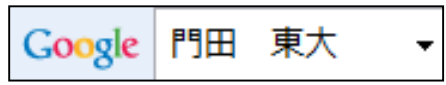
バイオインフォマティクス人材育成カリキュラム（次世代シークエンサ）速習コース
3. データ解析基礎 | 3-2. R 基礎2

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
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


http://www.iu.a.u-tokyo.ac.jp/~kadota/r_seq.html
(Rで)塩基配列解析
(Rで)塩基配列解析
~NGS、RNA-seq、ゲノム、トランスクリプトーム、正規化、発現変動、統計、モデル、バイオイ
(last modified 2014/08/22, since 2010)
What's new?
• 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 | 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
• イントロ | 一般 | 相補鎖(complement)を取得
• 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 | トランスクリプトーム配列 | 公共DBから(last modified 2014/04/02)
• イントロ | 一般 | 配列取得 | トランスクリプトーム配列 | 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などとの比較(last modified 2014/07/11) NEW
• イントロ | NGS | 可視化(ゲノムブラウザやViewer)(last modified 2014/06/25) NEW
• イントロ | NGS | 配列取得 | FASTQ or SRALite | 公共DBから(last modified 2014/06/28) NEW
• イントロ | NGS | 配列取得 | FASTQ or SRALite | SRAdb(Zhu 2013)(last modified 2014/06/26) NEW

- イントロ | 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descrip[illegible]n)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([illegible]modified 2013/06/14)
- イントロ | 一般 | 相補鎖(complement)を取得([illegible]st 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 |
- イントロ | 一般 |
- イントロ | 一般 |
- イントロ | 一般 |
- イントロ | 一般 |
- イントロ | 一般 |
- イントロ | 一般 |
- イントロ | 一般 |
- イントロ | 一般 |
- イントロ | 一般 |
- イントロ | NGS |
- イントロ | NGS |
- イントロ | NGS |
- イントロ | NGS |
- イントロ | NGS |

• 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

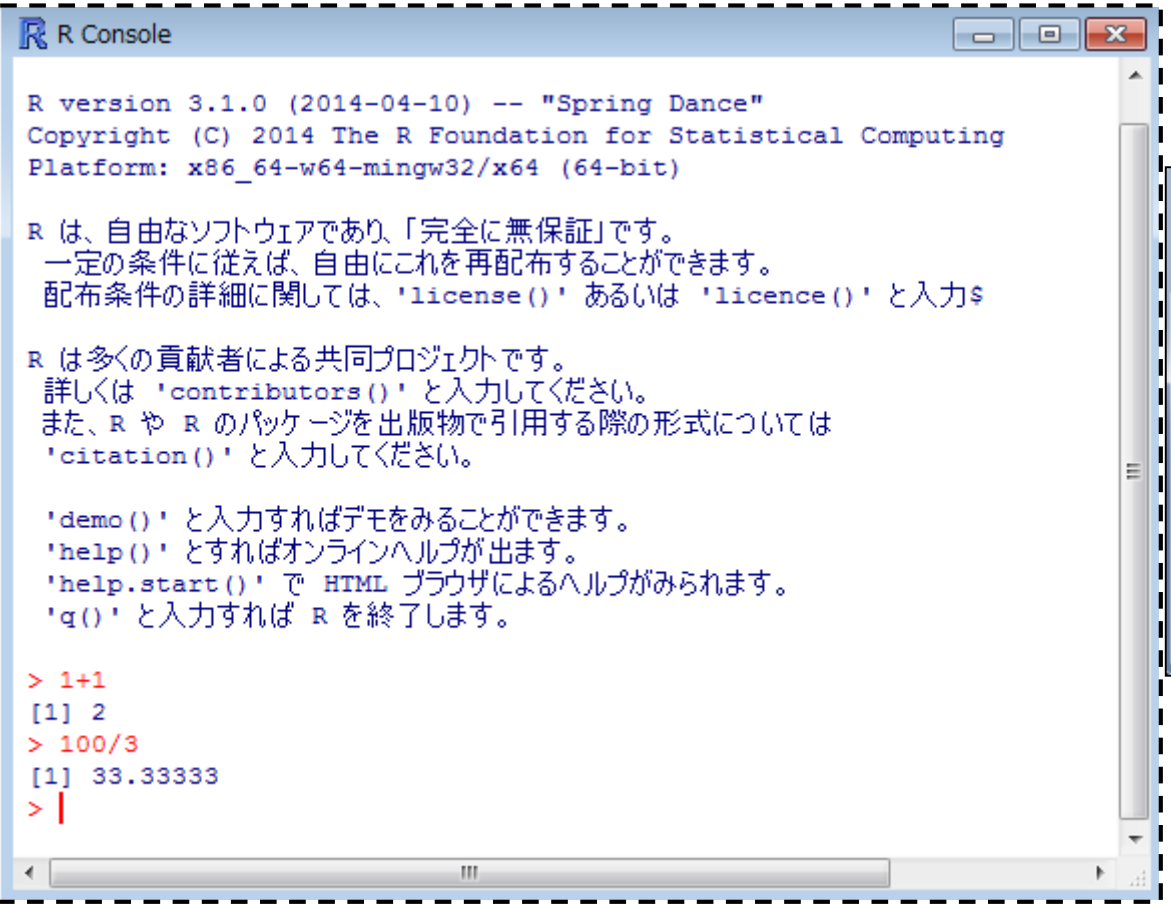
塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

