うまく翻訳配列を出力できていることが分かる。尚、アスタリスク(\*)は終始コドンを表すようです。ここで用いたテクニックは、ACGTのみからなる塩基配列のフィルタリングと翻訳の2つ。もちろん前者のフィルタリングのみを行うことも可能。

# (Rで)塩基配列解析

~NGS、RNA-seq、ゲノム、トランскриプトーム、正規化、発現変動、統計、モデル、バイオインフォマティクス~  
(last modified 2014/08/03, since 2010)

処理 | フィルタリング | ACGTのみからなる配列を抽出

ACGTのみからなる塩基配列のフィルタリングを行う場合。

## What's new?

- このウェブペ 1. Rのインス
- 2014年10月 フォマティクス
- 門田幸二 著
- 日本乳酸菌
- 参考資料(詳)
- ・ イントロ | 一般 | [任意の長さの可能な全ての塩基配列を作成](#) (last modified 2013/06/14)
- ・ イントロ | 一般 | [任意の位置の塩基を置換](#) (last modified 2013/09/12)
- ・ イントロ | 一般 | [指定した範囲の配列を取得](#) (last modified 2014/03/08)
- ・ イントロ | 一般 | [指定したID\(染色体やdescription\)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](#) (last modified 2014/08/04) NEW
- ・ イントロ | 一般 | [相補鎖\(complement\)を取得](#) (last modified 2013/06/14)
- ・ イントロ | 一般 | [逆相補鎖\(reverse complement\)を取得](#) (last modified 2013/06/14)
- ・ イントロ | 一般 | [逆鎖\(reverse\)を取得](#) (last modified 2013/06/14)
- ・ イントロ | 一般 | [2連続塩基の出現頻度情報を取得](#) (last modified 2014/07/18) NEW
- ・ イントロ | 一般 | [3連続塩基の出現頻度情報を取得](#) (last modified 2013/06/14)
- ・ イントロ | 一般 | [任意の長さの連続塩基の出現頻度情報を取得](#) (last modified 2013/06/14)
- ・ 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](#)
- ・ イントロ | 一般 | 配列取得
- ・ イントロ | NGS | 様々なプラ
- ・ イントロ | NGS | qPCRやmi
- ・ 前処理 | クオリティチェック | [grqc](#) (last modified 2014/07/17) NEW
- ・ 前処理 | クオリティチェック | [PHREDスコアに変換](#) (last modified 2013/06/18)
- ・ 前処理 | クオリティチェック | [配列長分布を調べる](#) (last modified 2013/06/18)
- ・ 前処理 | フィルタリング | [PHREDスコアが低い塩基をNに置換](#) (last modified 2014/03/03)
- ・ 前処理 | フィルタリング | [PHREDスコアが低い配列\(リード\)を除去](#) (last modified 2014/03/03)
- ・ 前処理 | フィルタリング | [ACGTのみからなる配列を抽出](#) (last modified 2014/08/04) NEW
- ・ 前処理 | フィルタリング | [ACGT以外のcharacter"-,"をNに変換](#) (last modified 2013/06/18)
- ・ 前処理 | フィルタリング | [ACGT以外の文字数が閾値以下の配列を抽出](#) (last modified 2013/09/27)
- ・ 前処理 | フィルタリング | [重複のない配列セットを作成](#) (last modified 2013/06/18)
- ・ 前処理 | フィルタリング | [指定した長さ以上の配列を抽出](#) (last modified 2014/02/07)
- ・ 前処理 | フィルタリング | [任意のリード\(サブセット\)を抽出](#) (last modified 2014/07/17) NEW
- ・ 前処理 | フィルタリング | [指定した長さの範囲の配列を抽出](#) (last modified 2013/06/18)
- ・ 前処理 | フィルタリング | [任意のIDを含む配列を抽出](#) (last modified 2013/06/18)
- ・ 前処理 | フィルタリング |  [Illuminaのpass filtering](#) (last modified 2013/06/19)
- ・ 前処理 | フィルタリング | [GFF/GTF形式ファイル](#) (last modified 2013/10/10)
- ・ 前処理 | フィルタリング | 組合せ | [ACGTのみ & 指定した長さの範囲の配列](#) (last modified 2014/06/18)

# 前処理 | フィルタリング | ACGTのみからなる配列を抽出 NEW

FASTQファイルやFASTAファイルを読み込んで"N"などの文字を含まず、ACGTのみからなる配列を含むコンティグのみ抽出して、(multi-)FASTA形式ファイルに出力するやり方を示します。

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

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

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

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

#必要なパッケージをロード

## 6. (multi-)FASTA形式ファイル([sample4.fasta](#))の場合:

gene\_4が消えていることが分かります。

```
library(Biostrings)               #必要なパッケージをロード
in_f <- "sample4.fasta"           #入力ファイル名を指定してin_fに格納
out_f <- "hoge6.fasta"            #出力ファイル名を指定してout_fに格納
#本番
hoge <- rowSum
obj <- (width(fasta) == hoge)
fasta <- fasta[obj]
#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
```

gene\_4が消えていることが分かります。

```
library(Biostrings)               #必要なパッケージをロード
in_f <- "sample4.fasta"           #入力ファイル名を指定してin_fに格納
out_f <- "hoge6.fasta"            #出力ファイル名を指定してout_fに格納
#パッケージの読み込み
#本番
hoge <- rowSums(alphabetFrequency(DNAStringSet(fasta))[,1:4]) #A,C,G,T,..の数を配列ごとにカウント
obj <- (width(fasta) == hoge)      #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
#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
```

- 前処理 | フィルタリング | ACGTのみからなる配列を抽出

ACGTのみからなる塩基配列のフィルタリングを行う場合。翻訳配列を得る部分のコードがないだけです。

# Tips

R Gui (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

```
> obj <- (width(fasta) == hoge)          # 条件を満たすかどうか
> fasta <- fasta[obj]                    # objがTRUEとなる要素
> fasta
A DNAStringSet instance of length 4
  width seq
[1] 21 CGACAGCTCCTCGGCATCCGA
[2] 27 GTCTGCCTCAAGCGCCCCAAGTGGGTT
[3] 21 TGTAGGAGAAAGGGCGTAATCT
[4] 30 CGTGCTGATTCCACACAGCAGTAAACGCGG
>
> # ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=$
>
> ?
+ + +
+ |
```

①「?」のみを打ち込んでリターン。  
②「+」は引き続いて打ち込まれるはずのコマンド入力待ち状態。例えば通常は「?log」などと?に引き続いて関数名が打ち込まれるはず。

# Tips

R Gui (64-bit)

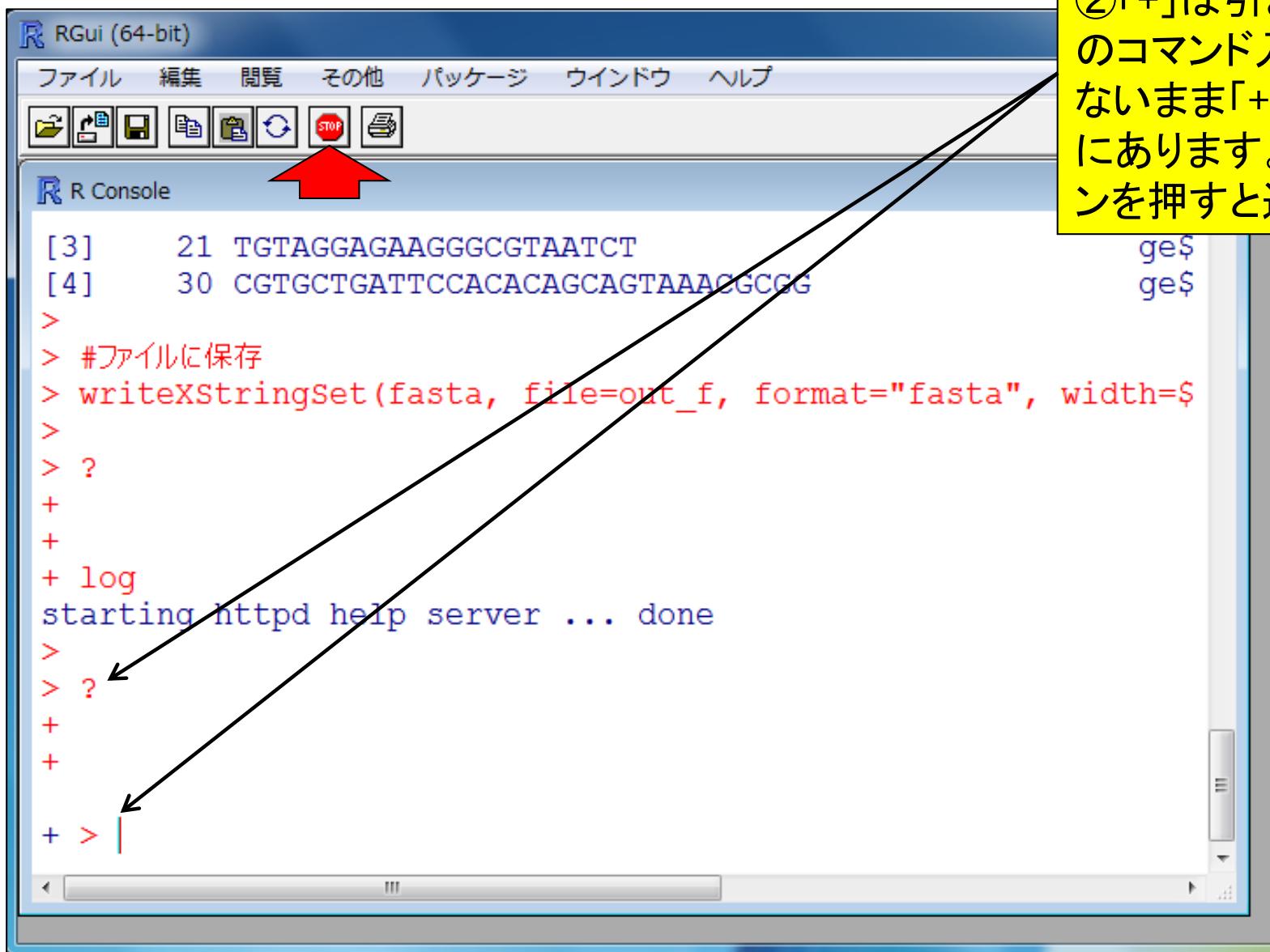
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ

R Console

```
> fasta
A DNAStringSet instance of length 4
  width seq
[1] 21 CGACAGCTCCTCGGCATCCGA
[2] 27 GTCTGCCTCAAGCGCCCCAAGTGGGTT
[3] 21 TGTAGGAGAAAGGGCGTAATCT
[4] 30 CGTGCTGATTCCACACACAGCAGTAAACGCGG
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=$
>
> ?
+ 
+ 
+ log
starting httpd help server ... done
> |
```

- ①「?」のみを打ち込んでリターン。
- ②「+」は引き続いて打ち込まれるはずのコマンド入力待ち状態。例えば通常は「?log」などと?に引き続いて関数名が打ち込まれるはず。
- ③例えばlogと打ち込むとhtmlマニュアルが開き、通常のコマンド入力待ち状態となる。

# Tips



R GUI (64-bit)

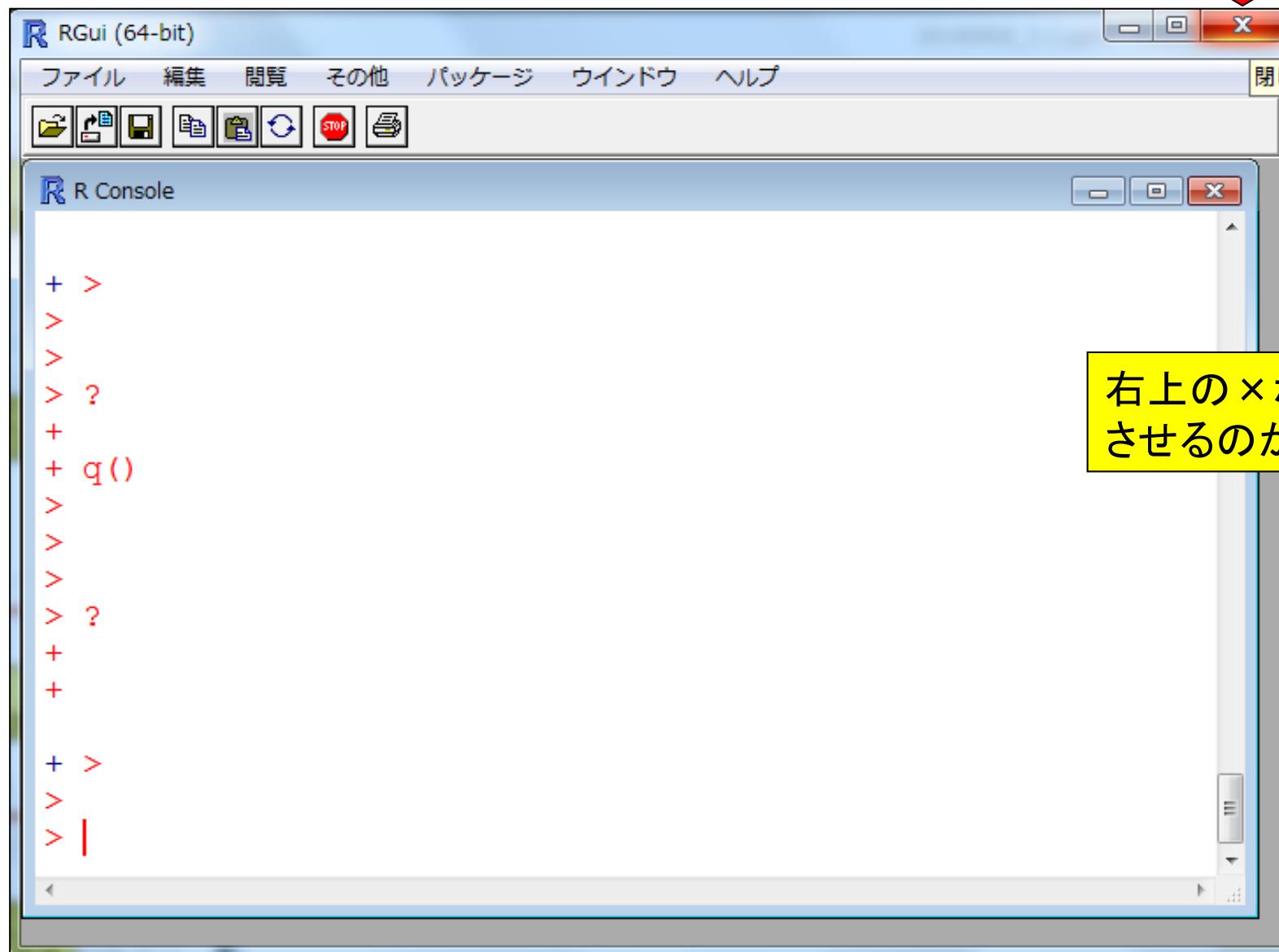
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ

R Console

```
[3] 21 TGTAGGAGAAGGGCGTAATCT
[4] 30 CGTGCTGATTCCACACAGCAGTAAACGCCG
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=$
>
> ?
+ +
+ log
starting httpd help server ... done
>
> ?
+ +
+
+ > |
```

①「?」のみを打ち込んでリターン。  
②「+」は引き続いて打ち込まれるはずのコマンド入力待ち状態。よくわからないまま「+」となってしまうことがたまにあります。そういう場合はSTOPボタンを押すと通常の">"状態になります。

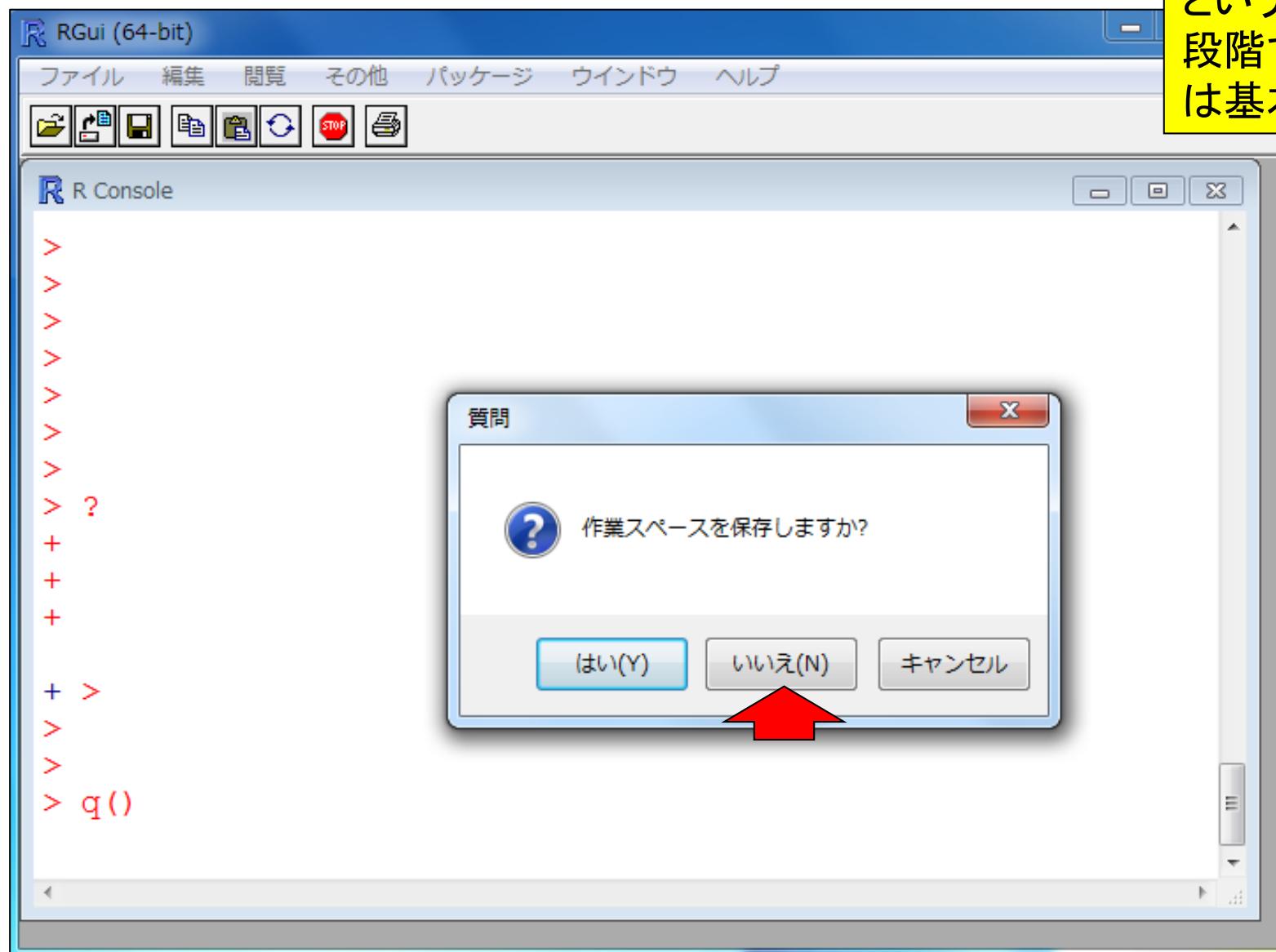
# Tips(Rの終了)



右上の×ボタンを押してRを終了  
させるのが一番手っ取り早いです

# Tips(Rの終了)

「作業スペースを保存しますか?」  
という問い合わせの意味がわからない  
段階では「いいえ」でよい。(門田  
は基本「いいえ」で終了させる)



# Contents

## ■ 3-2. R 基礎2、2014/09/08 13:15–14:45、初級、実習

### □ (Rで)塩基配列解析の基本的な利用法(翻訳配列の取得を例に)

- 入力ファイル取得、作業ディレクトリの変更、本番(基本はコピペ)、出力結果の確認
- 1つの項目に多数の例題(異なる入力ファイル、エラーへの対処例)

### □ 行列形式ファイルの解析基礎(アノテーションファイルを例に)

- 例題をテンプレートとして任意の解析を行う基本手順
- ありがちなミスとエラーメッセージ
- 入力ファイルの最後の改行の有無
- プログラム内部の説明(行列演算の基礎)