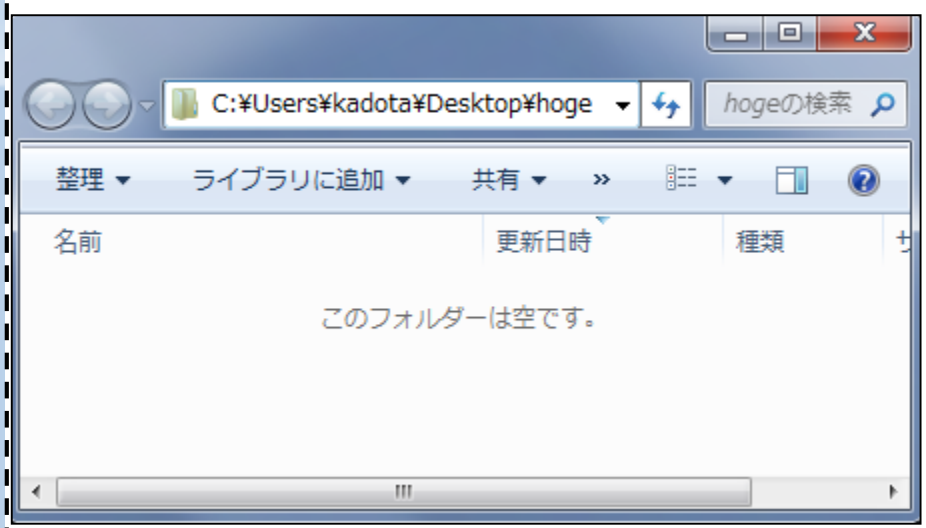
1. FASTA形式ファイル(sample1.fasta)の場合:

```
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descript
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
```

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





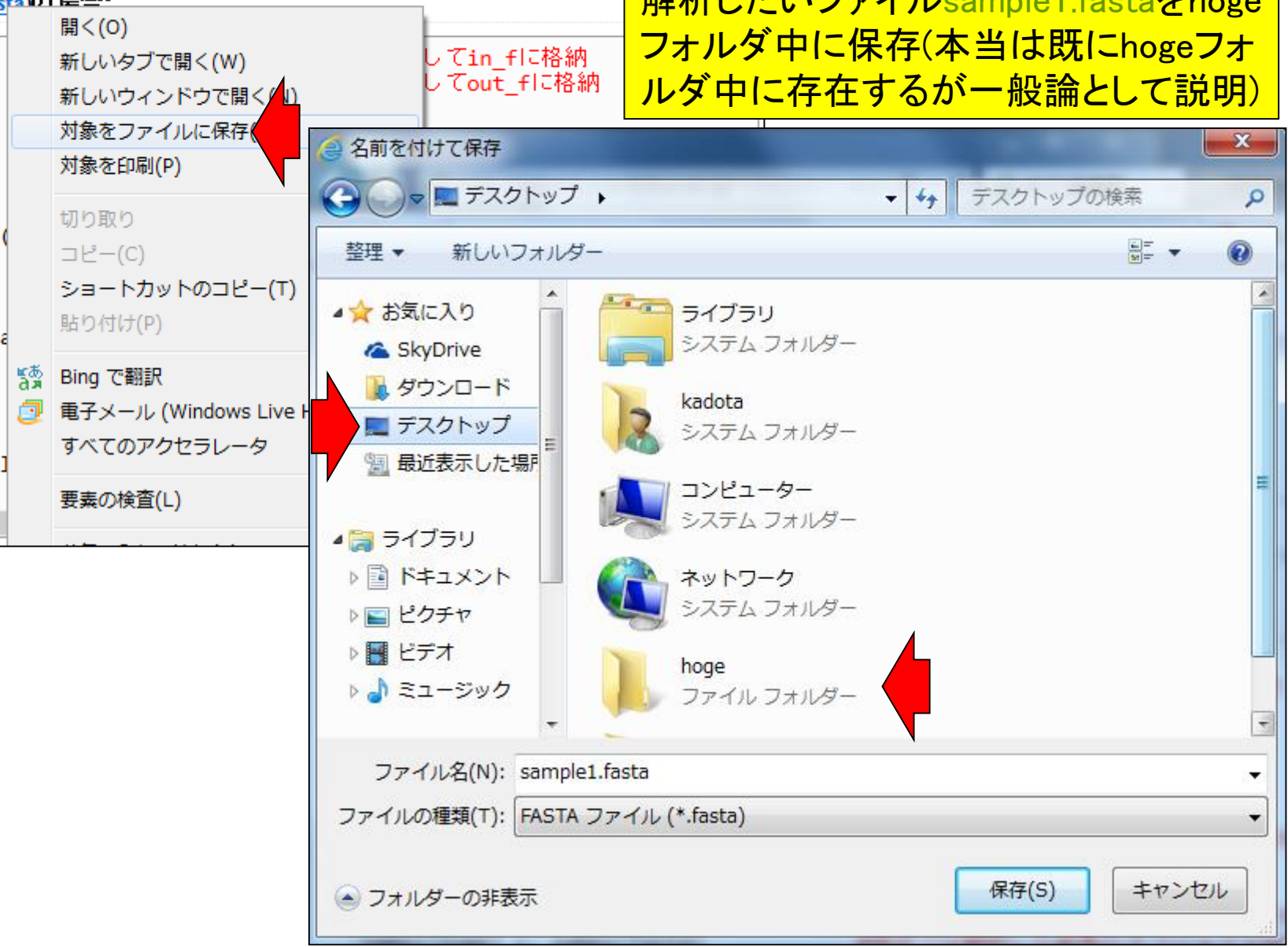
イントロ | 一般 | 翻訳配列(translate)を取得 NEW

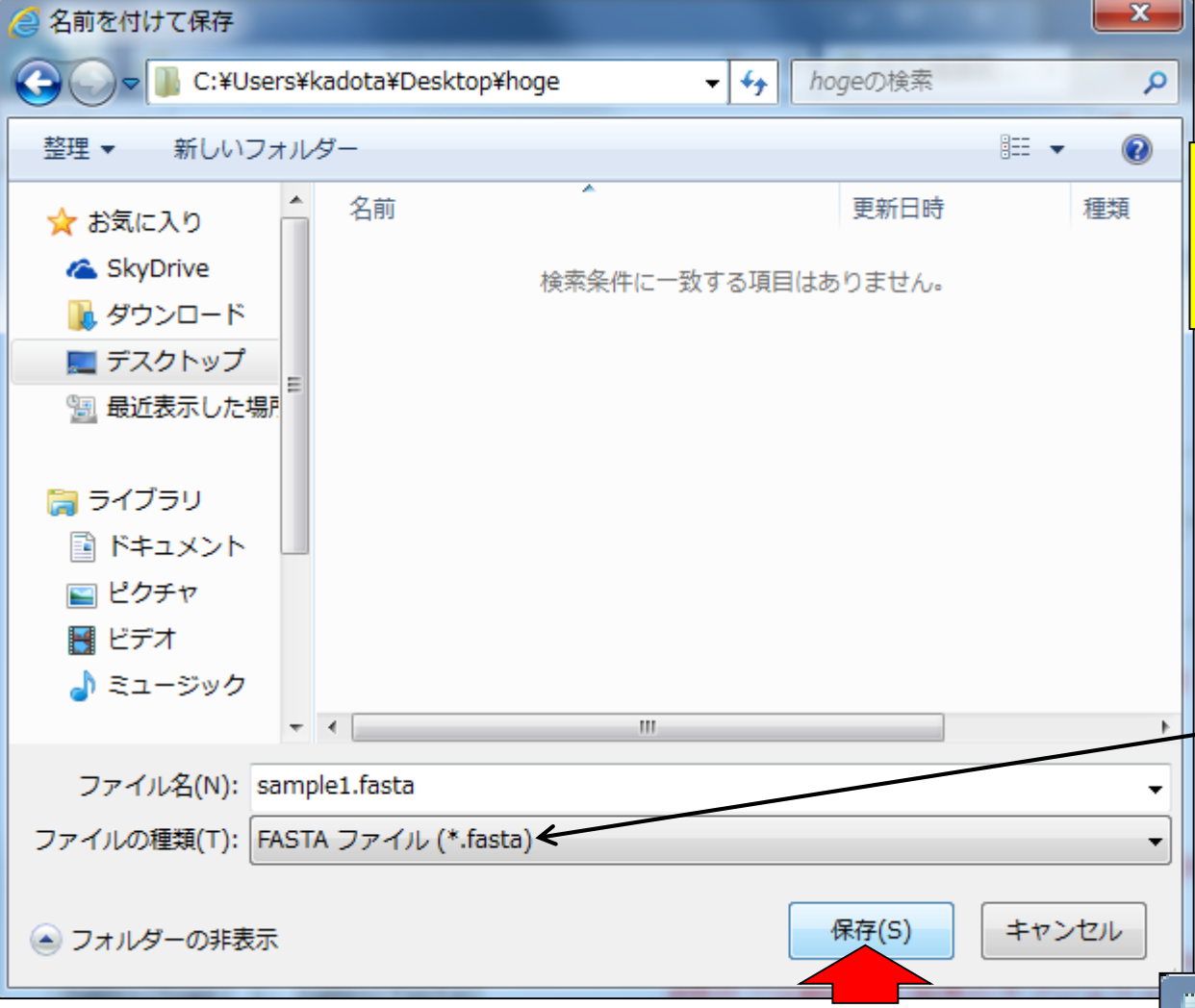
塩基配列を読み込んでアミノ酸配列に翻訳するやり方を示します。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