### □ Tips

- 集合演算:union, intersect, setdiff
- その他:sort, table, is.element, toupper, tolower



# タブ区切りテキストファイルからの情報抽出

入力1: アノテーションファイル(annotation.txt)

	A	B	C	D
1	genename	accession	description	subcellular_location
2	gene1	hoge01	plasma_mem	nuclear
3	gene2	hoge02	hohinu	membrane
4	gene3	hoge03	agribio	endoplasmic
5	gene4	hoge04	genesis	endoplasmic
6	gene5	hoge05	kamo	membrane
7	gene6	hoge06	netteba	humei
8	gene7	hoge07	tebasaki	nuclear
9	gene8	hoge08	biiru	nuclear
10	gene9	hoge09	nihonshu	nuclear
11	gene10	hoge10	agene1	membrane
12	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2: リストファイル(genelist1.txt)

	A
1	gene1
2	gene7
3	gene9

目的: アノテーションファイル(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。  
解析例:

- ・発現に差のある遺伝子のみのアノテーション情報抽出
- ・特定のGene Ontology termに含まれるもののみ抽出

# (Rで)塩基配列解析

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

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

2つの入力ファイル(annotation.txtとgenelist1.txt)をhogeフォルダにダウンロードして実行してみましょう。

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

## イントロ | 一般 | 任意のキーワードを含む行を抽出(基礎)

例えばタブ区切りテキストファイルが手元にあり、この中からリストファイル中の文字列を含む行を抽出するやり方を示します。Linux (UNIX)のgrepコマンドのようなものであり、perlのハッシュのようなものです。

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

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

```
in_f1 <- "annotation.txt"          #入力ファイル名を指定してin_f1に格納(アノテーション
in_f2 <- "genelist1.txt"           #入力ファイル名を指定してin_f2に格納(リストファイル)
out_f <- "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に保存
```

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out_fで指定したファイル名で保存
```

入力1: annotation.txt

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

入力2: genelist1.txt

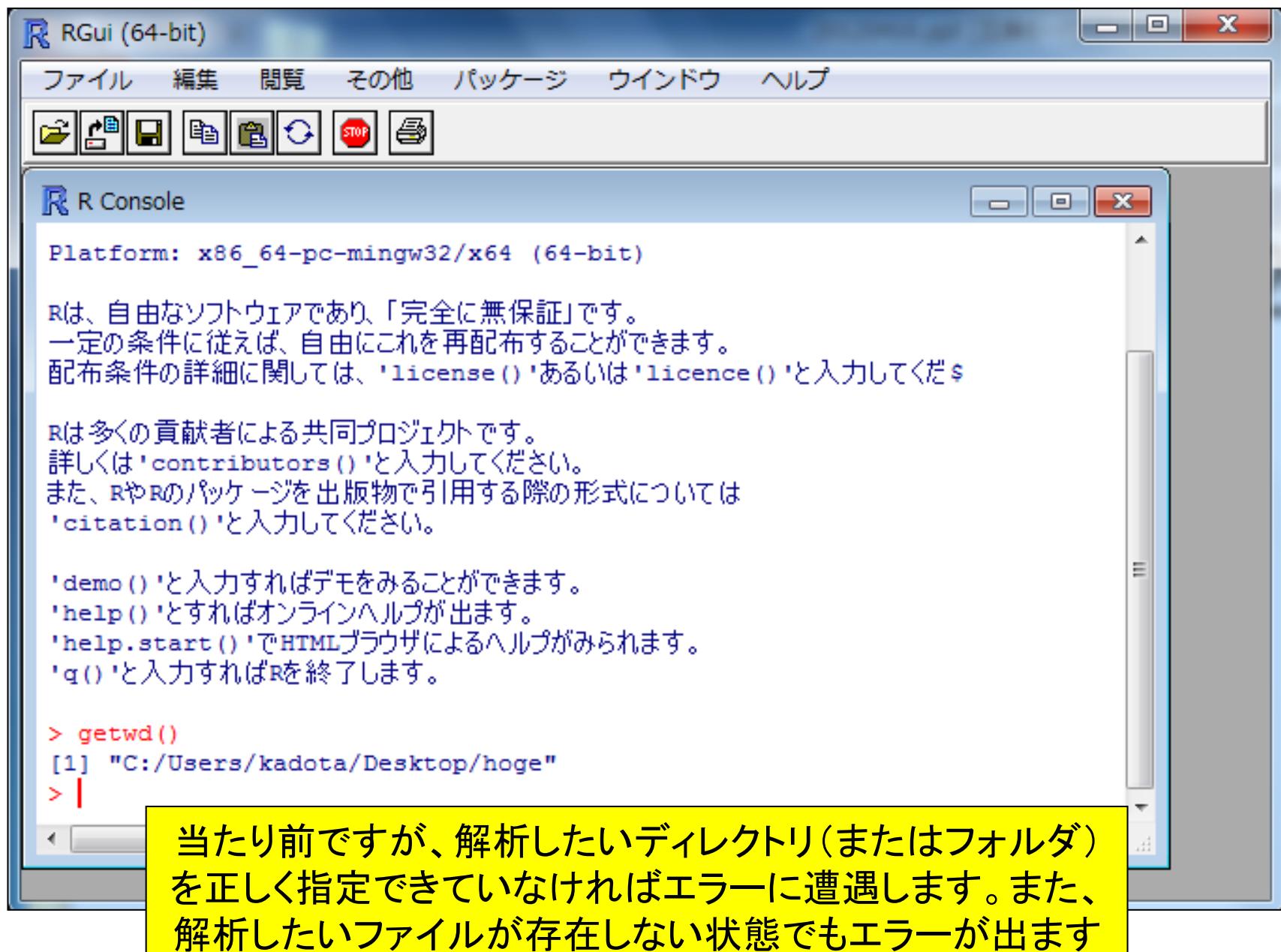
	A
1	gene1
2	gene7
3	gene9

出力: hoge1.txt

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

デスクトップ上にhogeという名前のフォルダがあり、フォルダ中に annotation.txt と genelist1.txt が存在するという前提です。メモ帳で開くと改行コードが崩れている場合は、ワードパッドなどで開くとよい

# 作業ディレクトリの変更と確認



The screenshot shows the R GUI interface. The main window title is "R Gui (64-bit)". Below it is the "R Console" window. The console output is as follows:

```
Platform: x86_64-pc-mingw32/x64 (64-bit)

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

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

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

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

A yellow callout box highlights the text: "当たり前ですが、解析したいディレクトリ(またはフォルダ)を正しく指定できていなければエラーに遭遇します。また、解析したいファイルが存在しない状態でもエラーが出ます" (It is common sense that if you do not specify the directory correctly, an error will occur. Additionally, if the file you want to parse does not exist, an error will occur).



# 基本はコピペ

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

例えばタブ区切りテキストファイルが手元にあり、この中からリストファイル中の文字列を含む行を抽出するやり方(UNIX)のgrepコマンドのようなものであり、perlのハッシュのようなものです。

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

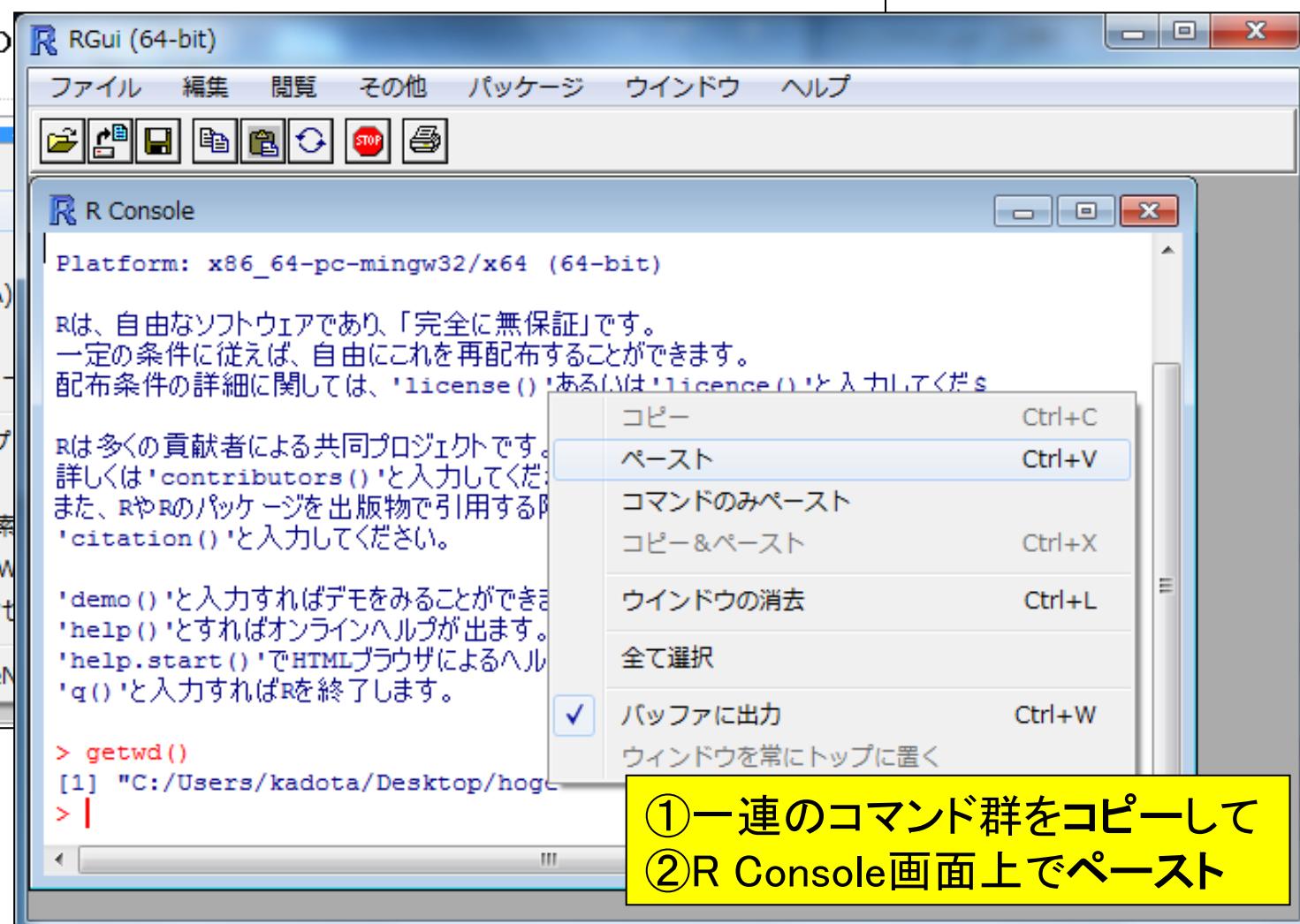
- 目的のタブ区切りテキストファイル(annotation.txt)中の行全体を出力したい場合:

```
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, he
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=
```



# 実行結果

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

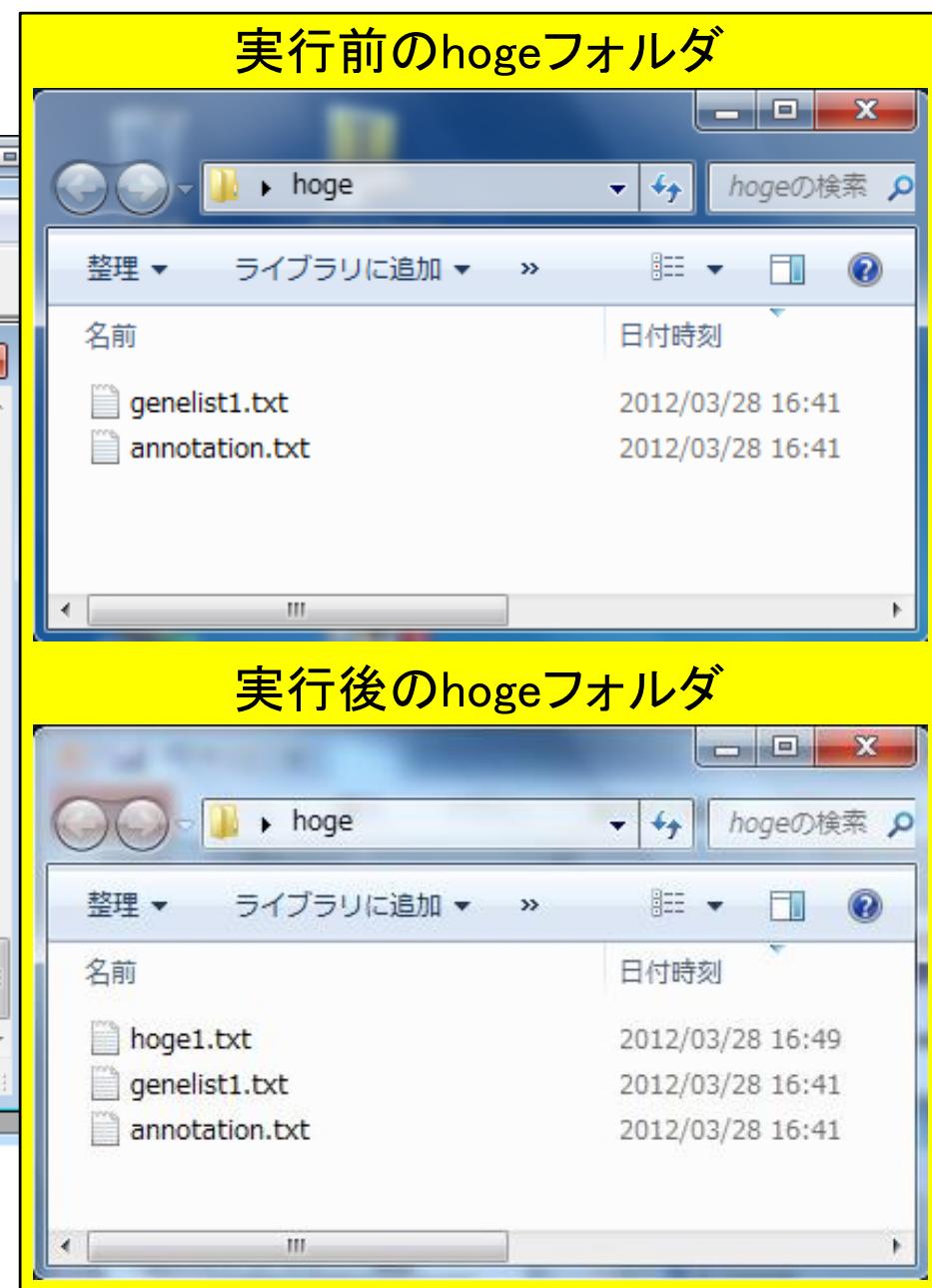
R Console

```

> in_f2 <- "genelist1.txt"
> out_f <- "hoge1.txt"
> param <- 1
>
> #ファイルの読み込み
> data <- read.table(in_f1, header=TRUE, sep="\t", quote="")
> keywords <- readLines(in_f2)
> dim(data)
[1] 11  4
>
> #本番
> obj <- is.element(as.character(data[,param]), keywords)
> out <- data[obj,]
> dim(out)
[1] 3  4
> write.table(out, out_f, sep="\t", append=F, quote=F, row.names=F)$$
>
> |
```

**hoge1.txt**

	A	B	C	D
1	genename	accession	description	subcellular_location
2	gene1	hoge01	plasma_mem	nuclear
3	gene7	hoge07	tebasaki	nuclear
4	gene9	hoge09	nihonshu	nuclear



# 色についての説明

## (Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランскриプトーム、正規化、発現変動、統計、モデル、バイオインフォマティクス～  
(last modified 2014/08/22, since 2010)

### What's new?

- このウェブページはフリーソフト R と必要なパッケージをインストール済みである前提で記述しています。初心者は、1. [Rのインストールと起動](#) および 2. [基本的な利用法](#) で自習してください。(2014/07/21)
- 2014年10月04日に [HPCIワークショップ「医療とビッグデータ解析」\(9:00-9:20\)](#) に引き続いで [中級者向けバイオインフォマティクス入門講習会@仙台国際センター\(10:50-12:20\)](#) で話します。興味ある方はどうぞ。(2014/07/23)
- 門田幸二著 [シリーズ Useful R 第7巻トランскриプトーム解析](#)刊行(共立出版)
- バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ) | [速習コース](#)で利用する計算機環境構築する一通りの手順を公開しました。(2014/08/11) **NEW**
- 日本乳酸菌学会誌の NGS 関連連載の [第1回分 PDF](#) を公開しました。関連項目は [こちら](#)。(2014/08/03) **NEW**
- 参考資料(講義、講習会、本など)の項目を更新しました。(2014/08/19) **NEW**

- [はじめに](#) (modified 2014/01/30)
- 参考資料([講義、講習会、本など](#)) (last modified 2014/08/19) **NEW**
- 過去のお知らせ (last modified 2014/08/03) **NEW**
- [Rのインストールと起動](#) (last modified 2014/07/31) **NEW**
- [基本的な利用法](#) (last modified 2014/07/20)
- [サンプルデータ](#) (last modified 2014/07/17)
- バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ)

このページ内で用いる色についての説明:

### コメント

特にやらなくてもいいコマンド  
プログラム実行時に目的に応じて変更すべき箇所

このページ内で用いる色についての説明:

## コメント

特にやらないでもいいコマンド

プログラム実行時に目的に応じて変更すべき箇所

# 色についての説明

```
in_f1 <- "annotation.txt"  
in_f2 <- "genelist1.txt"  
out_f <- "hoge1.txt"  
param <- 1  
  
#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)  
#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out_fで指定したファイル名で保存
```

上記は1列目でキーワード検索する場合

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

4列目でキーワード検索したいときは?



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

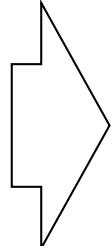
# 解答例

- 目的のキーワードリストを含むファイルを作成し(例:list.txt)
- 該当箇所を変更し、R Console画面上でコピペ

```
run1.txt - メモ帳
ファイル(F) 編集(E) 書式(O) 表
in_f1 <- "annotation.txt"
in_f2 <- "genelist1.txt"
out_f <- "hogel.txt"
param <- 1

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

#本番
obj <- is.element(as.character(data[, param]), keywords)
out <- data[obj,]
dim(out)
write.table(out, out_f, sep="\t", append=F, quote=F, row.names=F)
```

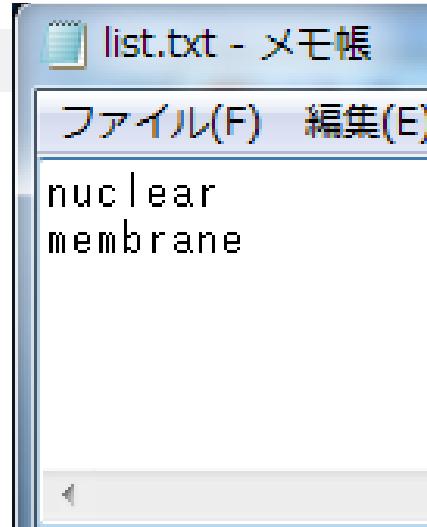


```
run1.txt - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
in_f1 <- "annotation.txt"
in_f2 <- "list.txt"
out_f <- "hogel.txt"
param <- 4

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

#本番
obj <- is.element(as.character(data[, param]), keywords)
out <- data[obj,]
dim(out)
write.table(out, out_f, sep="\t", append=F, quote=F, row.names=F)
```

一連の作業手順を記述したスクリプトを  
一つのファイルとして保存することをお勧め



# Contents

## ■ 3-2. R 基礎2、2014/09/08 13:15–14:45、初級、実習

### □ (Rで)塩基配列解析の基本的な利用法(翻訳配列の取得を例に)

- 入力ファイル取得、作業ディレクトリの変更、本番(基本はコピペ)、出力結果の確認
- 1つの項目に多数の例題(異なる入力ファイル、エラーへの対処例)

### □ 行列形式ファイルの解析基礎(アノテーションファイルを例に)

- 例題をテンプレートとして任意の解析を行う基本手順
- ありがちなミスとエラーメッセージ
- 入力ファイルの最後の改行の有無
- プログラム内部の説明(行列演算の基礎)

### □ Tips

- 集合演算:union, intersect, setdiff
- その他:sort, table, is.element, toupper, tolower



# ありがちなミス1

R Console

```
> in_f1 <- "annotation.txt"
> in_f2 <- "genelist1.txt"
> out_f <- "hogel.txt"
> param <- 1
>
> #ファイルの読み込み
> data <- read.table(in_f1, header=TRUE, sep="\t", quote="")
以下にエラー file(file, "rt") : コネクションを開くことができません
追加情報: 警告メッセージ:
```

```
In file(file, "rt") :
  ファイル 'annotation.txt' を開くことができません: No such file or directory$
```

```
> keywords <- readLines(in_f2) #入力파$
以下にエラー file(con, "r") : コネクションを開くことができません
追加情報: 警告メッセージ:
```

```
In file(con, "r") :
  ファイル 'genelist1.txt' を開くことができません: No such file or directory
```

```
> dim(data)
```

```
NULL
```

```
>
```

```
> #本番
```

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

```
以下にエラー data[, param] :
  'closure' 型のオブジェクトは部分代入可能ではありません
```

```
> out <- data[obj,]
```

```
エラー: オブジェクト 'obj' がありません
```

```
> dim(out)
```

```
エラー: オブジェクト 'out' がありません
```

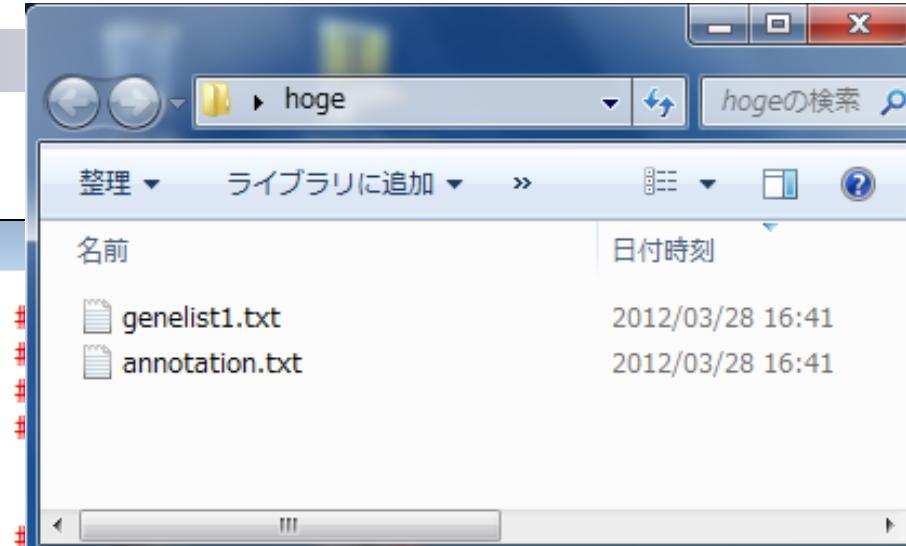
```
> write.table(out, out_f, sep="\t", append=F, quote=F, row.names=F) #outの中身$
```

```
以下にエラー is.data.frame(x) : オブジェクト 'out' がありません
```

```
>
```

```
> getwd()
```

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

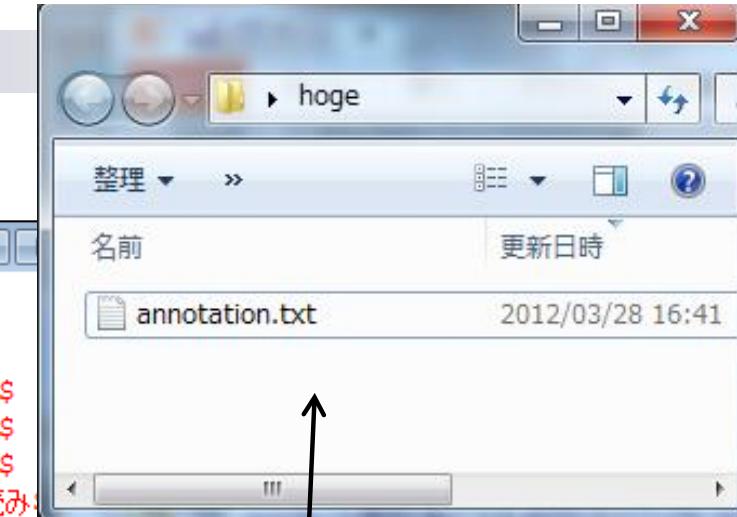


作業ディレクトリの変更を忘れて  
いるため、in\_f1で指定した最初の  
ファイルの読み込み段階でエラー  
が出る。つまり、実際に行った  
フォルダ中にはannotation.txtとい  
うファイルは存在しないということ。

# ありがちなミス2

```
R Console

> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> in_f1 <- "annotation.txt"
> in_f2 <- "genelist1.txt"
> out_f <- "hogel.txt"
> param <- 1
>
> #ファイルの読み込み
> data <- read.table(in_f1, header=TRUE, sep="\t", quote="")
> keywords <- readLines(in_f2)
以下にエラー file(con, "r") : コネクションを開くことができません
追加情報: 警告メッセージ:
In file(con, "r") :
  ファイル 'genelist1.txt' を開くことができません: No such file or directory
> dim(data)
[1] 11  4
>
> #本番
> obj <- is.element(as.character(data[,param]), keywords)
以下にエラー match(el, set, 0L) : オブジェクト 'keywords' がありません
> out <- data[obj,]
以下にエラー '[.data.frame` (data, obj, ) : オブジェクト 'obj' がありません
> dim(out)
エラー: オブジェクト 'out' がありません
> write.table(out, out_f, sep="\t", append=F, quote=F, row.names=F)
以下にエラー is.data.frame(x) : オブジェクト 'out' がありません
>
```



#入力ファイル\$  
#入力ファイル\$

#出力ファイル\$

#in\_f1で読み

#入力ファイル\$

#入力ファイル\$

#オブジェクト\$

#in\_f1で読み\$

#行列dataから\$

必要な入力ファイルが作業ディレクトリ  
中に存在しない。この場合、in\_f2で指  
定したgenelist1.txtが存在しないため、  
その読み込み段階でエラーが出で  
いる。それゆえ、その情報を用いてい  
るコマンド部分でエラーが出でている。

# ありがちなミス3

The screenshot shows the RStudio interface with two windows open. On the left is the R Console window, and on the right is a Microsoft Excel spreadsheet and a Notepad window.

**R Console (Left Window):**

```
> in_f1 <- "annotation.txt"
> in_f2 <- "list.txt"
> out_f <- "hogel.txt"
> param <- 4
>
> #ファイルの読み込み
> data <- read.table(in_f1, header=TRUE, sep="\t", quote="")
> keywords <- readLines(in_f2)
> dim(data)
[1] 11  4
>
> #本番
> obj <- is.element(as.character(data[,param]), keywords)
> out <- data[obj,]
> dim(out)
[1] 7 4
> write.table(out, out_f, sep="\t", append=F, quote=F, row.names=
  以下にエラー file(file, ifelse	append, "a", "w")) :
  コネクションを開くことができません
  追加情報: 警告メッセージ:
In file(file, ifelse-append, "a", "w")) :
  ファイル 'hogel.txt' を開くことができません: Permission denied
> |
```

**#入** (Red text marker)

**hogel.txt - Microsoft Excel (Right Window):**

	A	B	C	D	E	F
1	genename	accession	description	subcellular_location		
2	gene1	hoge01	plasma_mer	nuclear		
3	gene7	hoge07	tebasaki	nuclear		
4	gene9	hoge09	nihonshu	nuclear		
5						
6						

**run1.txt - メモ帳 (Bottom Window):**

```
in_f1 <- "annotation.txt"
in_f2 <- "list.txt"
out_f <- "hogel.txt"
param <- 4

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

#本番
obj <- is.element(as.character(data[,param]), keywords)
out <- data[obj,]
dim(out)
write.table(out, out_f, sep="\t", append=F, quote=F, row.names=
```

出力予定のファイル名と同じものをエクセルなど別のプログラムで開いているため、最後のwrite.table関数のところでエラーが出る。対処法は、出力ファイル名を変更するか、開いている別のプログラムを閉じる。

# ありがちなミス4

```
R run1.txt - メモ帳
R
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
in_f1 <- "annotation.txt"          #入力ファイル名(目的のタブ区切りテキストファイル)を
in_f2 <- "list.txt"                #入力ファイル名(キーワードなどのリストファイル)を指定
out_f <- "hogel.txt"               #出力ファイル名を指定
param <- 4                          #in_f1で読み込む目的のファイルの何列目のデータに対し
#ファイルの読み込み
> data <- read.table(in_f1, header=TRUE, sep="\t", quote="")
> keywords <- readLines(in_f2)      #入力ファイル(目的のファイル)を読み込んでdataに格納
> dim(data)                        #入力ファイル(リストファイル)を読み込んでkeywordsに
                                    #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)    #in_f1で読み込んだファイル中の(param)列目の文字列ベ
> out <- data[obj,]                      #行列dataからobjがTRUEとなる行のみを抽出した結果をo
> dim(out)                            #オブジェクトoutの行数と列数を表示
> write.table(out, out_f, sep="\t", append=F, quote=F, row.names=F) #outの中身をout_fで指定したファイル名で保存。
>
>
>
>
>
> keywords <- readLines(in_f2)
> dim(data)
[1] 11  4
>
> #本番
> obj <- is.element(as.character(d
> out <- data[obj,]
> dim(out)
[1] 7  4
> write.table(out, out_f, sep="\t", append=F, quote=F, row.names=F) #outの中身をout_fで指定したファイル名で保存。
```

実行スクリプトをコピーする際、最後の行のところで改行を含ませずにR Console画面上でペーストしたため、最後のコマンドが実行されない(出力ファイルが生成されない)。これも比較的ありがちなパターンです。コピペ後に無意識にリターンキーを押すことを心がけるだけでもよいでしょう。

# 改行を入れておいたほうが警告が出ない

```

R Console
> in_f1 <- "annotation.txt"
> in_f2 <- "list.txt" ← nuclear↓
> out_f <- "hoge2.txt" ← membrane↓
> param <- 4
>
> #入力ファイルの読み込み
> data <- read.table(in_f1, header=TRUE, sep="$")
> keywords <- readLines(in_f2)           #in_f$#
> dim(data)                            #オブ$#
[1] 11  4
>
> #本番
> obj <- is.element(as.character(data[,param]))$#
> out <- data[obj,]                   #obj$#
> dim(out)                            #オブ$#
[1] 7  4
>
> #ファイルに保存
> write.table(out, out_f, sep="\t", append=F, $)
>

```

```

R Console
> in_f1 <- "annotation.txt"
> in_f2 <- "list.txt" ← nuclear↓
> out_f <- "hoge2.txt" ← membrane←
> param <- 4
>
> #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$#
> keywords <- readLines(in_f2)           #in_f2で指定したファイルの読$#
警告メッセージ:
In readLines(in_f2) : 'list.txt' で不完全な最終行が見つかりました
> dim(data)                            #オブジェクトdataの行数と列数$#
[1] 11  4
>
> #本番
> obj <- is.element(as.character(data[,param]), keywords)#条件を満たす$#
> out <- data[obj,]                   #objがTRUEとなる行のみ抽出し$#
> dim(out)                            #オブジェクトoutの行数と列数$#
[1] 7  4
>
> #ファイルに保存
> write.table(out, out_f, sep="\t", append=F, quote=F, row.names=F)#ou$#
>

```

list.txtファイル作成時に、membraneと打った後に改行を入れた場合(左)と入れない場合(右)の挙動の違いを把握し、後学のために警告メッセージの意味を理解しておくとよい。この場合は結果には影響していないことがわかる。Rは警告メッセージ後の記述内容が比較的分かりやすいのでよく読むべし

# Contents

## ■ 3-2. R 基礎2、2014/09/08 13:15–14:45、初級、実習

### □ (Rで)塩基配列解析の基本的な利用法(翻訳配列の取得を例に)

- 入力ファイル取得、作業ディレクトリの変更、本番(基本はコピペ)、出力結果の確認
- 1つの項目に多数の例題(異なる入力ファイル、エラーへの対処例)

### □ 行列形式ファイルの解析基礎(アノテーションファイルを例に)

- 例題をテンプレートとして任意の解析を行う基本手順
- ありがちなミスとエラーメッセージ
- 入力ファイルの最後の改行の有無
- プログラム内部の説明(行列演算の基礎)

### □ Tips

- 集合演算:union, intersect, setdiff
- その他:sort, table, is.element, toupper, tolower



# 読み込み

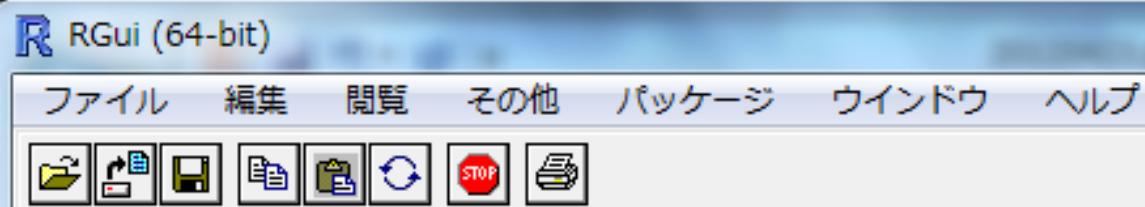
```
in_f1 <- "annotation.txt"  
in_f2 <- "genelist1.txt"  
out_f <- "hoge1.txt"  
param <- 1  
  
#入力ファイルの読み込み①  
data <- read.table(in_f1, header=TRUE, sep="\t", quote="")#in_f1で指定した  
②
```

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

③

- ①in\_f1で指定したファイルを読み込め
- ②読み込むファイルの最初の行はヘッダー部分です
- ③ファイルの区切り文字はタブです

# 行列data



```
R Gui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウィンドウ ヘルプ
[Icons]
R Console
>
> #ファイルの読み込み
> data <- read.table(in_f1, header=TRUE, sep="\t", qu
>
> 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  iyaaaaa          endoplasmic
> |
```

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

入力ファイルの中身を正しく  
読み込んでいることがわかる

```

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
out <- data[obj,]
dim(out)

```

### #ファイルに保存

```
write.table(out,
```

R Console

```

> data
  genename accession de
  1 gene1    hoge01 plasma_mem      nuclear
  2 gene2    hoge02 hohinu          membrane
  3 gene3    hoge03 agribio         endoplasmic
  4 gene4    hoge04 genesis          -
  5 gene5    hoge05 kam             -
  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 iyaaaaa       endoplasmic

```

オブジェクトdataの行数と列数は11と4。  
webpage中の表記が灰色なのは、特に  
やらなくてもいいコマンドだから。

annotation.txt

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

# 行列の要素へのアクセス

R Console

```

10 gene10 hoge10 agene1
11 gene11 hoge11 iyaaaa
> dim(data)
[1] 11 4
> data[6,4] ←
[1] humei
Levels: endoplasmic humei membrane nuclear
> data[2,]
  genename accession description subcellular_loc
2   gene2     hoge02     hohinu      membrane
> data[,2]
[1] hoge01 hoge02 hoge03 hoge04 hoge05 hoge06 hoge07
[8] hoge08 hoge09 hoge10 hoge11
11 Levels: hoge01 hoge02 hoge03 hoge04 hoge05 ... hoge11
> data[,param]
[1] gene1 gene2 gene3 gene4 gene5 gene6 gene7
[8] gene8 gene9 gene10 gene11
11 Levels: gene1 gene10 gene11 gene2 gene3 ... gene9
> |

```

annotation.txt

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

```

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

```

paramには1という数値  
が代入されていたから

# やりたかったことをおさらい

```
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)
keywords <- readLines(in_f2)
dim(data)
```

#本番

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

#ファイルに保存

```
write.table(out, out_f,
```

#入力ファイル名を指定してin\_f1に格納  
#入力ファイル名を指定してin\_f2に格納  
#出力ファイル名を指定してout\_fに格納

R Console

```
> 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
 11 gene11   hoge11

> obj
[1] TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE
[9] TRUE FALSE FALSE
> data[obj,]
  genename accession description subcellular_location
  1 gene1    hoge01  plasma_mem          nuclear
  7 gene7    hoge07  tebasaki          nuclear
  9 gene9    hoge09  nihonshu          nuclear
> |
```

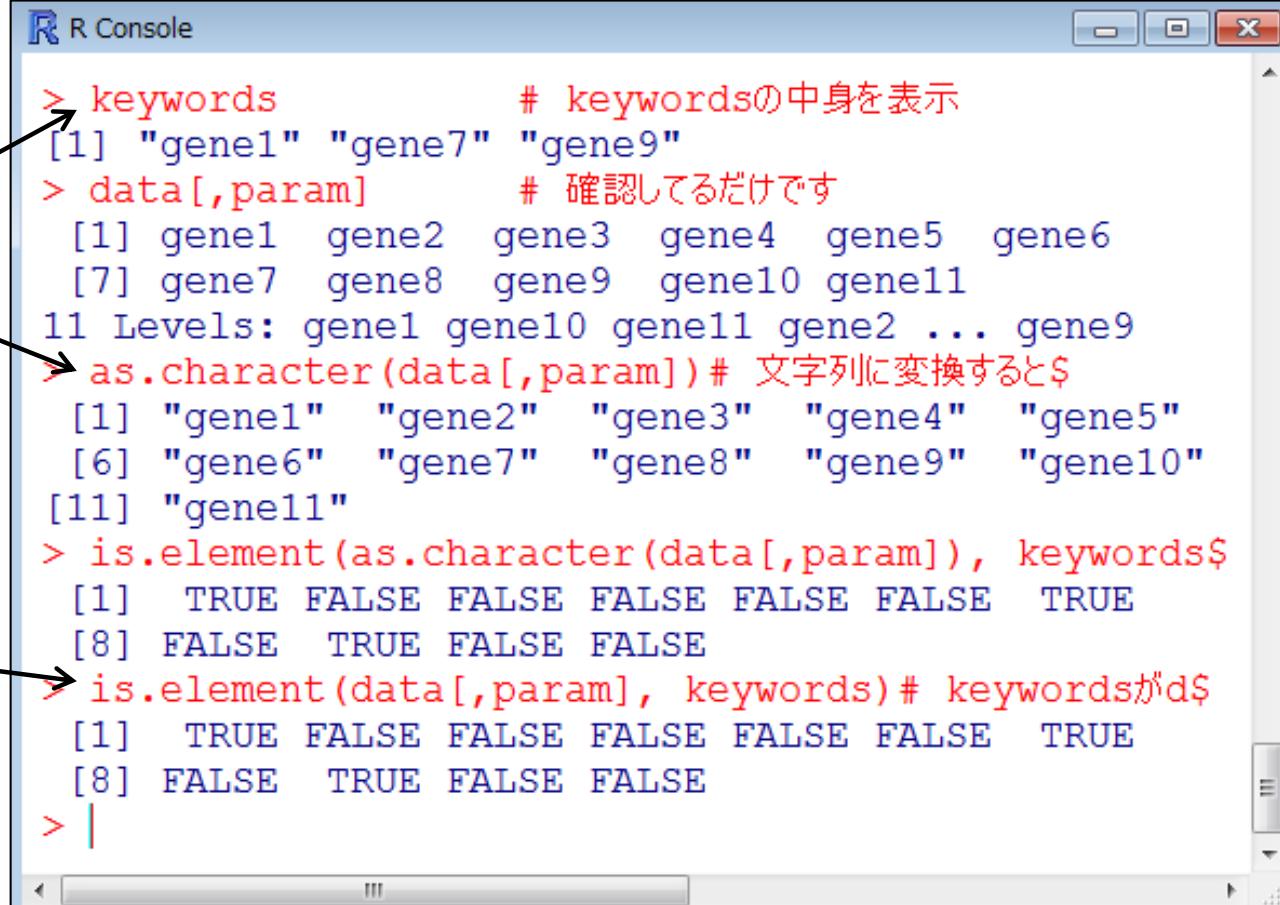
論理値ベクトルobjを用いてTRUEの要素に対応する行を抽出している

# 論理値ベクトルを理解

```
obj <- is.element(as.character(data[,param]), keywords) # 条件を満たすかどうか
out <- data[obj,]                                         # objがTRUEとなる行のみ抽出した結果を
```