1. FASTA形式ファイル(sample1.fasta)の場合:

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

#必要なパッケージをロード
library(Biostrings)

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

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

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

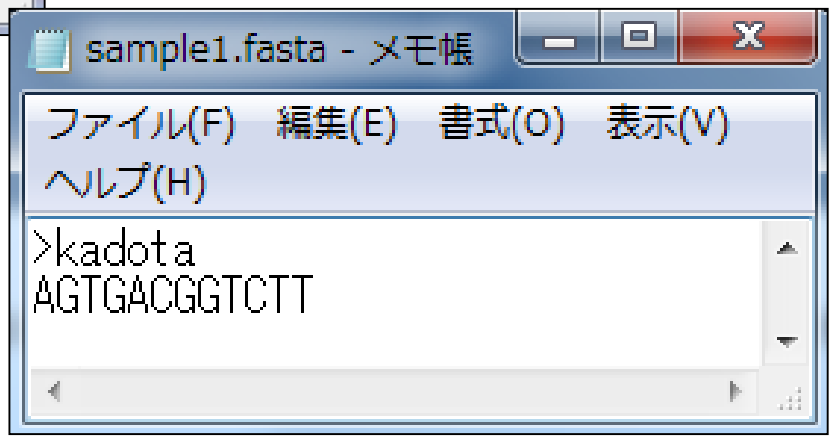

解析したいファイルsample1.fastaをhogeフォルダ中に保存(本当は既にhogeフォルダ中に存在するが一般論として説明)





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)