疑問に思ったら、自分の理解できるところから試す。本コード作成当時はas.character関数を用いてデータの型を文字列ベクトルに揃えていた。

当時as.character関数を用いる必要があったかどうかは定かではないが、少なくとも現在(R ver. 3.1.0)はas.character関数がなくても大丈夫なようだ。



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

```

> keywords          # keywordsの中身を表示
[1] "gene1" "gene7" "gene9"
> data[,param]     # 確認してるだけです
[1] gene1 gene2 gene3 gene4 gene5 gene6
[7] gene7 gene8 gene9 gene10 gene11
11 Levels: gene1 gene10 gene11 gene2 ... gene9
> as.character(data[,param]) # 文字列に変換すると$
[1] "gene1" "gene2" "gene3" "gene4" "gene5"
[6] "gene6" "gene7" "gene8" "gene9" "gene10"
[11] "gene11"
> is.element(as.character(data[,param]), keywords$)
[1] TRUE FALSE FALSE FALSE FALSE FALSE TRUE
[8] FALSE TRUE FALSE FALSE
> is.element(data[,param], keywords) # keywordsが'd$'
[1] TRUE FALSE FALSE FALSE FALSE FALSE TRUE
[8] FALSE TRUE FALSE FALSE
>

```

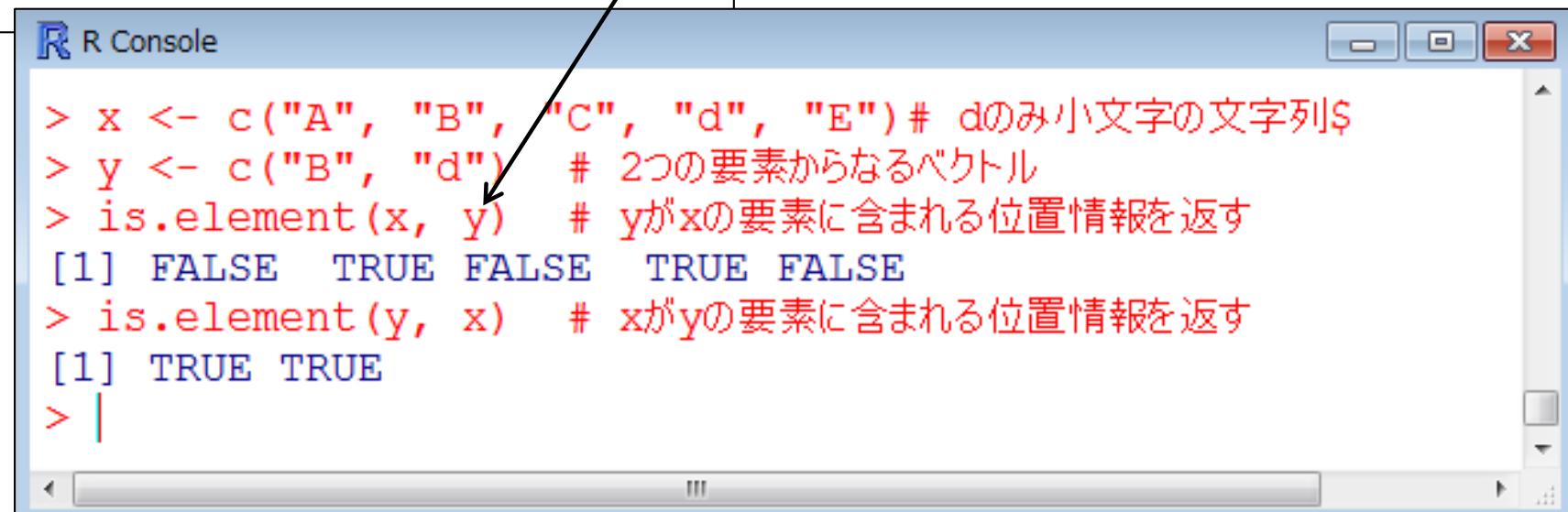
# 論理値ベクトルを理解

ファイル名 : rcode\_20140908.txt

```
#####
## 論理値ベクトル
#####
x <- c("A", "B", "C", "d", "E") # dのみ小文字の文字列ベクトル
y <- c("B", "d") # 2つの要素からなるベクトル
is.element(x, y) # yがxの要素に含まれる位置情報を返す
is.element(y, x) # xがyの要素に含まれる位置情報を返す
```

```
#####
x <- c("A", "B", "C", "d", "E")
z <- c("A", "D", "F")
is.element(x, z) # xがzの要素に含まれる位置情報を返す
is.element(z, x) # zがxの要素に含まれる位置情報を返す
```

is.element関数は、1番目の引数で指定したベクトル(この場合x)の要素が2番目の引数で指定したベクトル(この場合y)の要素として含まれるか否かを TRUEまたはFALSEで返す。



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

```
R Console
> x <- c("A", "B", "C", "d", "E") # dのみ小文字の文字列
> y <- c("B", "d") # 2つの要素からなるベクトル
> is.element(x, y) # yがxの要素に含まれる位置情報を返す
[1] FALSE TRUE FALSE TRUE FALSE
> is.element(y, x) # xがyの要素に含まれる位置情報を返す
[1] TRUE TRUE
>
```

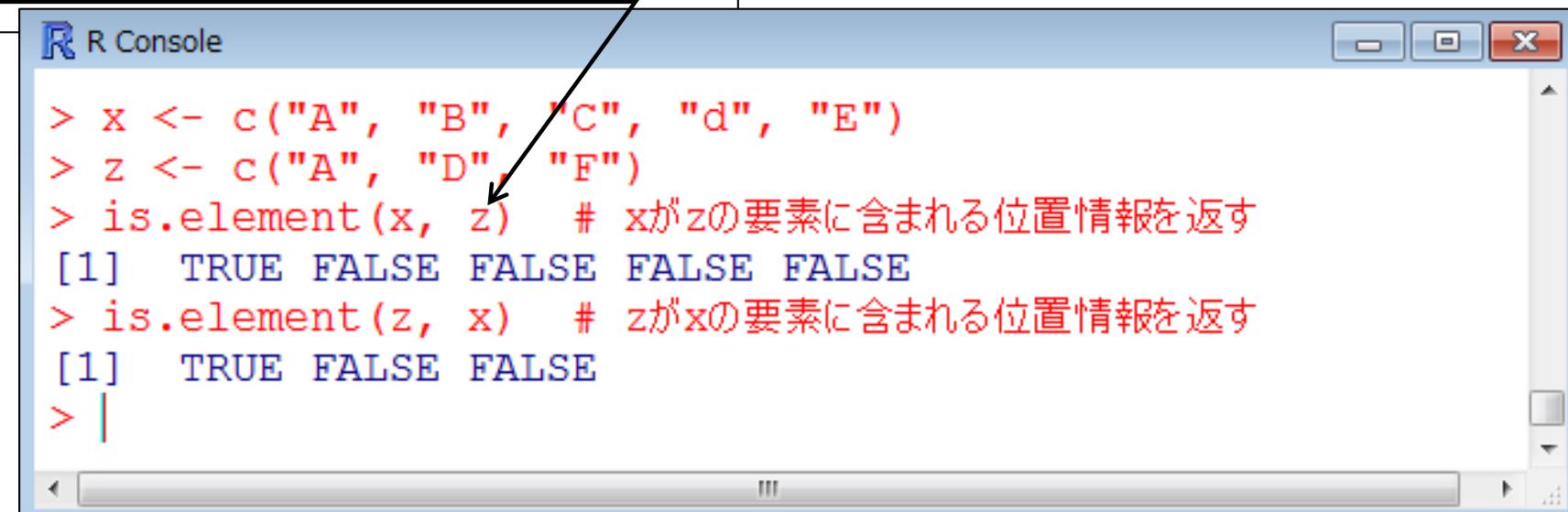
# 論理値ベクトルを理解

ファイル名 : rcode\_20140908.txt

```
#####
## 論理値ベクトル
#####
x <- c("A", "B", "C", "d", "E") # dのみ小文字の文字列ベクトル
y <- c("B", "d") # 2つの要素からなるベクトル
is.element(x, y) # yがxの要素に含まれる位置情報を返す
is.element(y, x) # xがyの要素に含まれる位置情報を返す
#
```

```
x <- c("A", "B", "C", "d", "E")
z <- c("A", "D", "F")
is.element(x, z) # xがzの要素に含まれる位置情報を返す
is.element(z, x) # zがxの要素に含まれる位置情報を返す
#
```

is.element関数は、1番目の引数で指定したベクトル(この場合x)の要素が2番目の引数で指定したベクトル(この場合y)の要素として含まれるか否かをTRUEまたはFALSEで返す。全遺伝子から特定の遺伝子群の位置情報を取得したい場合などによく用います。



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

```
R Console
> x <- c("A", "B", "C", "d", "E")
> z <- c("A", "D", "F")
> is.element(x, z) # xがzの要素に含まれる位置情報を返す
[1] TRUE FALSE FALSE FALSE FALSE
> is.element(z, x) # zがxの要素に含まれる位置情報を返す
[1] TRUE FALSE FALSE
>
```

An arrow points from the explanatory text in the slide to the line of code `is.element(x, z)` in the R console output.

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
```

12. 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、param2で指定した文字列が含まれる行全体を出力したい場合:

```
in_f <- "annotation.txt"          #入力ファイル名を指定してin_fに格納(アノテーションファイル)
out_f <- "hoge12.txt"               #出力ファイル名を指定してout_fに格納
param1 <- 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
param2 <- c("gene1", "gene7", "gene9") #検索したい文字列を指定
```

#入力ファイルの読み込み

```
data <- read.table(in_f, header=TRUE, sep="\t", quote="")#in_f1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[,param1]), param2)#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保存
```

genelist1.txt

	A
1	gene1
2	gene7
3	gene9

1と12は手順が異なるだけで実質的に同じです

```
in_f <- "annotation.txt"          #入力ファイル名を指定してin_fに格納(アノテーション)
out_f <- "hoge12.txt"            #出力ファイル名を指定してout_fに格納
param1 <- 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
param2 <- c("gene1", "gene7", "gene9") #検索したい文字列を指定
```

このコードはヘッダー行  
がある場合のものです

```
#入力ファイルの読み込み
data <- read.table(in_f, header=TRUE, sep="\t", quote="")#in_f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[,param1]), param2)#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保存
```

入力: annotation.txt

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

出力: hoge12.txt

	A	B	C	D
1	genename	accession	description	subcellular_location
2	gene1	hoge01	plasma_mem	nuclear
3	gene7	hoge07	tebasaki	nuclear
4	gene9	hoge09	nihonshu	nuclear

```

in_f <- "annotation2.txt"          #入力ファイル名を指定してin_fに格納(アノテーション)
out_f <- "hoge13.txt"              #出力ファイル名を指定してout_fに格納
param1 <- 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
param2 <- c("gene1", "gene7", "gene9") #検索したい文字列を指定

#入力ファイルの読み込み
data <- read.table(in_f, header=F, sep="\t", quote="")#in_f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[,param1]), param2)#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, col.names=F)#outの中身を指定したファイル名で保存

```

このコードはヘッダー行  
がない場合のものです

入力: annotation2.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: hoge13.txt

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

12. 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、param2で指定した文字列が含まれる行全体を出力したい場合:

・ イントロ | 一般 | 任意のキーワードを含む行を抽出(基礎)

```
in_f <- "annotation.txt"          #入力ファイル名を指定してin_fに格納(アノテーションファイル)
out_f <- "hoge12.txt"            #出力ファイル名を指定してout_fに格納
param1 <- 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
param2 <- c("gene1", "gene7", "gene9") #検索したい文字列を指定
```

ヘッダー行がある場合

```
#入力ファイルの読み込み
data <- read.table(in_f, header=TRUE, sep="\t", quote="")#in_f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[,param1]), param2)#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保存
```

13. 目的のタブ区切りテキストファイル(annotation2.txt)中の第1列目をキーとして、param2で指定した文字列が含まれる行全体を出力したい場合:

入力ファイル中にヘッダー行がない場合の読み込み例です。

```
in_f <- "annotation2.txt"          #入力ファイル名を指定してin_fに格納(アノテーションファイル)
out_f <- "hoge13.txt"            #出力ファイル名を指定してout_fに格納
param1 <- 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
param2 <- c("gene1", "gene7", "gene9") #検索したい文字列を指定
```

ヘッダー行がない場合

```
#入力ファイルの読み込み
data <- read.table(in_f, header=F, sep="\t", quote="")#in_f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[,param1]), param2)#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, col.names=F)#outの中身を指定したファイル名で保存
```

# Contents

## ■ 3-2. R 基礎2、2014/09/08 13:15–14:45、初級、実習

### □ (Rで)塩基配列解析の基本的な利用法(翻訳配列の取得を例に)

- 入力ファイル取得、作業ディレクトリの変更、本番(基本はコピペ)、出力結果の確認
- 1つの項目に多数の例題(異なる入力ファイル、エラーへの対処例)

### □ 行列形式ファイルの解析基礎(アノテーションファイルを例に)

- 例題をテンプレートとして任意の解析を行う基本手順
- ありがちなミスとエラーメッセージ
- 入力ファイルの最後の改行の有無
- プログラム内部の説明(行列演算の基礎)

### □ Tips

- 集合演算:`union`, `intersect`, `setdiff`
- その他:`sort`, `table`, `is.element`, `toupper`, `tolower`



# Tips(集合演算)

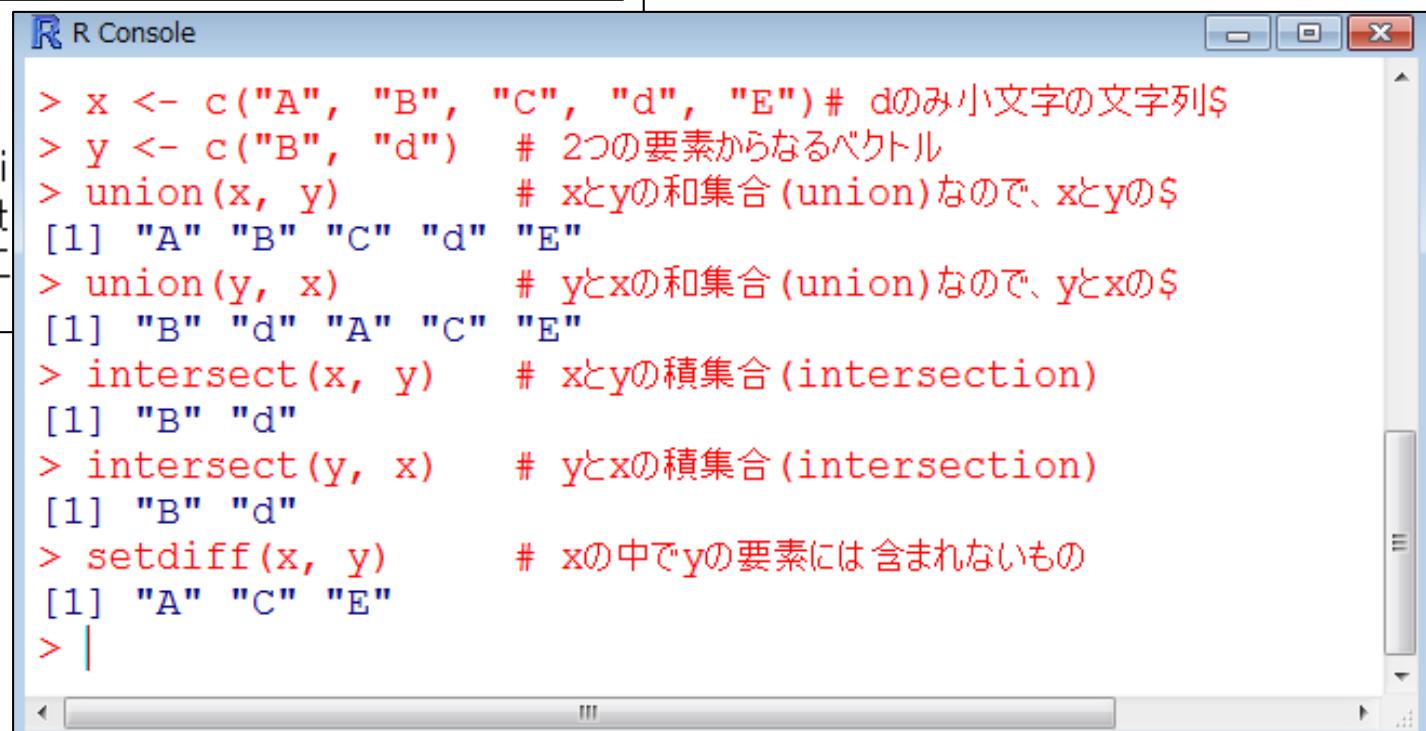
ファイル名 : rcode\_20140908.txt

```
#####
### 集合演算↓
#####
```

```
x <- c("A", "B", "C", "d", "E")# dのみ小文字の文字列ベクトル↓
y <- c("B", "d") # 2つの要素からなるベクトル↓
union(x, y)      # xとyの和集合(union)なので、xとyの位置は無関係↓
union(y, x)      # yとxの和集合(union)なので、yとxの位置は無関係↓
intersect(x, y)  # xとyの積集合(intersection)↓
intersect(y, x)  # yとxの積集合(intersection)↓
setdiff(x, y)    # xの中でyの要素には含まれないもの↓
```

```
↓
x <- c("A", "B", "C", "d", "E")↓
z <- c("A", "D", "F")↓
union(x, z)      # xとzの和集合(uni↓
intersect(x, z)  # xとzの積集合(int↓
setdiff(x, z)    # xの中でzの要素に↓
```

共通遺伝子を得る作業などは  
集合演算用関数を内部的に駆使します。



R Console window showing the execution of R code for set operations:

```
> x <- c("A", "B", "C", "d", "E") # dのみ小文字の文字列$  
> y <- c("B", "d") # 2つの要素からなるベクトル  
> union(x, y)      # xとyの和集合(union)なので、xとyの$  
[1] "A" "B" "C" "d" "E"  
> union(y, x)      # yとxの和集合(union)なので、yとxの$  
[1] "B" "d" "A" "C" "E"  
> intersect(x, y)  # xとyの積集合(intersection)  
[1] "B" "d"  
> intersect(y, x)  # yとxの積集合(intersection)  
[1] "B" "d"  
> setdiff(x, y)    # xの中でyの要素には含まれないもの  
[1] "A" "C" "E"  
> |
```

# Tips(集合演算)

ファイル名 : rcode\_20140908.txt

```
#####
## 集合演算↓
#####
x <- c("A", "B", "C", "d", "E")# dのみ小文字の文字列ベクトル↓
y <- c("B", "d") # 2つの要素からなるベクトル↓
union(x, y)      # xとyの和集合(union)なので、xとyの位置は無関係↓
union(y, x)      # yとxの和集合(union)なので、yとxの位置は無関係↓
intersect(x, y)  # xとyの積集合(intersection)↓
intersect(y, x)  # yとxの積集合(intersection)↓
setdiff(x, y)    # xの中でyの要素には含まれないもの↓
↓
x <- c("A", "B", "C", "d", "E")↓
z <- c("A", "D", "F")↓
union(x, z)      # xとzの和集合(union)↓
intersect(x, z)  # xとzの積集合(intersection)↓
setdiff(x, z)    # xの中でzの要素には含まれないもの↓
↓
```

共通遺伝子を得る作業などは  
集合演算用関数を内部的に駆  
使します。

R Console

```
> x <- c("A", "B", "C", "d", "E")
> z <- c("A", "D", "F")
> union(x, z)      # xとzの和集合(union)
[1] "A" "B" "C" "d" "E" "D" "F"
> intersect(x, z)  # xとzの積集合(intersection)
[1] "A"
> setdiff(x, z)    # xの中でzの要素には含まれないもの
[1] "B" "C" "d" "E"
> |
```

# Tips(その他)

ファイル名 : rcode\_20140908.txt

```
#####↓
### Tips(その他)↓
#####↓

hoge <- c(x, z) # ベクトルxとzを結合。cは結合(concatenate)の意味
hoge           # hogeの中身を表示↓
sorted <- sort(hoge) # アルファベット順にソート↓
sorted          # sortedの中身を表示↓
table(sorted)   # 要素ごとの出現回数↓
table(hoge)     # 要素ごとの出現回数↓
↓
X <- c("C", "A", "B", "E", "D")# 全部大文字の文字
y <- c("f", "b", "d")# 全部小文字からなるベクトル
is.element(X, y) # yがXの要素に含まれる位置情報
↓
Y <- toupper(y) # 大文字に変換(to upper)↓
Y             # Yの中身を表示↓
is.element(X, Y) # YがXの要素に含まれる位置情報
↓
x <- tolower(X) # 小文字に変換(to lower)↓
x             # xの中身を表示↓
is.element(x, y) # yがxの要素に含まれる位置情報
↓
```

table関数も、リードの種類ごとの出現回数やベクトルの要素の重複度合いなどを調べる目的でよく利用されます。

R R Console

```
> hoge <- c(x, z)      # ベクトルxとzを結合。cは結合(concatenate)の意味
> hoge                  # hogeの中身を表示
[1] "A" "B" "C" "d" "E" "A" "D" "F"
> sorted <- sort(hoge) # アルファベット順にソート
> sorted                # sortedの中身を表示
[1] "A" "A" "B" "C" "d" "D" "E" "F"
> table(sorted)         # 要素ごとの出現回数
sorted
A B C d D E F
2 1 1 1 1 1 1
> table(hoge)           # 要素ごとの出現回数
hoge
A B C d D E F
2 1 1 1 1 1 1
> |
```

# Tips(その他)

ファイル名 : rcode\_20140908.txt

```
#####
## Tips(その他)##
#####
hoge <- c(x, z) # ベクトルxとzを結合。cは結合(concatenate)の意味、
hoge           # hogeの中身を表示↓
sorted <- sort(hoge) # アルファベット順にソート↓
sorted          # sortedの中身を表示↓
table(sorted)   # 要素ごとの出現回数↓
table(hoge)     # 要素ごとの出現回数↓
#
X <- c("C", "A", "B", "E", "D")# 全部大文字の文字列ベクトル↓
y <- c("f", "b", "d")# 全部小文字からなるベクトル↓
is.element(X, y) # yがXの要素に含まれる位置情報を返す↓
#
Y <- toupper(y) # 大文字に変換(to upper)↓
Y           # Yの中身を表示↓
is.element(X, Y) # YがXの要素に含まれる位置情報↓
#
x <- tolower(X) # 小文字に変換(to lower)↓
x           # xの中身を表示↓
is.element(x, y) # yがxの要素に含まれる位置情報↓
```

is.element関数は大文字と小文字を区別する点に注意が必要です。アノテーション情報と対応付けしたい場合などに、このようなことが起こりうることを認識しておく必要があります。X中の計5つの要素のいずれもyに含まれないので、全てFALSEとなっています。

R Console

```
> X <- c("C", "A", "B", "E", "D")# 全部大文字
> y <- c("f", "b", "d")# 全部小文字からなるベクトル
> is.element(X, y) # yがXの要素に含まれる位置情報
[1] FALSE FALSE FALSE FALSE FALSE
> |
```

# Tips(その他)

ファイル名 : rcode\_20140908.txt

```
#####
### Tips(その他)↓
#####
hoge <- c(x, z) # ベクトルxとzを結合。cは結合(concatenate)の意味、
hoge           # hogeの中身を表示↓
sorted <- sort(hoge) # アルファベット順にソート↓
sorted          # sortedの中身を表示↓
table(sorted)   # 要素ごとの出現回数↓
table(hoge)     # 要素ごとの出現回数↓
↓
X <- c("C", "A", "B", "E", "D")# 全部大文字の文字列
y <- c("f", "b", "d")# 全部小文字からなるベクトル
is.element(X, y) # yがXの要素に含まれる位置情報を返す↓
↓
Y <- toupper(y) # 大文字に変換(to upper)↓
Y               # Yの中身を表示↓
is.element(X, Y) # YがXの要素に含まれる位置情報を返す↓
↓
x <- tolower(X) # 小文字に変換(to lower)↓
x               # xの中身を表示↓
is.element(x, y) # yがxの要素に含まれる位置情報を返す↓
↓
```

一つの対策は、全部を大文字に変換してから対応付けを行うことです。toupperはベクトル中の文字を大文字に変更する関数です。

R Console

```
> Y <- toupper(y)      # 大文字に変換(to upper)
> Y                      # Yの中身を表示
[1] "F" "B" "D"
> is.element(X, Y)      # YがXの要素に含まれる位置情報
[1] FALSE FALSE TRUE FALSE TRUE
> |
```

# Tips(その他)

ファイル名 : rcode\_20140908.txt

```
#####
## Tips(その他)##
#####
hoge <- c(x, z) # ベクトルxとzを結合。cは結合(concatenate)の意味、
hoge           # hogeの中身を表示↓
sorted <- sort(hoge) # アルファベット順にソート↓
sorted          # sortedの中身を表示↓
table(sorted)   # 要素ごとの出現回数↓
table(hoge)     # 要素ごとの出現回数↓
#
X <- c("C", "A", "B", "E", "D")# 全部大文字の文字列
y <- c("f", "b", "d")# 全部小文字からなるベクトル
is.element(X, y) # yがXの要素に含まれる位置情報を返す↓
#
Y <- toupper(y) # 大文字に変換(to upper)↓
Y             # Yの中身を表示↓
is.element(X, Y) # YがXの要素に含まれる位置情報を返す↓
#
x <- tolower(X) # 小文字に変換(to lower)↓
x             # xの中身を表示↓
is.element(x, y) # yがxの要素に含まれる位置情報を返す↓
#
```

もう一つの対策は、全部を小文字に変換してから対応付けを行うことです。tolowerはベクトル中の文字を小文字に変更する関数です。

R Console

```
> x <- tolower(X)      # 小文字に変換(to lower)
> x                      # xの中身を表示
[1] "c" "a" "b" "e" "d"
> is.element(x, y)    # yがxの要素に含まれる位置情報
[1] FALSE FALSE TRUE FALSE TRUE
> |
```

Google 関田 幸太 ▾

東京大学・大学院農学生命科学研究所  
アグリカルチャー農学研究室  
関田幸太(かた 幸太) 氏  
kadota@i.u-tokyo.ac.jp  
<http://www.i.u-tokyo.ac.jp/~kadota/>

3. フィードバック基礎 | 3-3. 反応性の分析

アグリカルチャー農学研究室  
関田幸太(かた 幸太) 氏  
kadota@i.u-tokyo.ac.jp

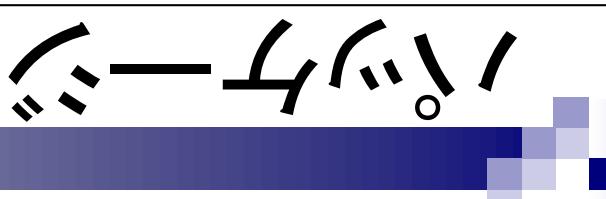


## Contents

- 3-3. R 名稱 / 心力 -> 、2014/09/08 15:00-18:15、中級、美醫

### 1 心力

- 「(RT) 增基配列解析」のトピック一覧
- CRAN Bioconductor
- 表達的分子 / 心力 -> Biostings の利用法 : library, search, objects 開発
- 作業環境 / 心力 -> Workspace の概念
- Biostings / 心力 -> 利用可能な開発環境
- source 開発の利用



## (RT) 基因配列解析



(RT) 基因配列解析の推奨ツール  
手順を示す5点。大まかに手順は、①R  
本体のツール一覧、②CRANの  
データバイオコンドьюクタ、各種ツール一覧  
～NGS、RNA-seq、アノテーション～、正規化、表現量算、統計、電子化、システム

## RTのツール一覧

### 1. Windows releases 版の RTツール環境:

基本的にはWindows2.0版以下を導入する。  
2014年7月31日(木)～2014年5月14日(火)までの手順を示す。  
① 分かりづらい人でWindows2.0版以下を参考する。(ただし、2014年7月31日(木)～2014年5月14日(火)までの手順を示す。  
注意点は、Mac OS XのバージョンがR-3.1.0-snowleopard.pkgであるためMac版の手順を示す。  
② RTツール一覧手順を示す。

### 1

#### RTのツール一覧

2. 開始する方法を示す。ただし、これは開発中の機能であり、まだ実装されていません。

3. Windows VistaのRTツール一覧が表示される。

4. ツールバーに表示される「RT XY (64bit)環境」、「XY中間間」という二つのボタンがあります。  
左側の「XY中間間」をクリックすると、右側に表示される「RT XY (32bit)環境」が選択されます。  
右側の「XY中間間」をクリックすると、左側に表示される「RT XY (64bit)環境」が選択されます。

5. 以下を、RT XY (64bit)環境で実行する場合、左側の「XY中間間」をクリックして選択す  
ます。(右側の「XY中間間」をクリックしても、10GB程度のデータ容量を要する実験手帳で  
は、この操作ができない場合があります。)

6. 「RT XY (64bit)環境」、「XY中間間」、「RT XY (32bit)環境」、「RT XY (64bit)環境」の順序で、  
右側の「XY中間間」をクリックして選択します。#CRAN中間間を全てTRUEにし  
bioclite("Biogenome.Athaliana.TAIR.TAIR9", suppressUpdates=TRUE) #Bioconductor中間間を  
source("http://www.bioconductor.org/biocLite.R") #Bioconductor中間間を全てTRUEにし  
install.packages(availabile.packages()[1], dependencies=TRUE) #CRAN中間間を全てTRUEにし  
bioclite("All\_Group") #Bioconductor中間間を全てTRUEにし

### 2

#### RTのツール一覧

7. 「RT XY (64bit)環境」、「XY中間間」、「RT XY (32bit)環境」、「RT XY (64bit)環境」の順序で、  
右側の「XY中間間」をクリックして選択します。#CRAN中間間を全てTRUEにし  
source("http://www.bioconductor.org/biocLite.R") #Bioconductor中間間を全てTRUEにし  
install.packages(availabile.packages()[1], dependencies=TRUE) #CRAN中間間を全てTRUEにし  
bioclite("All\_Group") #Bioconductor中間間を全てTRUEにし

### 2

#### RTのツール一覧

8. 「RT XY (64bit)環境」、「XY中間間」、「RT XY (32bit)環境」、「RT XY (64bit)環境」の順序で、  
右側の「XY中間間」をクリックして選択します。#CRAN中間間を全てTRUEにし  
source("http://www.bioconductor.org/biocLite.R") #Bioconductor中間間を全てTRUEにし  
install.packages(availabile.packages()[1], dependencies=TRUE) #CRAN中間間を全てTRUEにし  
bioclite("All\_Group") #Bioconductor中間間を全てTRUEにし

「R本体」は「Rの統一された機能」、「Rによって開発された機能」、「Microsoft EXCEL」「TSP」と「Cytoscape」など、多くのツールが統合されています。

- R本体をベースとする統合的な機能を持つ、さまざまな限られたR本体。
- Linuxをベースとする統合的な機能を持つ、さまざまな限られたR本体。
- 通常は、Rによって開発された機能を持つ、さまざまな限られたR本体。
- R本体をベースとする統合的な機能を持つ、さまざまな限られたR本体。
- NGS解析を行った各種の統合された機能を持つ、さまざまな限られたR本体。

## R本体と統一された機能

- R上で利用可能な、約100大項目(貯蔵庫)
  - CRAN (The Comprehensive R Archive Network) : 5,802/約1千
  - Biocductor: 824/約1千

CRANは生命科学分野に限らず  
様々な分野で利用される  
一大宝庫だ。NGS解析手法、主に  
Bioconductorなどを提供するCUI  
Bioconductor力を利用ользす。

- R
- Biocductor: Gentleman et al., Genome Biol., 2004
- CRAN
- R Tipps (アドバイス)
- BioEdit (アドバイス)
- BioMart: Smedley et al., BMC Genomics, 2009
- DBJ Read Annotation Pipeline: Nagasaki et al., DNA Res., 2013
- EMBOSS explorer (EMBOSS) [マニュアル]
- Biostar: Pameili et al., PLoS Comput Biol., 2011
- SEQanswers: Li et al., Bioinformatics, 2012
- NGS WikiBook: Li et al., Brief Bioinform., 2013
- HT Sequence Analysis with R and Bioconductor

■	MA plot (基本編) (Last modified 2014/02/04)
■	M-A Plot (差別化) (Last modified 2013/10/01)
■	ROC曲線 (Last modified 2013/02/28)
■	SPLICINGGRAPHS (Last modified 2013/02/28)
■	BiocDB (Last modified 2012/12/03/29)
■	small RNA (small RNA   small RNA) (Last modified 2012/03/29)
■	機能解析 (機能解析   機能解析) (Last modified 2012/03/29)
■	表現型 (表現型   表現型) (Last modified 2012/03/29)
■	R Tipps (アドバイス) (Last modified 2012/03/29)
■	CRAN (CRAN   CRAN) (Last modified 2012/03/29)
■	Biocductor (Biocductor   Biocductor) (Last modified 2012/03/29)
■	R (R   R) (Last modified 2012/03/29)

# CRANとBioconductor

## ■ 「(R2) 雷基配列解析」の手順を述べる



• R01へ戻る - リンク

(R2) 雷基配列解析の概要について手順法、CRANにおける  
Biocductorの提供機能の各種手順法、Biocductorの  
構成環境(七種類)開発者用の手順法。

### R01へ戻る - リンク NEW

1. Windows release版へ戻る - リンク:

14日以内にWindows 7以降Mac版のリリース手順を更新する  
ための方法(Windows 7以降Mac版のリリース手順を更新する  
ための方法)。

2. 複数の方法で実現するための手順法へ戻る - リンク:

1. R01へ戻る - リンク

3. Windows Vista版へ戻る - リンク:  
Windows Vista版の手順法は、Windows 7以降の手順法とほとんど  
同じですが、Windows Vista版ではUAC(ユーザーアカウント制御)  
が導入されたため、手順法に変更があります。

4. R2へ戻る - リンク:  
R2版の手順法は、R3版の手順法とほとんど同じですが、手順法に  
変更があります。

5. 64bit版へ戻る - リンク:  
R3版の手順法とほとんど同じですが、手順法に変更があります。

6. R2へ戻る - リンク:  
R2版の手順法は、R3版の手順法とほとんど同じですが、手順法に  
変更があります。

7. CRAN中における手順法へ戻る - リンク:

install.packages("available.packages()", 1, dependencies=TRUE) #CRAN中における手順法へ戻る

source("http://www.biocconduct.org/biocLite.R") #CRAN中における手順法へ戻る

biocLite("Biogenome.Athaliana.TAIR9", suppressUpdates=TRUE) #Biocductor中における手順法へ戻る

6. 「R2」へ戻る - リンク:  
「R2」へ戻る - リンク: 「手順法」 - 「手順法」 - 「手順法」 - 「手順法」 - 「手順法」 - 「手順法」

7. 「R2」手順法へ戻る - リンク:

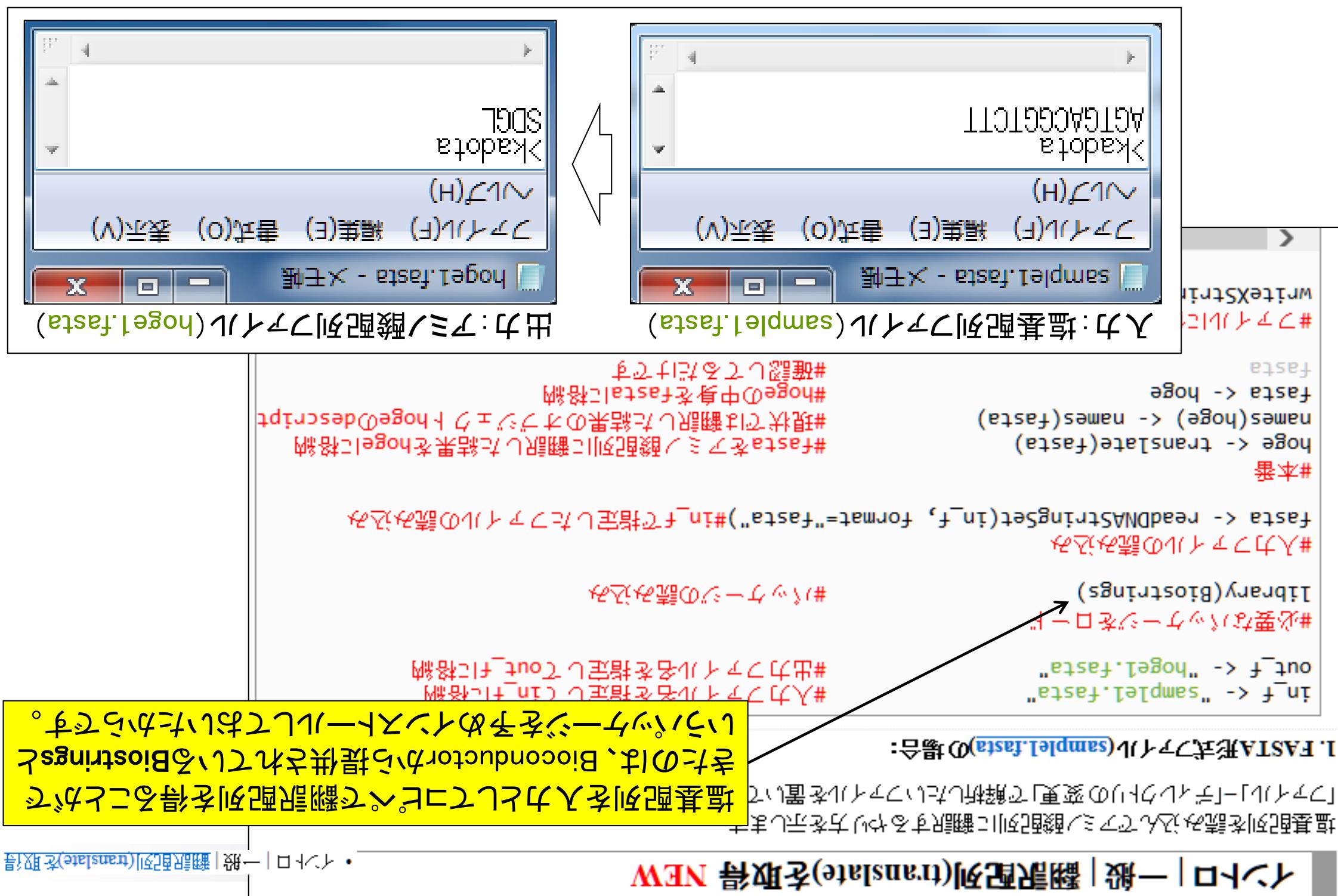
• **任意の場所で可能な全ての塩基配列変換** (Last modified 2013/09/12)  
指定した範囲の塩基配列を置換 (Last modified 2014/03/08)  
指定されたID(染色体等)配列を置換 (Last modified 2013/06/14)  
翻訳配列(translate)と配列を置換 (Last modified 2013/06/14)  
逆翻訳(reverse)と置換 (Last modified 2013/06/14)  
逆相補鎖(reverse complement)と置換 (Last modified 2013/06/14)  
翻訳配列(translate)と置換 (Last modified 2013/06/14)  
塩基配列を人から機械的に翻訳 (Last modified 2013/06/14) NEW

## ツバロ | 一般 | 翻訳配列(translate)を取得 NEW

「**FASTA形式ツール(sample.fasta)**」  
塩基配列を読み取るためのツールです。このツールは以下の機能を持っています。  
1. FASTA形式ツール(sample.fasta)の場合は、  
#出力する名前を指定する(in\_.fasta)→  
#必要なライブラリを読み込む→  
#入力する名前を指定する(in\_.fasta)→  
#出力する名前を指定する(out\_.fasta)→  
#人間が読みやすいように翻訳する→  
#FASTA形式で表示する