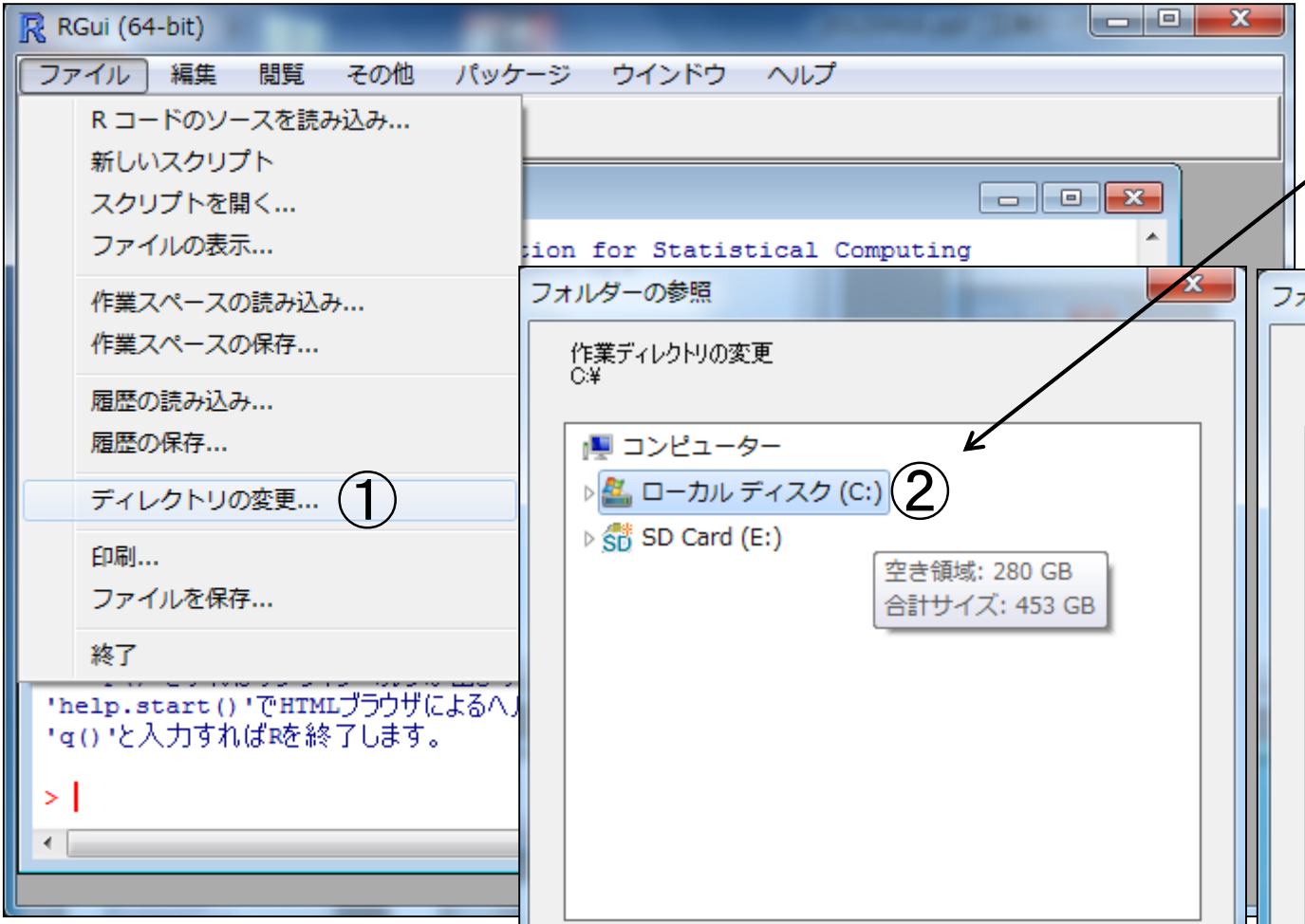
ときどき拡張子が*.txtなどと勝手に変わっていることがあるのでファイルの種類欄に注意すべし