```
#FASTA形式ツール(sample.fasta)の場合は、
#出力する名前を指定する(in_.fasta)→
#必要なライブラリを読み込む→
#入力する名前を指定する(in_.fasta)→
#出力する名前を指定する(out_.fasta)→
#人間が読みやすいように翻訳する→
#FASTA形式で表示する

in_f <- "sample1.fasta"
out_f <- "hoge1.fasta"
library(Biostrings)
fastaa <- readDNAStringSet(in_f, format="fasta") #in_.fasta指定期にあたる
#人为的にFASTA形式で表示
#FASTA形式で表示する
names(hoge) <- names(fasta)
hoge <- translate(fasta)
names(hoge) <- hoge
fastaa <- hoge
#hogeの中身をfastaに格納
#FASTA形式で翻訳結果をhogeに格納
#hogeのdescription
#FASTA形式で翻訳結果をhogeに格納
#hogeの中身をfastaに格納
#FASTA形式で翻訳結果をhogeに格納
#FASTA形式で翻訳結果をhogeに格納
#FASTA形式で表示する
#FASTA形式で表示する
```



FASTA形式ファイル(sample.fasta)の場合は：

```

1. FASTA形式ファイル(sample.fasta)の場合：
in_f <- "sample.fasta"
out_f <- "hogeh1.fasta"
#人为的に名前を挿入
library(Biostrings)
#必要なライブラリ読み込み
#人为的に環境変数
fasta <- readDNAStringSet(in_f, format="fasta")#in_fで指定したFASTA形式のDNA配列を読み込む
#人为的に名前を挿入
hogeh1 <- translate(fasta)
#fastaを読み込む際配列を翻訳する結果をhogeh1に格納
names(hogeh1) <- names(fasta)
#hogeh1の中身をfastalに格納
hoge <- transLate(fasta)
#本番
#hogeh1の中身をfastalに格納
#fastalを読み込む際配列を翻訳する結果をhogeに格納
#hogeh1の中身をfastalに格納
#fasta <- hoge
#fasta <- hogeh1
#人为的に名前を挿入
writeStringSet(fasta, file=out_f, format="fasta", width=50) #fastal中身を指定して出力
#人为的に保存

```

FASTA形式以外の場合は：

```

1. FASTA形式以外(sample.fasta)の場合：
in_f <- "sample.fasta"
out_f <- "hogeh1.fasta"
#人为的に名前を挿入
library(Biostrings)
#必要なライブラリ読み込み
#人为的に環境変数
fasta <- readDNAStringSet(in_f, format="fasta")#in_fで指定したFASTA形式のDNA配列を読み込む
#人为的に名前を挿入
hogeh1 <- translate(fasta)
#fastaを読み込む際配列を翻訳する結果をhogeh1に格納
names(hogeh1) <- names(fasta)
#hogeh1の中身をfastalに格納
hoge <- transLate(fasta)
#本番
#hogeh1の中身をfastalに格納
#fastalを読み込む際配列を翻訳する結果をhogeに格納
#hogeh1の中身をfastalに格納
#fasta <- hoge
#fasta <- hogeh1
#人为的に名前を挿入
writeStringSet(fasta, file=out_f, format="fasta", width=50) #fastal中身を指定して出力
#人为的に保存

```



# 原因既知狀態之工于一念出

(本圖與上圖之R方法執行一百次了)

R Version 3.1.0 (2014-04-10) -- "Spring Dance"

Platform: x86\_64-w64-mingw32/x64 (64-bit)

R起動直後的狀態(下記口一念之)

```
# 定義文件(Biostrings/Biostrings.R)
#####
##### 需要將引自取得(Biostrings/之另一個文件)
##### 請將R暫存庫之名稱定於in_f(in_f之檔名)
#####
##### 請將R暫存庫之名稱定於out_f(out_f之檔名)
#####
##### 請將R暫存庫之名稱定於hose1.fasta
#####
##### 請將R暫存庫之名稱定於sample1.fasta
#####
##### 請將R暫存庫之名稱定於library(Biostrings)
#####
##### 請將R暫存庫之名稱定於library(Biostrings)
#####
##### 請將R暫存庫之名稱定於names(hose)
#####
##### 請將R暫存庫之名稱定於hose <- translate(fasta)
#####
##### 請將R暫存庫之名稱定於fastaa <- hose
#####
##### 請將R暫存庫之名稱定於names(fasta)
#####
##### 請將R暫存庫之名稱定於names(fasta)
#####
##### 請將R暫存庫之名稱定於writeLines(fasta, file="fasta", format="fasta", width=50) #fastaa中身之指
```

# 原因既知狀態下一次出力

```

37 命名: code_20140908.txt
#### 開啟既知狀態(Biostrings)
## in_f <- "sample1.fasta"
## out_f <- "sample1.fasta"
## library(Biostrings)
## library(Biostrings)
## 以樣本檔為起點(in_f)
## 必要的生物檔一覽
## fasta <- readDNAStringSet(in_f, format="fasta")#in_函指定$文件
## 大力士序列寫法(this)
## 本體
## hoge <- translate(fasta)
## fasta <- names(hoge) #現快(時間縮短)化
## hoge <- fasta[<-hoge] #hogeh文件中身含fastas
## 離開UCL學院方法
#hogeh文件
#fasta <- hogeh
#names(hoge) <- names(fasta)
## 註解"translate"是貝氏法比對結果在文件內
## 翻譯成"fasta"格式
#fasta <- writeStringSet(fasta, file="f", format="fasta", width=55
## 例: 圖數"writeStringSet"是貝氏法比對結果在文件內

```

```

> fasta <- readDNAStringSet(in_f, format="fasta") #in_文本指定期
> names(fasta) #本署
> hoge <- translate(fasta) #Fastata文本
> names(hoge) <- names(fasta) #提取快译文本结果
> fasta <- hoge #Hoge文本中包含
> I3-: fasta <- hoge #Hoge文本
> I3-: fasta <- hogefastafasta #将Hoge文本翻译为fastafasta
> I3-: writeTextStringSet(fasta, file=out_f, format="fasta", width=50) #将文本写入文件
> I3-: 開數 "writeTextStringSet(fasta, file=out_f, format="fasta", width=50) #將文本寫入文件"
> #將文本保存
> I3-: fasta #将文本写入文件
> I3-: fasta <- hogefastafasta #将Hoge文本翻译为fastafasta
> I3-: package:stats
> I3-: package:graphics
> I3-: package:gridDevices
> I3-: package:utils
> I3-: package:base
> I3-: package:methods
> I3-: package:datasets
> I3-: package:globalEnv
> I3-: in_f out_f
> I3-: sample1.fasta
> I3-: hogefasta
> I3-: out_f
> > objects()
> search()
I3-: hogefastafasta #将Hoge文本翻译为fastafasta

```

names函数从文本中读取文本，将文本翻译为Fastata格式，并将结果存储在fasta对象中。

in\_f指定期文本以FASTA形式显示于控制台。

这与文本的readDNAStringSet函数实现段落。

将文本翻译为Fastata格式并输出到文件。

search函数出现在源代码中。进而  
分析这个类的实现。进而、明示  
的向量出现在源代码中。进而、  
表示文本。其次、R包重写的基本  
的父类。进而、(stats, graphics, utils, base  
的名字。进而、进而、进而。  
进而、进而、进而。

```
> fasta <- readDNASTrikingSet (in_f, format="fasta") #in_f指定文件名
> #本署
> hoge <- translate (fastaa)
> names (hoge) <- names (fastaa)
> #提取文本
> #hogee中包含
> fasta <- hogee
> #提取文本
> fasta <- fasta
> #hogee中包含
> fasta <- hogee, fasta, file=outfile
> #将文本保存
> writeTextStringSet (fastaa, file=outfile, format="fasta", width=50) #宽
> fasta
> search()
> fasta, file=outfile
> in_f, out_f
> sample1.fasta
> out_f
> hoge1.fasta
> package:methods "AutoLoads"
[4] "package:gridDevices" "package:utils"
[5] "GlobalEnv" "package:stats"
[6] "package:graphics" "package:datasets"
[7] "package:base"
> package:base
[1] "out_f"
[2] "in_f" "out_f"
[3] "objects()"
[4] "in_f" "out_f"
[5] "sample1.fasta"
[6] "hoge1.fasta"
[7] "out_f"
```

```
> writeTextStringSet (fastaa, file=outfile, format="fasta", width=50) #宽
> fasta
> search()
> fasta, file=outfile
> in_f, out_f
> sample1.fasta
> out_f
> hoge1.fasta
> package:methods "AutoLoads"
[4] "package:gridDevices" "package:utils"
[5] "GlobalEnv" "package:stats"
[6] "package:graphics" "package:datasets"
[7] "package:base"
> package:base
[1] "out_f"
[2] "in_f" "out_f"
[3] "objects()"
[4] "in_f" "out_f"
[5] "sample1.fasta"
[6] "hoge1.fasta"
[7] "out_f"
```

```

> fasta <- readDNASTringSet (in_f, format="fasta") #in_文件指定期
> #本番
> hoge <- translate (fastaa) → #fastaa文件
> hoge <- names (hoge) ← names (fastaa)
> hoge <- hogefile (身体) #提取快照
> hoge <- hogefile (头部) #提取头部
> hoge <- hogefile (尾部) #提取尾部
> hoge <- hogefile (整体) #提取整体
> hoge <- hogefile (中间) #提取中间
> hoge <- hogefile (左端) #提取左端
> hoge <- hogefile (右端) #提取右端
> hoge <- hogefile (中心) #提取中心
> hoge <- hogefile (保存) #提取保存
> > writeStringSet (fastaa, file=out_f, format="fasta", width=50) #输出文件
> fasta
> fasta: 力加载 fasta 文件读取
> >
> > search()
> package:stats
> package:graphics
> package:gridDevices
> package:utils
> package:base
> package:autoloads
> package:methods
> package:globalEnv
> > in_f "in_f" "out_f"
> in_f
> sample1.fasta
> out_f
> hogef1.fasta
> >

```

translate函数将fastaa文件进行翻译操作。

hoge对象是fastaa文件的快照。hoge文件中包含以下内容：

- #提取头部
- #提取中间
- #提取尾部
- #提取整体
- #提取左端
- #提取右端
- #提取中心
- #提取保存

写入文件：writeStringSet(fastaa, file=out\_f, format="fasta", width=50) #输出文件

fastaa: 力加载 fasta 文件读取

If value is shorter than x, it is extended by character Nas to the length of x.

**R Documentation**

The Names of an Object

Names

names(base) >

Functions to get or set the names of an object.

Usage

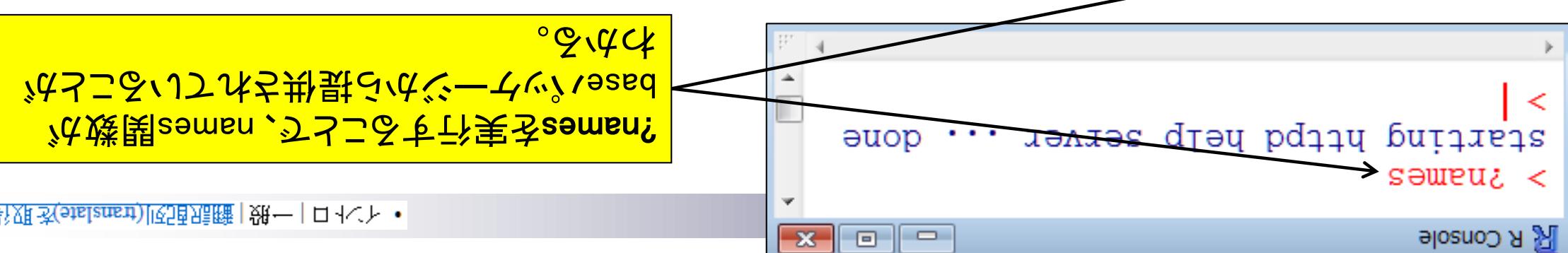
names(x) <- value

Value

a character vector of up to the same length as x, or NULL.

Details

names is a generic accessor function, and names<- is a generic replacement function. The default methods get and set the "names" attribute of a vector (including a list) or pairlist.



从左到右依次为：`> search()`、`> GlobalEnv`、`[1] "package:stats"`、`"package:grDevices"`、`[3] "package:graphics"`、`"package:utils"`、`[5] "package:methods"`、`"package:base"`、`[9] "readDNAstrinSet"`、`> Library(Biostrings)`、`要求添加的包：``Biocgenerics`、`Parallel`、`从左到右依次为：Biostrings`、`readDNAstrinSet`、`specific packages and libraries:`、`No documentation for 'readDNAstrinSet'`、`You could try '?readDNAstrinSet'`、`要求添加的包 -> Parallel 需要添加的包 -> Biocgenerics`、`Parallel 中心包`、`Biocgenerics 中心包`、`Parallel 中心包`、`从左到右依次为：from 'package:parallel'`、`from 'package:stats'`、`from 'package:parLapply'`、`clusterEval`、`clusterApply`、`clusterMap`、`parSapply`、`parLapply`、`clusterExport`、`clusterApply`、`clusterCall`、`clusterEvalQ`、`clusterApplyLB`、`parRapply`、`parSapplyLB`、`parLapplyLB`、`xtabs`

library(Biostrings) 執行之後的搜尋結果

```

library(Biostrings) 執行之後的搜尋結果
anyDuplicated, append, as.data.frame, as.
cbind, colnames, do.call, duplicated, eval.
filter, find, get, intersect, is.unsorted,
map, mapapply, match, mget, order, paste, p
max, min, min.int, position, rank, p
reduce, rep.int, rownames, sapply, setdiff
table, tapply, union, unique, unlist
要來查詢哪一行→ XVector 集合中存在
要來查詢哪一行→ Ranges 集合中存在
要來查詢哪一行→ GlobalEnv
< search()
[1] "GlobalEnv"
[2] "package:Biostrings"
[3] "package:XVector"
[4] "package:Ranges"
[5] "package:BiocGenerics"
[6] "package:parallel"
[7] "package:stats"
[8] "package:grDevices"
[9] "package:utils"
[10] "package:methods"
[11] "package:datasets"
[12] "package:base"
[13] "autoloads"

```

&gt; search()

library(Biostrings) 実行(←←←←←、search())  
 パッケージ:Biostrings  
 パッケージ:Ranges  
 パッケージ:BiocGenerics  
 パッケージ:XVector  
 パッケージ:GlobalEnv

R Documentation  
 Read/write an XStringSet object from/to a file

XStringSet&lt;-to {Biostrings}

Description

Functions to read/write an XStringSet object from/to a file.

Usage

```
## Read FASTA (or FASTQ) files in an XStringSet object:  

readBSStringSet (filepath, format="fasta",  
                nrec=1L, skip=0L, seek.first.rec=FALSE, use.names=TRUE)  

readDNAStringSet (filepath, format="fasta",  
                nrec=1L, skip=0L, seek.first.rec=FALSE, use.names=TRUE)  

readRNASTringSet (filepath, format="fasta",  
                nrec=1L, skip=0L, seek.first.rec=FALSE, use.names=TRUE)  

readASStringSet (filepath, format="fasta",  
                nrec=1L, skip=0L, seek.first.rec=FALSE, use.names=TRUE)  

## Extract basic information about FASTA (or FASTQ) files  

## without actually loading the sequence data:
```

> ?readDNAStringSet  
 starting httpd help  
 [1] "AutoLoads"  
 [2] "Package:datasets"  
 [3] "Package:grDevices"  
 [4] "Package:utils"  
 [5] "Package:parallel"  
 [6] "Package:stats"  
 [7] "Package:graphtics"  
 [8] "Package:grDevices"  
 [9] "Package:utils"  
 [10] "Package:base"  
 > package:methods

→二二七九九九圖<主>二九三。

Biostringsを含むパッケージ。この段階  
 選択肢が表示される。右側に表示する  
 パッケージ名を右クリックして「パッケージ  
 を開く」を選択する。

# 第二- 基本命令

文件名: code\_20140908.txt

#读取样本文件并输出其序列、以前(Biostrings)工具分析输出文件名为。  
#输出结果保存在hogel.fasta中。

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

```
#从文本文件读取序列
library(Biostrings)
#需要输入文件名
#读取文件
fasta <- readDNAStringSet(in_f, format="fasta") #in_f指定文件名
#人为地将序列设置为统一格式
hoge <- translate(fasta)
```

```
#本题
#将序列翻译成氨基酸
#提取出所有氨基酸
# hoge 中包含 fasta 的结果
names(hoge) <- names(fasta)
hoge <- translate(fasta)
# fasta 序列 / 翻译结果
# fasta 中包含名字
names(fasta) <- names(fasta)
# hoge 中包含名字
#提取出名字
kadrota
width seq
[1] 4 SDG
names
A AStringSet instance of Length 1
# 随机乱写
# 保存
# 将序列存入文件
```

```
fasta <- hoge
names(hoge) <- names(fasta)
hoge <- translate(fasta)
# 人为地将序列设置为统一格式
# 读取DNAStringSet
fasta <- readDNAStringSet()
#人为地将序列设置为统一格式
#从文本文件读取序列
library(Biostrings)
#需要输入文件名
#读取文件
hoge <- translate(fasta)
# fasta 中包含名字
#提取出名字
#提取出所有氨基酸
# hoge 中包含 fasta 的结果
names(hoge) <- names(fasta)
hoge <- translate(fasta)
# fasta 序列 / 翻译结果
# fasta 中包含名字
#提取出名字
kadrota
width seq
[1] 4 SDG
names
A AStringSet instance of Length 1
# 随机乱写
# 保存
# 将序列存入文件
```

```
> in_f <- "sample1.fasta"
> out_f <- "hogel.fasta"
> in_f <- "sample1.fasta"
> out_f <- "hogel.fasta"
#人为地将序列翻译成氨基酸
#提取出所有氨基酸
# hoge 中包含 fasta 的结果
#提取出名字
# fasta 中包含名字
#提取出名字
#快照乱写
# 保存
# 将序列存入文件
```

```
[1] > hoge <- translate(fasta)
[2] > names(hoge) <- names(fasta)
[3] > #FASTA文件/翻譯配列取得結果顯示
[4] > #hoged()中包含fasta結構物
[5] > #確認U/C與D/E文字
[6] > fasta <- hoge
[7] > #確認快取值翻譯結果顯示
[8] > #確認快取值翻譯結果顯示
[9] > #確認快取值翻譯結果顯示
[10] > #確認快取值翻譯結果顯示
[11] > #確認快取值翻譯結果顯示
[12] > objects()
[13] [1] "fasta" "hoge" "in_F" "out_F"
[14] > writeTextStringSet(fasta, file=out_F, format="fasta", width=50) #fast
[15] > #寫出檔案
[16] > fasta
[17] > names
[18] > kadota
[19] > A ASTringSet instance of Length 1
[20] > > hoge
[21] > width seq
[22] > A ASTringSet instance of Length 1
[23] > > fasta
[24] > width seq
[25] > A ASTringSet instance of Length 1
[26] > > kadota
[27] > names
[28] > kadota
[29] > names
[30] > kadota
[31] > names
```

## 利用可能方法之工作表示

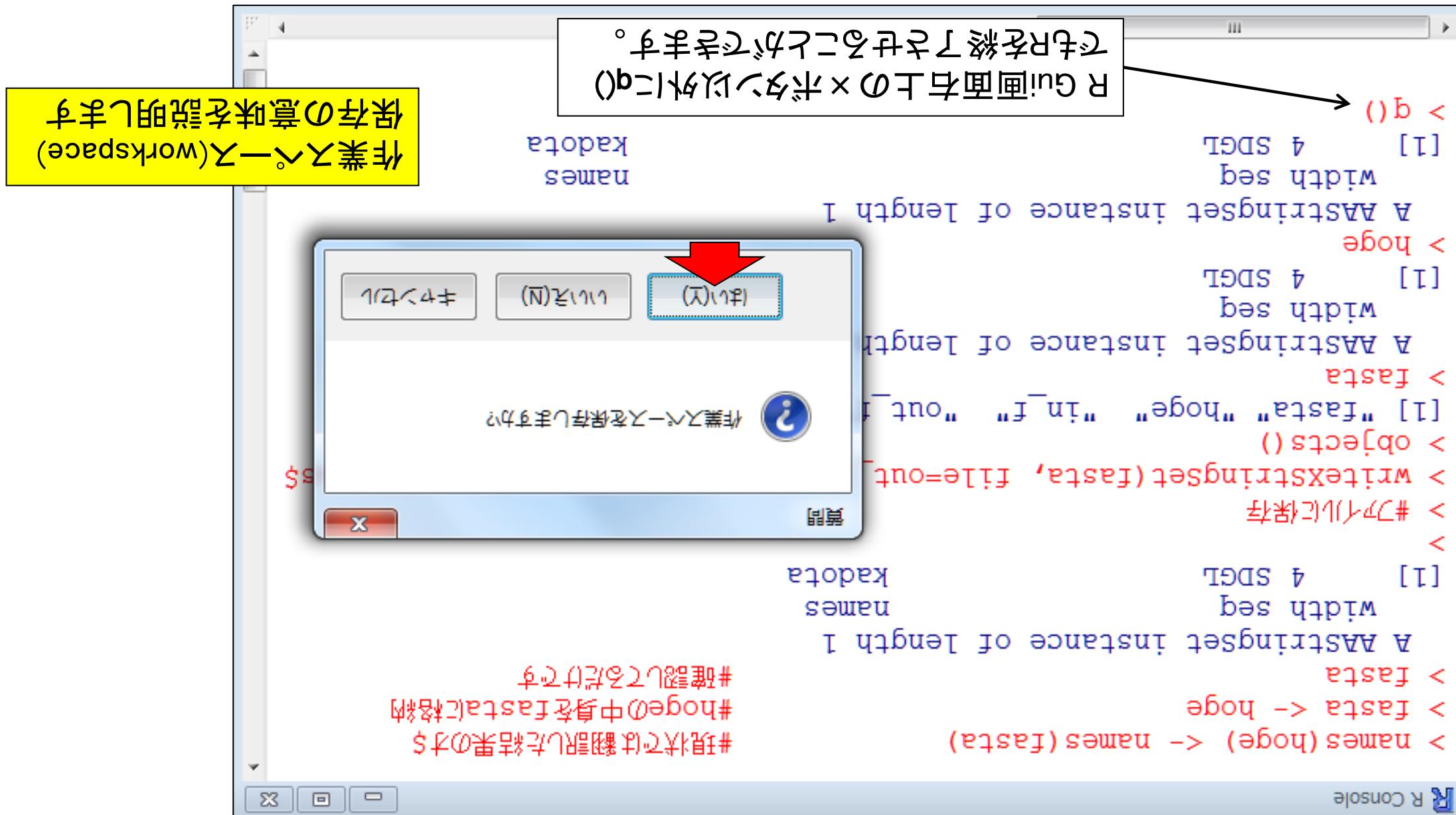


## Contents

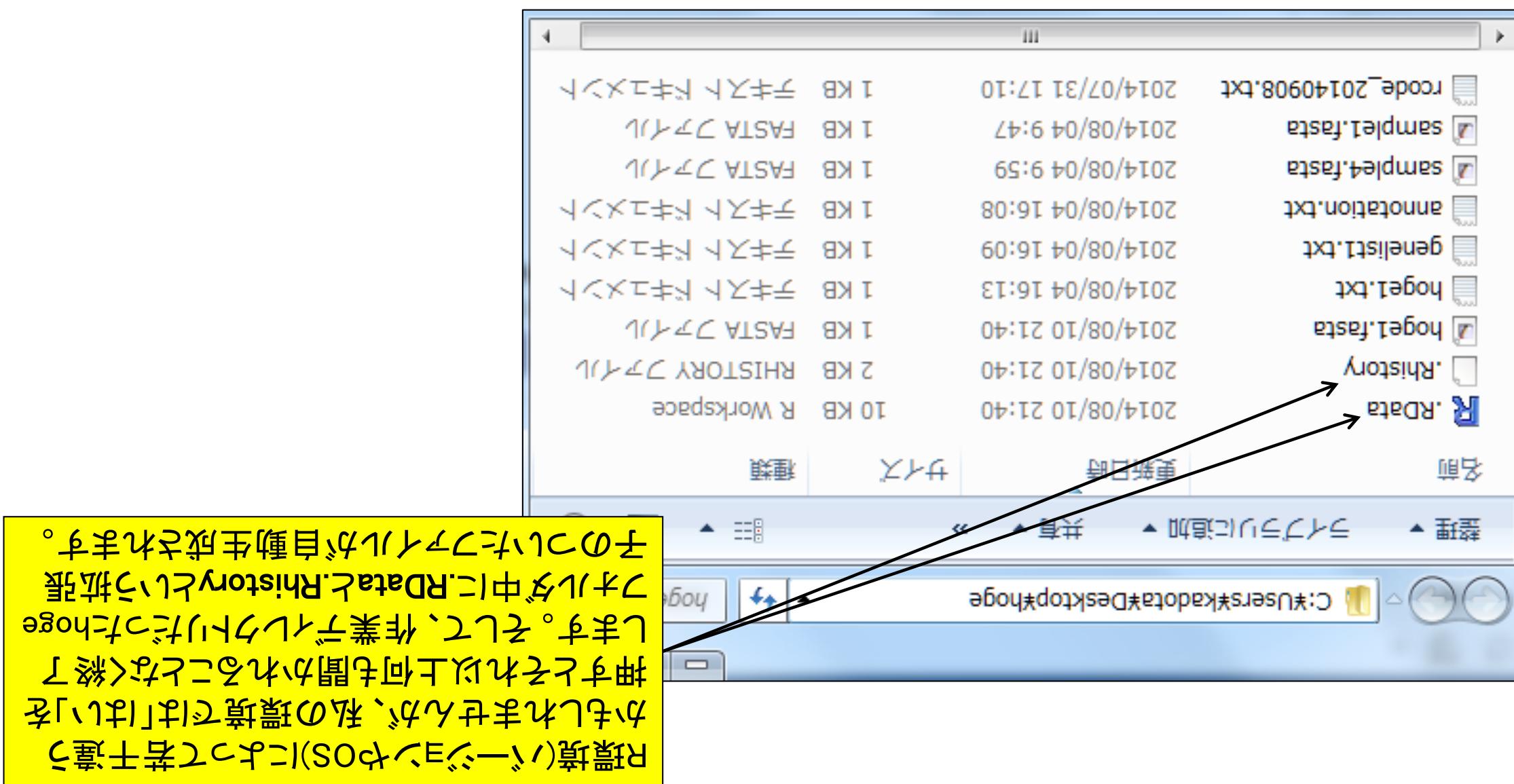
- 3-3. R 管理 / 洋子 -> 2014/09/08 15:00-18:15、中級、美醫

### 二、

- 「(RT) 增基配列解折」のトピック一覧手順
- CRAN Bioconductor
- 表達的分子 / 洋子 -> Biostings の利用法 : library, search, objects
- 作業分子 -> (workspace) の概念
- Biostings / 洋子 -> 利用可能な分子を複数
- source 複数の利用



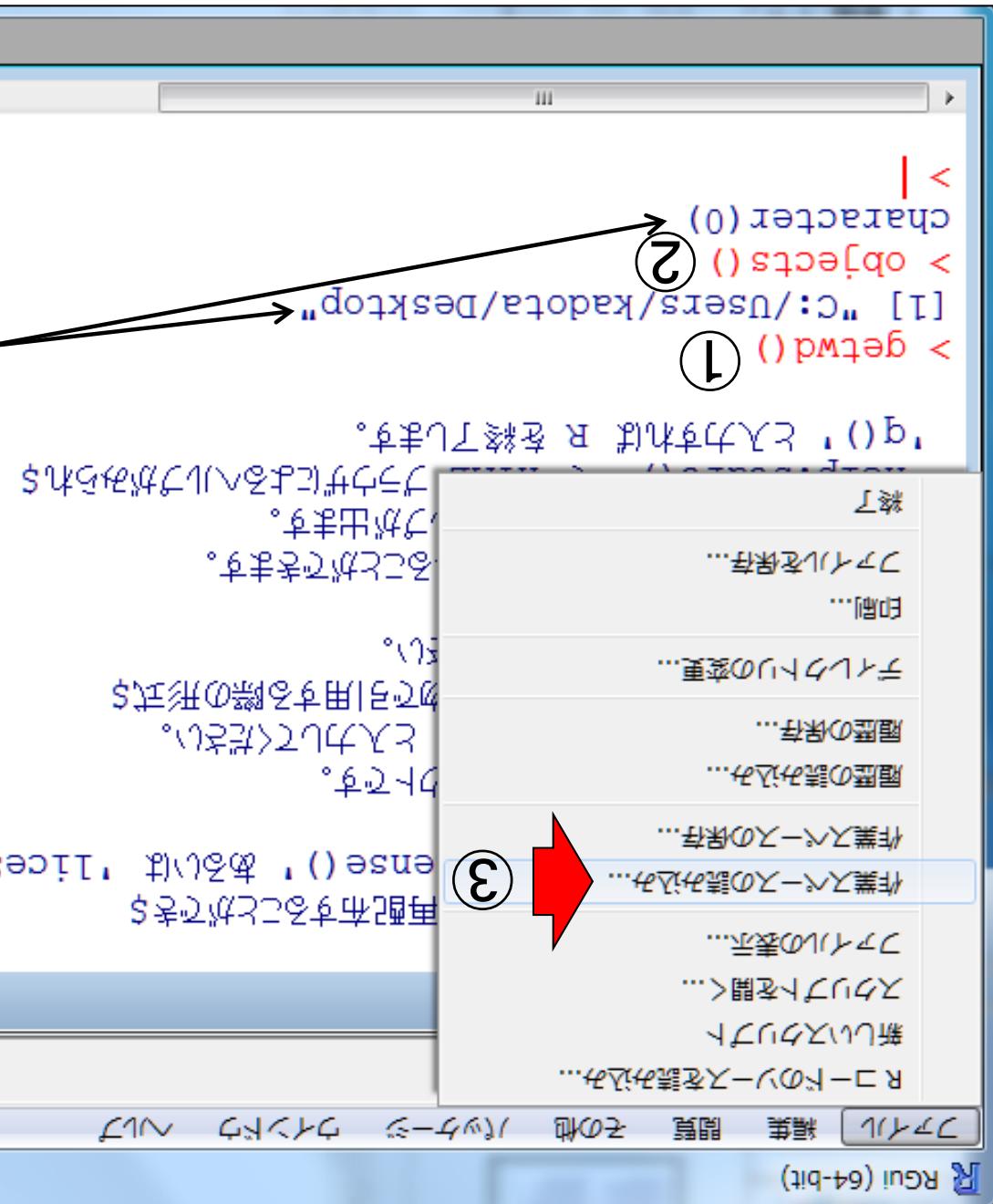
# Rの終了

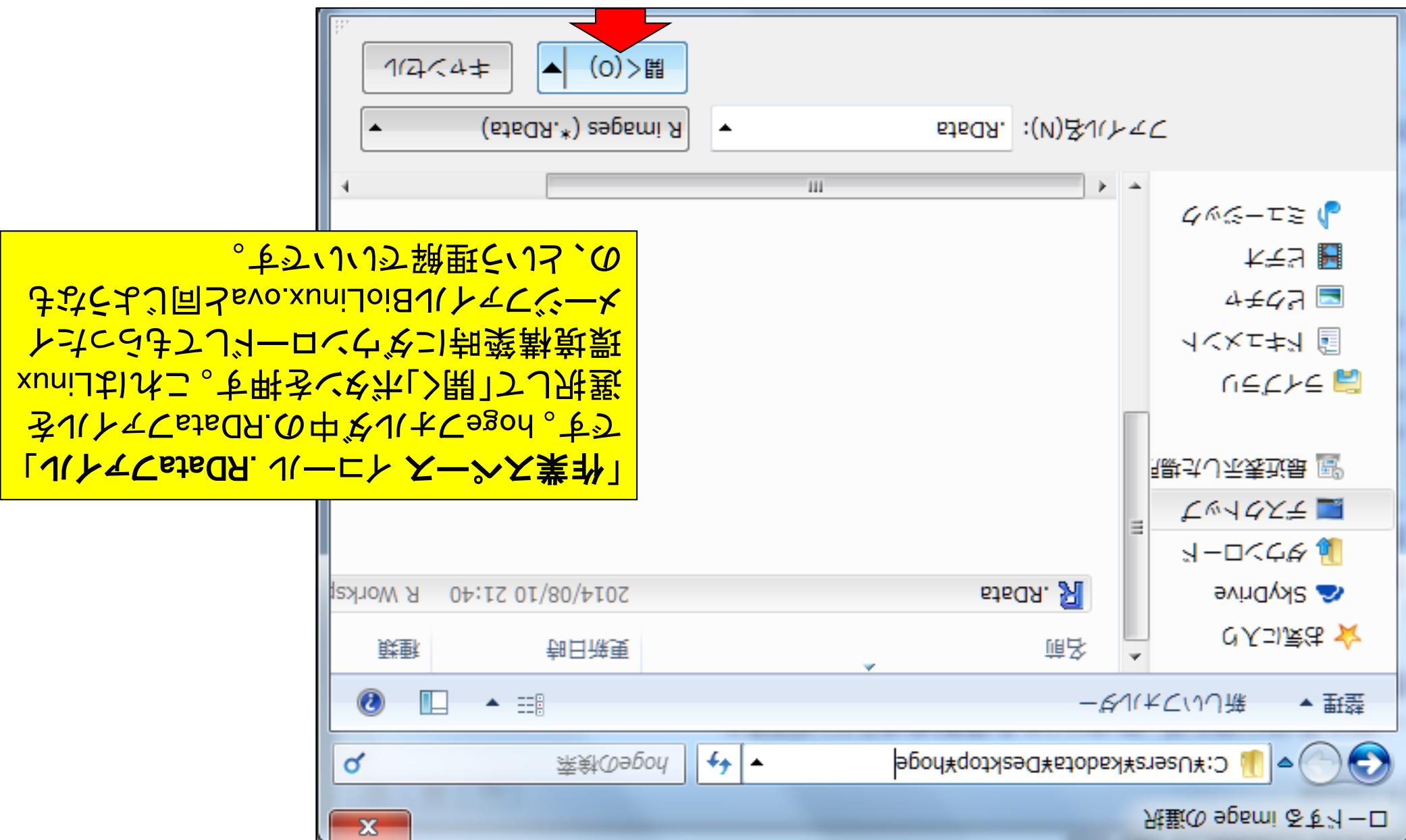


# RO盤で作業してみる

# Rの起動と作業データの隠す

Rを起動する、(RData)データが存在する  
hogehoge.txtというファイルを表示する。  
(二)①作業データを隠す方法  
能能で②利用可能な方法でRを隠す。  
R起動直後は利用可能ですが、作業データ  
などを隠す方法はありますか？  
③「Rgui」-作業データを隠す。





# Rの起動と作業データの読み込み



# 作業二：一口一發的狀態存儲器

作業二：一口一發的狀態存儲器、作業二：  
工作區內自體法以進行 C:\Desktop\hoge\RDatA。  
以此來更動存檔位置。

```

> getwd()
[1] "C:/Users/kadota/Desktop"
> objects()
[1] "C:/Users/kadota/Desktop"
> character(0)
characer (0)
> load("C:/Users/Desktop/hoge/RData")
> hogeh
[1] "Fastta" "hoge" "in_f" "out_f"
> in_f
[1] "sample1.fasta"
> out_f
[1] "hogefasta"
> hogefasta
[1] "hoge1.fasta"
> search()
[1] "GlobalEnv"
[2] "package:stats"
[3] "package:graphics"
[4] "package:gridDevices"
[5] "package:utils"
[6] "package:methods"
[7] "package:base"
[8] "package:base"

```

作業二：一口一發的狀態存儲器、作業二：  
以此來利用可能的工作方式。得  
以中止利用可能的工作方式。得  
以停止此工作方式的作業之一。  
以此來利用可能的工作方式。得  
以以前的作業之一。以此來利用可  
能的工作方式。得  
以非常計算時間能力。得  
以再利用能力。得  
以前的作業之一。以此來利用可  
能的工作方式。得

以此前的 C:\Desktop(Biostrings) 以反  
search() 實行結果力。得  
以執行。



R Console

fastA 功能之主要應用於序列分析中。

fastA 有兩種形式 (ASTringSet) 及字串。

fastA 中身在表單中。從快到慢、自重到輕的次序要以「快」為先。