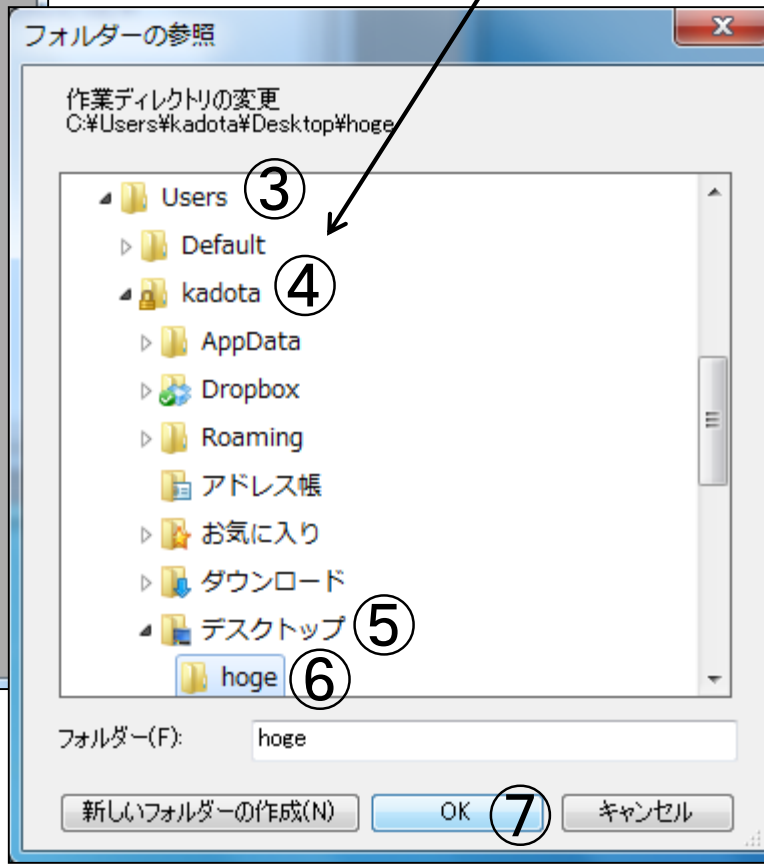


作業ディレクトリの変更

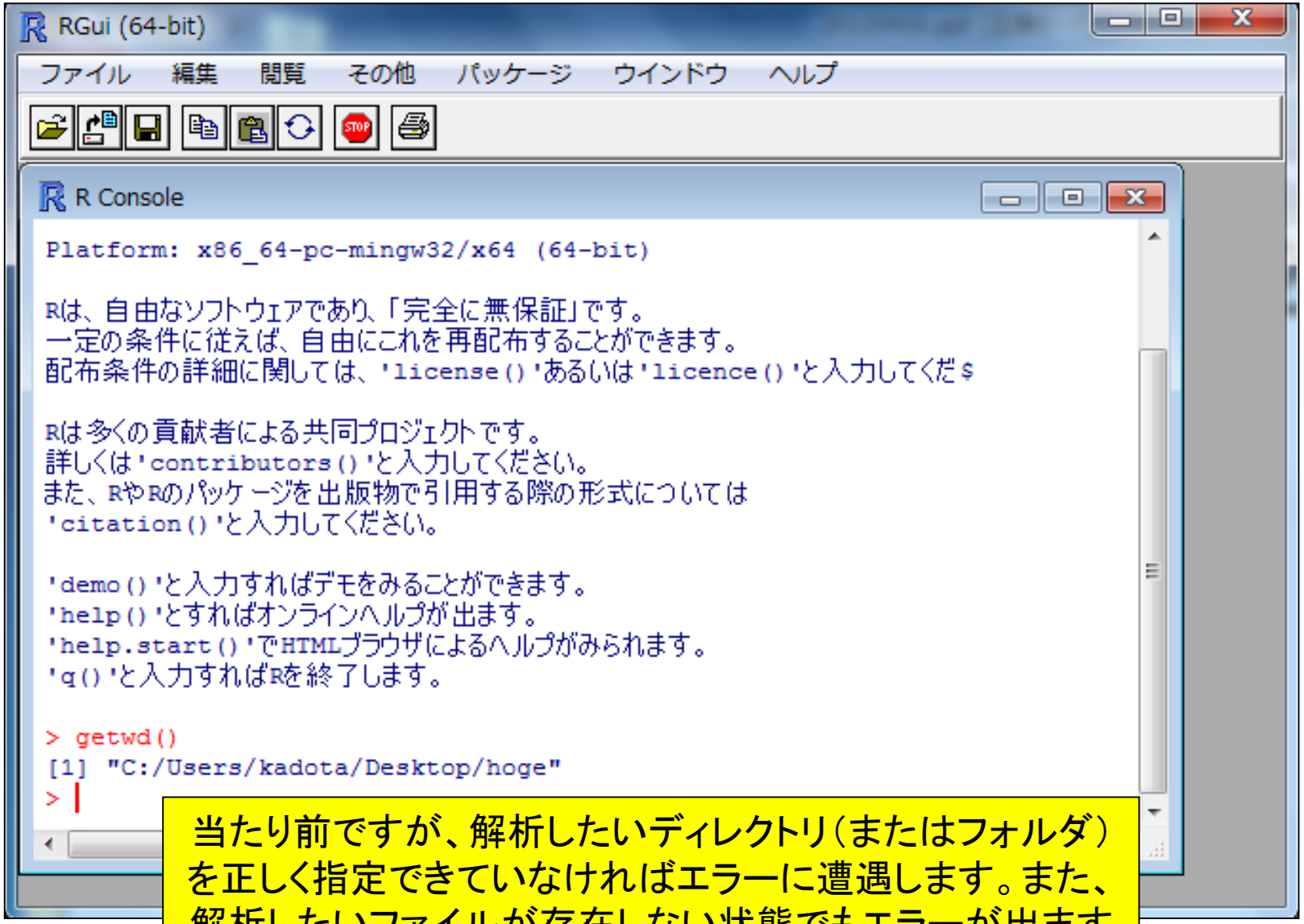
「Windows(C:)」となっている場合もあるが、気にしない

④はヒトそれぞれ





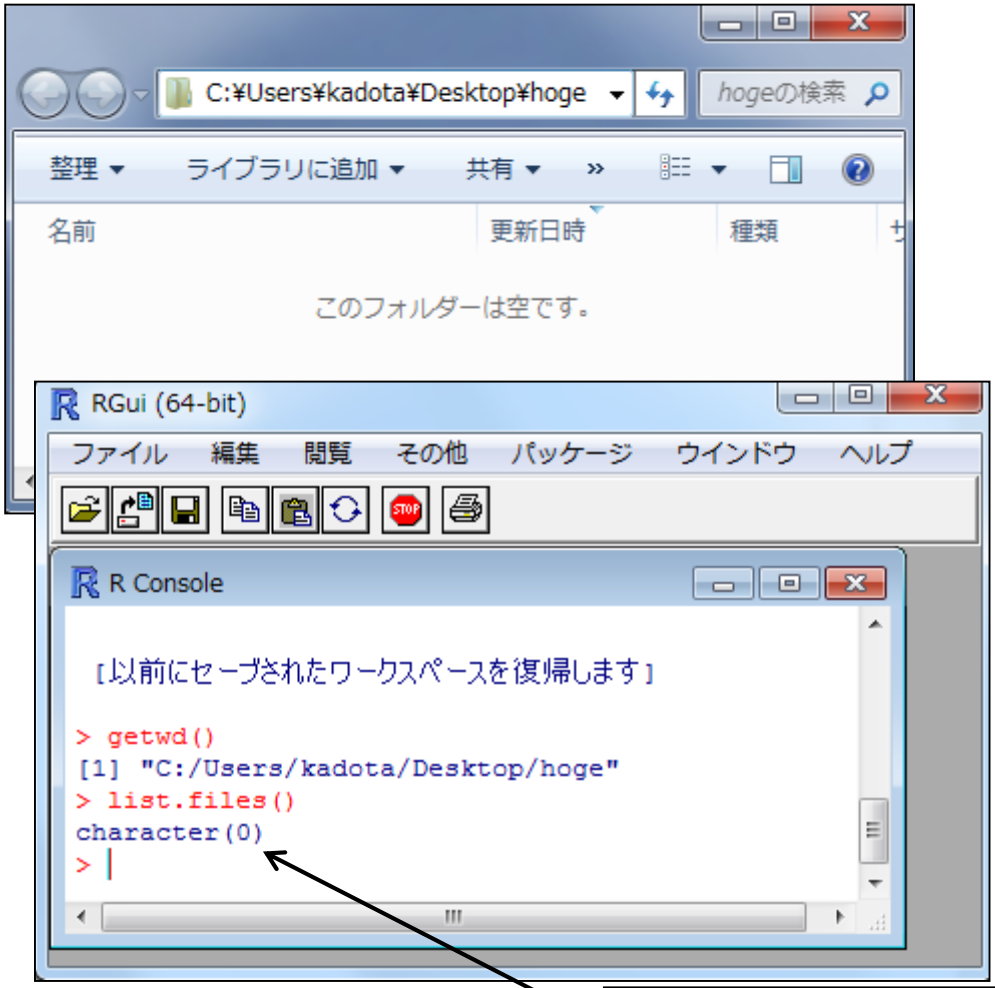
getwd()と打ち込んで確認



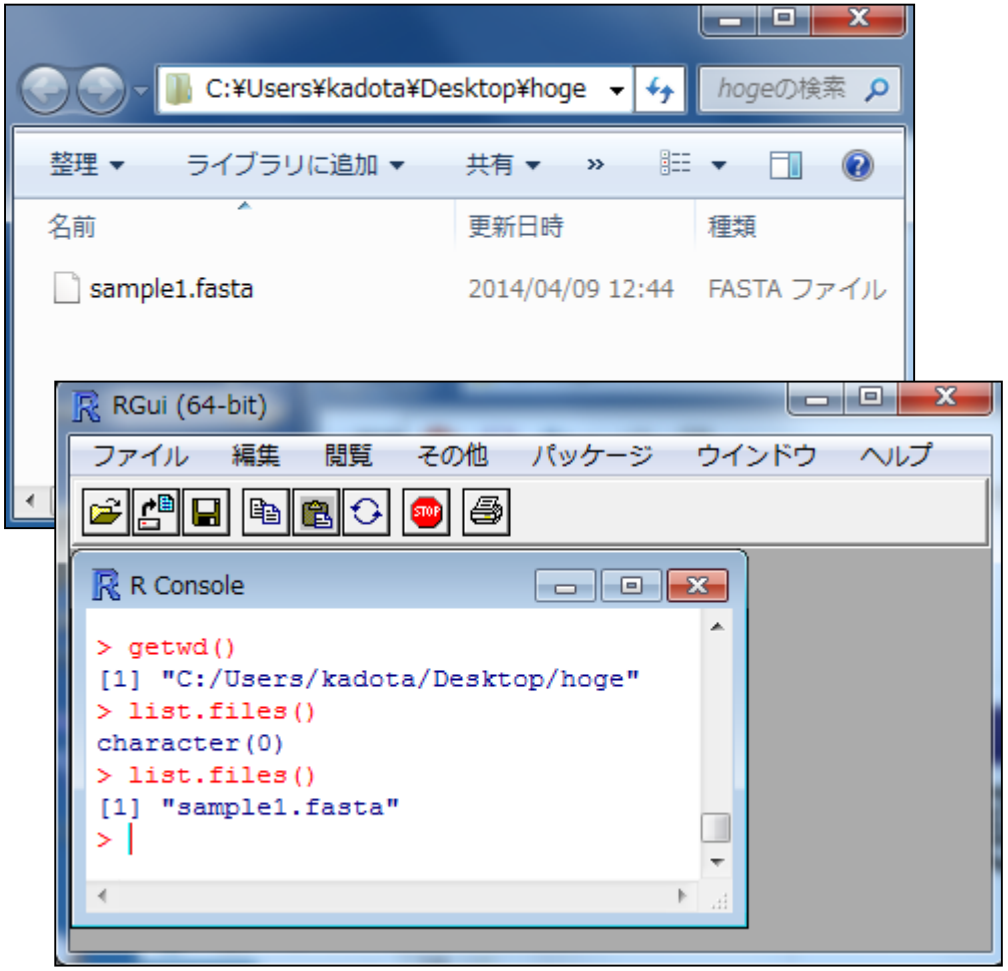
当たり前ですが、解析したいディレクトリ（またはフォルダ）を正しく指定できていなければエラーに遭遇します。また、解析したいファイルが存在しない状態でもエラーが出ます