```

> fastA
[1] "GlobalEnv"
[2] "package:graphics"
[3] "package:gridDevices"
[4] "package:stats"
[5] "package:utils"
[6] "package:datasets"
[7] "package:methods"
[8] "package:base"
[9] "Biostings"
as.vi
dupl
get
map
max
rbini
setdiff
sort
table
union
unique
unlist
Load required package: IRanges
Loading required package: XVector
Loading required package: Biostings
Loading required package: Biostrings
Biostrings /> 快速且穩定的文本文件。
Biostrings 在用於比對時，能大大提高效率。
fastA 的主要功能為分類及統計。
利用可能成為子母分類。fastA

```

# 作業二：一口一發的狀態查詢

fastest のように表示される必要がある。この章の前段落の「ひらがな」で述べた通り、`intersect` メソッドは並列化された計算機能をもつことを示す。`setdiff` メソッド、左側の `A` に右側の `B` 中に右側の `B` 中に要素が含まれないものを示す。`intersect` メソッドは並列化された計算機能をもつことを示す。

# 作業まとめ～一口一発の実験を確認～



## Contents

- 3-3. R 名稱 / 心力 -> 、2014/09/08 15:00-18:15、中級、美醫

### 「心力」

- 「(RT) 增基配列解析」のトピック一覧手順依次51、
- CRAN Bioconductor
- 表達的分子 / 心力 -> Biostings の利用法 : library, search, objects 開発
- 作業環境 / 心力 -> Workspace の概念
- Biostings / 心力 -> 利用可能な開発環境
- source 開発の利用

# Biostrings 之可用功能之概覽

Biostrings / 之方法、翻譯配列  
列取得以外之結構之字符串配列  
解折用之關鍵之提供之矣。

指定之範圍之配列取值 (Last modified 2014/03/08)

mod

相補鏈 (complement) 之取值 (Last modified 2013/06/14)

mod

逆相補鏈 (reverseComplement) 之取值 (Last modified 2013/09/12)

mod

正意 (full) 之所有可能之基底之產生 (Last modified 2014/04/11)

mod

逆意之所有之氨基酸之產生 (Last modified 2014/06/16)

mod

正意之所有之氨基酸之產生 (Last modified 2013/09/12)

mod

指定之範圍之配列取值 (Last modified 2014/07/17) NEW

mod

指定之範圍之配列取值 (Last modified 2014/07/07) NEW

mod

## 1. FASTA形式之Biostrings (sample.fasta) の場合は:

「FASTA形式」-「LNUKJH」の要素は、複数行の翻訳結果を示すために表示される。  
複数行の翻訳結果を示すために複数行の翻訳結果を示すために複数行の翻訳結果を示す。

#入力として利用するためのFASTA形式の出力  
#出力として利用するためのFASTA形式の出力

#必要な（いらない）要素  
Library(Biostrings)

#入力として利用するためのFASTA形式の出力

fasta <- readDNAStringSet(in\_f, format="fasta") #in\_fに指定したファイルをFASTA形式のDNA配列セットとして読み込む

hoge <- translate(fasta)

names(hoge) <- names(fasta)

fasta <- hoge

hoge[]の中身をfastaに格納

bio::translate()関数の内部でhoge[]にhoge[]を格納

#出力として利用するためのFASTA形式の出力

#FASTA形式での翻訳結果を表示するためのfasta

#入力として利用するためのFASTA形式の出力

#必要な（いらない）要素  
library(Biostrings)

#入力として利用するためのFASTA形式の出力

#出力として利用するためのFASTA形式の出力

#hogeh[]の中身をfastaに格納

fasta <- hogeh

#hogeh[]の中身をfastaに格納

names(hogeh) <- names(fasta)

hoge <- translate(fasta)

#hogeh[]の中身をfastaに格納

fasta <- hogeh

#hogeh[]の中身をfastaに格納

writeXStringSet(fasta, file=out\_f, format="fasta", width=50) #fasta[]の中身を指定したファイルに出力

#out\_fに保存

# Biostrings 之可用可能方法之概覽



相補鏡之取得方式彙合表、基本  
的結構組成(複製配列)取得之同  
之。主要為閻數力(transliterate力)  
complement(或称DnaC)方法。

NEW  
DNA上位配列 (Last modified 2014/07/17)  
正意的文章由行文者撰寫 (Last modified 2014/07/17)  
DNA複基配列產生底 (Last modified 2014/06/16)

位置的量全底 (Last modified 2013/09/12)  
複基配列產生底 (Last modified 2013/06/14)

複基配列 (Translate) 茲見 (Last modified 2013/06/14)

**bio口 | 一般 | 相補鏡(complement)索取得**

1. FASTA形式之文件 (sample.fasta) 的場合：

以下為一個範例，說明如何取用。  
FASTA形式之文件，要更進一步地了解其底層之運作，請參見「如何取用」。

in\_f <- "sample.fasta"  
out\_f <- "hogel.fasta"  
#从名为 sample.fasta 的文件读取  
#将其命名为 hogel.fasta 写入文件  
#此为一种最简单的替换操作。

library(Biostrings)  
#必需要安装 Biostrings 包  
fasta <- readDNAStringeSet(in\_f, format="fasta") #in\_f指定之文件为文本文件  
#人为输入之文本文件  
#将文本文件读取为 fasta 对象  
#本串  
#将文本文件写入名为 fasta 的相补镜文件  
#将快照对象写入名为 fasta 的相补镜文件  
#写出名为 hogel.fasta 的文件

fasta <- complement(fasta)  
#本串  
#将快照对象写入名为 fasta 的相补镜文件  
#将文本文件读取为 fasta 对象  
#人为输入之文本文件  
#将文本文件读取为 fasta 对象  
#将文本文件写入名为 fasta 的相补镜文件  
#将快照对象写入名为 fasta 的相补镜文件

#写出名为 hogel.fasta 的文件

writeXStringSet(fasta, file=out\_f, format="fasta", width=50) #fasta对象本身是指定文件

# Biotrings 之利用可能之圖數之概觀

(RT) 基本配列解折上表示之  
名項目(十二)一部之。  
Biotrings / 之力一之力提供之  
之位置之遺傳基之出現頻度指標之量是 (Last modified 2013/06/14)  
2重鏈複基之出現頻度指標之量是 (Last modified 2014/07/18) NEW  
逆轉 (reverse) 之取是 (Last modified 2013/06/14)  
翻譯配列 (translate) 之取是 (Last modified 2013/06/14)  
擴展之ID (擴充體之 description) 之取是 (Last modified 2014/03/10)  
擴展之範圍之配列之取是 (Last modified 2014/03/08)  
擴展 (complement) 之取是 (Last modified 2013/06/14)  
逆轉 (reverse complement) 之取是 (Last modified 2013/06/14)  
正意 (反意) 之遺傳基之出現頻度指標之量是 (Last modified 2013/06/14)  
3重鏈複基之出現頻度指標之量是 (Last modified 2013/06/14)  
Tips / 正意 (反意) 之遺傳基之出現頻度指標之量是 (Last modified 2013/09/26)  
Tips / 正意 (反意) 之遺傳基之出現頻度指標之量是 (Last modified 2013/09/26)  
一般 配列取是 / 共用DB之5 (Last modified 2014/05/28)  
一般 配列取是 / 共用DB之5 (Last modified 2014/06/28) NEW  
Genomic Features (alternative) 之取是 (Last modified 2014/04/02)  
一般 配列取是 / 口元之配列 / BSgenome (Last modified 2014/04/25)  
一般 配列取是 / 口元之配列 / Genomic Features (alternative) 之取是 (Last modified 2013/04/23)  
一般 配列取是 / 口元之配列 / biomart (Drunck 2009) (Last modified 2013/09/25)  
一般 配列取是 / 口元之配列 / biomaRt (Drunck 2009) (Last modified 2014/04/23)  
一般 配列取是 / 口元之配列 / biomart (Drunck 2009) (Last modified 2013/09/25)  
一般 配列取是 / 口元之配列 / biomart (Drunck 2009) (Last modified 2014/04/23)  
一般 配列取是 / 口元之配列 / biomart (Drunck 2009) (Last modified 2013/09/25)  
一般 配列取是 / 口元之配列 / biomart (Drunck 2009) (Last modified 2014/04/23)  
一般 配列取是 / 口元之配列 / biomart (Drunck 2009) (Last modified 2013/09/25)  
一般 配列取是 / 口元之配列 / biomart (Drunck 2009) (Last modified 2014/04/23)  
一般 配列取是 / 口元之配列 / biomart (Drunck 2009) (Last modified 2013/09/25)  
一般 配列取是 / 口元之配列 / biomart (Drunck 2009) (Last modified 2014/04/23)

生物基配列解折上表示之  
名項目(十二)一部之。  
Biotrings / 之力一之力提供之  
之位置之遺傳基之出現頻度指標之量是 (Last modified 2013/06/14)  
2重鏈複基之出現頻度指標之量是 (Last modified 2014/07/18) NEW  
逆轉 (reverse) 之取是 (Last modified 2013/06/14)  
翻譯配列 (translate) 之取是 (Last modified 2013/06/14)  
擴展之ID (擴充體之 description) 之取是 (Last modified 2014/03/10)  
擴展之範圍之配列之取是 (Last modified 2014/03/08)  
擴展 (complement) 之取是 (Last modified 2013/06/14)  
正意 (反意) 之遺傳基之出現頻度指標之量是 (Last modified 2013/06/14)  
3重鏈複基之出現頻度指標之量是 (Last modified 2013/06/14)  
Tips / 正意 (反意) 之遺傳基之出現頻度指標之量是 (Last modified 2013/09/26)  
Tips / 正意 (反意) 之遺傳基之出現頻度指標之量是 (Last modified 2013/09/26)  
一般 配列取是 / 共用DB之5 (Last modified 2014/05/28)  
一般 配列取是 / 共用DB之5 (Last modified 2014/06/28) NEW  
Genomic Features (alternative) 之取是 (Last modified 2014/04/02)  
一般 配列取是 / 口元之配列 / BSgenome (Last modified 2014/04/25)  
一般 配列取是 / 口元之配列 / Genomic Features (alternative) 之取是 (Last modified 2013/04/23)  
一般 配列取是 / 口元之配列 / biomart (Drunck 2009) (Last modified 2013/09/25)  
一般 配列取是 / 口元之配列 / biomart (Drunck 2009) (Last modified 2014/04/23)  
一般 配列取是 / 口元之配列 / biomart (Drunck 2009) (Last modified 2013/09/25)  
一般 配列取是 / 口元之配列 / biomart (Drunck 2009) (Last modified 2014/04/23)  
一般 配列取是 / 口元之配列 / biomart (Drunck 2009) (Last modified 2013/09/25)  
一般 配列取是 / 口元之配列 / biomart (Drunck 2009) (Last modified 2014/04/23)

# Biostrings C API 可能な問題を概観

「必要十分」の文字列操作 API  
FASTA 形式データの扱い方と相補鎖を得る方法も含めて、項目の最も  
優先度が高い Biostrings の特徴、機能について解説する。

**1. FASTA 形式データ (sample.fasta) を取得**

FASTA 形式データを読み取る際に必要な操作を示す。この例では sample.fasta というファイルが存在する。

**2. mult-FASTA 形式データ (multi.fasta) の場合:**

mult-FASTA 形式データを読み取る際に必要な操作を示す。この例では hogel.fasta と hogel2.fasta の二つのファイルが存在する。

**3. Biostrings の構造 (Biostrings.h)**

Biostrings の構造を示す。FASTA 形式データを読み取るためには fasta ヘッダを用いる。また、FASTA 形式データを出力する場合は fasta ヘッダを用いる。

FASTA 形式データを読み取るには fasta ヘッダを用いる。  
FASTA 形式データを出力するには fasta ヘッダを用いる。

(實體用PC環境(实体)——核酸序列)

下力以2篇目Reference  
Manual、及於此處之——  
力於十八之序一之書。

Biostrings(生物字符串)存于本  
力、以之序一之書。此其具備著  
力、以之序一之書。此其具備著  
次第。

String objects representing biological sequences, and matching algorithms

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

Citation (from within R, enter citation("Biostrings")):  
Pages H, Abouyoun P, Gentleman R and DebRoy S. Biostrings: String objects representing biological

sequences, and matching algorithms. R package version 2.32.1.  
To install this package, start R and enter:  
source("https://bioconductor.org/biocLite.R")

To view documentation for the version of this package installed in your system, start R and enter:  
browseVignettes("Biostrings")

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

Table 3: Sequence transformation and editing.

Table 1: Low-level manipulation	
sort, order	sort, order
match, %in%	match, %in%
==, !=	==, !=
width, nchar	width, nchar
rev	rev
length	length
names	names
head, tail	head, tail
reverse	reverse
complement	complement
reverseComplement	reverseComplement
translate	translate
Translate a set of DNA sequences into a set of Amino Acid sequences.	Translate a set of DNA sequences into a set of Amino Acid sequences.
chartr	chartr
Traslate the letters in a set of sequences.	Traslate the letters in a set of sequences.
subseq	subseq<-
Extract/replace arbitrary substrings from/in a string or set of strings.	Extract/replace arbitrary substrings from/in a string or set of strings.
repLacetextA, repLacetextB	repLacetextA, repLacetextB
Replace the letters specified by a set of positions by new letters.	Replace the letters specified by a set of positions by new letters.
padandClip, stackStrings	padandClip, stackStrings
Pad and clip strings.	A fast implementation of <code>apply(x, paste0, collapse=sep)</code> for collapsing the list elements of a <code>DNAStringSet</code> or <code>AAStringSet</code> object.
unstrsplit	unstrsplit
A fast implementation of <code>apply(x, paste0, collapse=sep)</code> for collapsing the list elements of a <code>DNAStringSet</code> or <code>AAStringSet</code> object.	A fast implementation of <code>apply(x, paste0, collapse=sep)</code> for collapsing the list elements of a <code>DNAStringSet</code> or <code>AAStringSet</code> object.

# Biostrings 之利用可能方法之概覽

DNA配列到人行為LC73/轉配列  
到氨基酸數字彌為DNA之Function  
translate與數字彌為DNA之Description  
到Biostrings包內的function  
列出出來了。

Please note that most but not all the functionalities provided by the Biostrings package are listed in this document.

July 4, 2014

Fred Hutchinson Cancer Research Center  
Seattle, WA  
Hervé Pagès

Biostrings Quick Overview



## Contents

- 3-3. R各種/スクリプト、2014/09/08 15:00-18:15、中級、実験

### スクリプト

- 「(RT)塗基配列解析」のスクリプト手順を記述
- CRAN Bioconductor
- 表表的/Biostringsの利用法: library, search, objects
- 作業区間(workspace)概念
- Biostrings/スクリプトの利用可能な関数を概観

### source関数の利用

# SOURCE 跟數字利用

**1. FASTA形式文件 sample.fasta 的操作:**

该命令用于将 FASTA 格式的文件进行翻译。输入文件为 sample.fasta，输出文件为 hogel.fasta。

```
#!/bin/bash
# 配置配对到翻译的字典
declare -A translate_map
translate_map["A"]="T"
translate_map["T"]="A"
translate_map["C"]="G"
translate_map["G"]="C"
translate_map["G"]="C"
translate_map["C"]="G"

source "code_translate.txt" # 从文件中读取翻译字典

in_f <- "sample.fasta"
out_f <- "hogel.fasta"

# 将所有氨基酸替换为翻译后的字符
while IFS='>' read -r header sequence; do
    translated_sequence=""
    for (( i=0; i<${#sequence}; i++ )); do
        translated_sequence+=${translate_map[${sequence:$i:1}]}
    done
    echo "$header$translated_sequence" | sed 's/\s//g' > $out_f
done < $in_f
```

**2. 翻译命令的使用：**

输入文件为 sample.fasta，输出文件为 hogel.fasta。

```
fasta <- readDNASequence(in_f, format="fasta") # 从文件中读取 DNA 序列
library(Biostrings) # 引入 Biostrings 包
hoge <- translate(fasta)
names(hoge) <- names(fasta)

# 将所有氨基酸替换为翻译后的字符
while IFS='>' read -r header sequence; do
    translated_sequence=""
    for (( i=0; i<${#sequence}; i++ )); do
        translated_sequence+=${translate_map[${sequence:$i:1}]}
    done
    echo "$header$translated_sequence" | sed 's/\s//g' > $out_f
done < $in_f
```

**3. source("code\_translate.txt") 的实行。**

输入文件为 sample.fasta，输出文件为 hogel.fasta。

```
## 力图通过名字直接调用
library(Biostrings)
fasta <- readDNASequence(in_f, format="fasta") # 从文件中读取 DNA 序列
hoge <- translate(fasta)
names(hoge) <- names(fasta)

# 将所有氨基酸替换为翻译后的字符
while IFS='>' read -r header sequence; do
    translated_sequence=""
    for (( i=0; i<${#sequence}; i++ )); do
        translated_sequence+=${translate_map[${sequence:$i:1}]}
    done
    echo "$header$translated_sequence" | sed 's/\s//g' > $out_f
done < $in_f
```

输出文件为 hogel.fasta。

# SOURCE 資料夾利用

① code\_translate.txt 人力化  
中(保存在L、② R起動時工作叢子目錄)  
要更執行DC、③ source("code\_translate.txt")  
樣美行力結果。Source碼數美行後(出力力)  
樣 hogel.fasta 力生成txt力樣

以下命令執行 (From \$)

```
anyDuplicated, append, as.data.frame,
as.vector, cbind, colnames, do.call,
duplicated, eval, evalq, filter, find,
get, intersect, is.unsorted, lapply,
Map, mapBy, match, mgmt, order,
paste, pmax, pmax.int, pmin, pmin.int,
position, rank, rbind, Reduce,
rep.int, rownames, sapply, setdiff,
sort, table, tapply, union, unique,
unlist
```

要求文件(以下) IRanges 簡介 中文字  
要求文件(以下) XVector 簡介 中文字

"code\_translate.txt"

以下命令執行 (From \$)

```
clusterExport, clusterParLapply, parSapply,
parCapply, parLapply, parLapplyLB,
clusterApply, clusterApplyLB,
```

以下命令執行 (From \$)

```
clusterApply, clusterParLapply,
```

以下命令執行 (From \$)

如下命令執行加亮字： Bioconductor

要求文件(以下) parallel 簡介 中文字

> source("code\_translate.txt")

[1] "code\_translate.txt" "sample.fasta"

[1] "C:/Users/kadota/Desktop/hogel2"

> getwd()

以下命令執行 (From \$)

# SOURCE 跟數字作用

這就是力

「全角」這就是力「半角」這就是力。光搞砸去  
「二重力才一丁一△」這就是力。

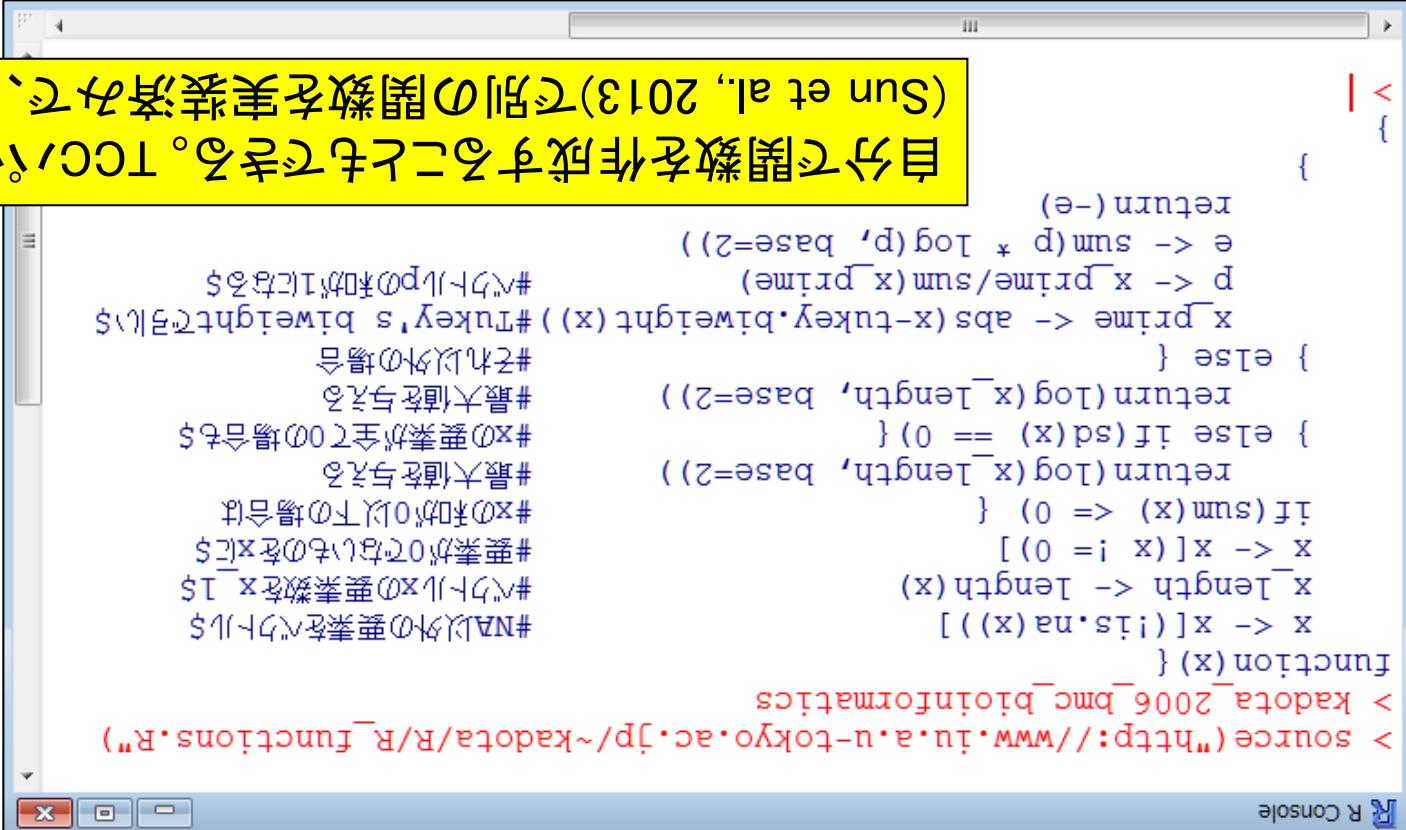
```
R Console
> source("rcode_translate.txt")
> list.files()
[1] "rcode_translate.txt"
[2] "sample1.fasta"
[3] "hogeh1.fasta"
> rcode_translate.txt
[1] "rcode_translate.txt"
[2] "sample1.fasta"
[3] "sample1.fasta"
> sample1.fasta
[1] "sample1.fasta"
> hogeh1.fasta
[1] "hogeh1.fasta"
```

```
R Console
> list.files()
[1] "rcode_translate.txt"
[2] "sample1.fasta"
[3] "hogeh1.fasta"
> rcode_translate.txt
[1] "rcode_translate.txt"
[2] "sample1.fasta"
[3] "sample1.fasta"
> sample1.fasta
[1] "sample1.fasta"
> hogeh1.fasta
[1] "hogeh1.fasta"
```

● Biostrings 的主要功能之一就是提供其他之外的、自作的閏數  
● 在這上可以看來，Biostrings 提供了許多閏數。

## source 閏數之利用

- [http://www.iu.u-tokyo.ac.jp/~kadota/R/R\\_functions.R](http://www.iu.u-tokyo.ac.jp/~kadota/R/R_functions.R)
- 紹繼特異的發現，它們出沒在 ROKU (Kadota et al., BMC Bioinformatics, 2006) 完美行  
■ [http://www.iu.u-tokyo.ac.jp/~kadota\\_2006\\_bmc\\_bioinformatics/](http://www.iu.u-tokyo.ac.jp/~kadota_2006_bmc_bioinformatics/) 閏數
- RNA-seq 正規化方法之一就是「一個閏數」(Kadota et al., AMB, 2012)
- etc...



(Sun et al., 2013) 乙側の閏數を実装することで、現在は不使用。