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

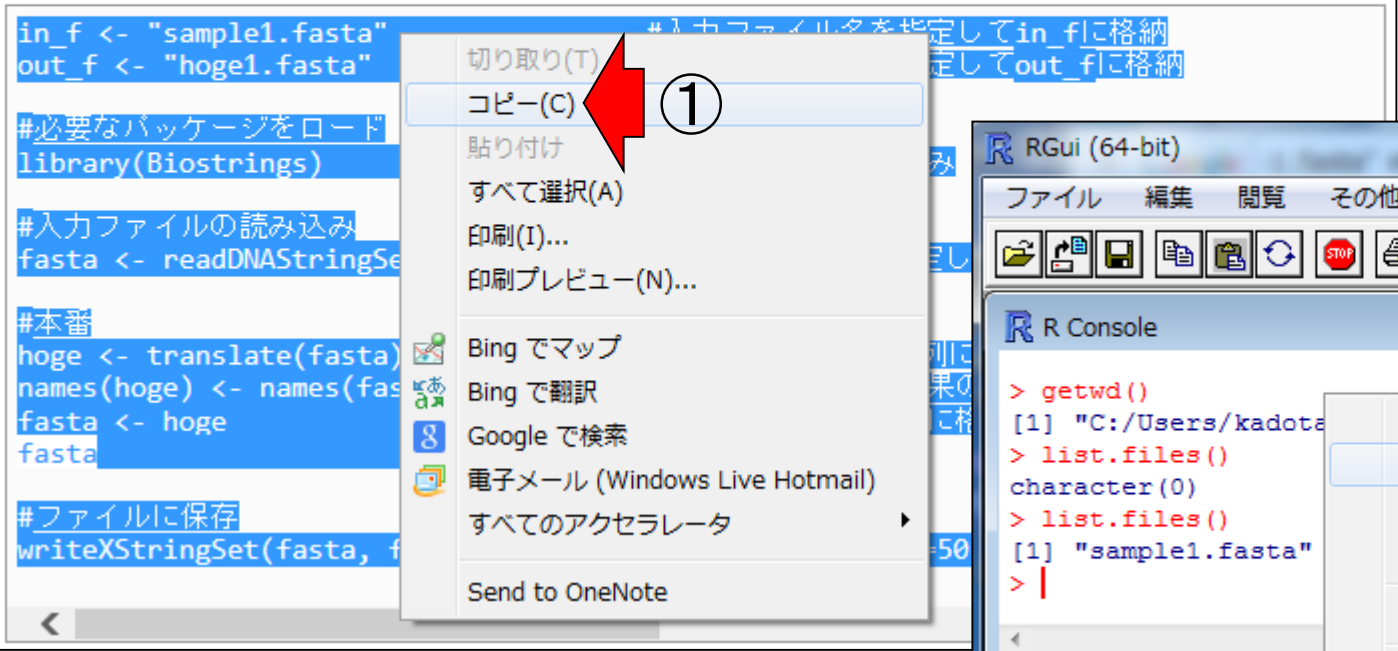
Windowsのヒトは、CTRLとALTキーを押しながらコードの枠内で**左クリック**すると、全選択できます。
Macintoshはよくわかりません。

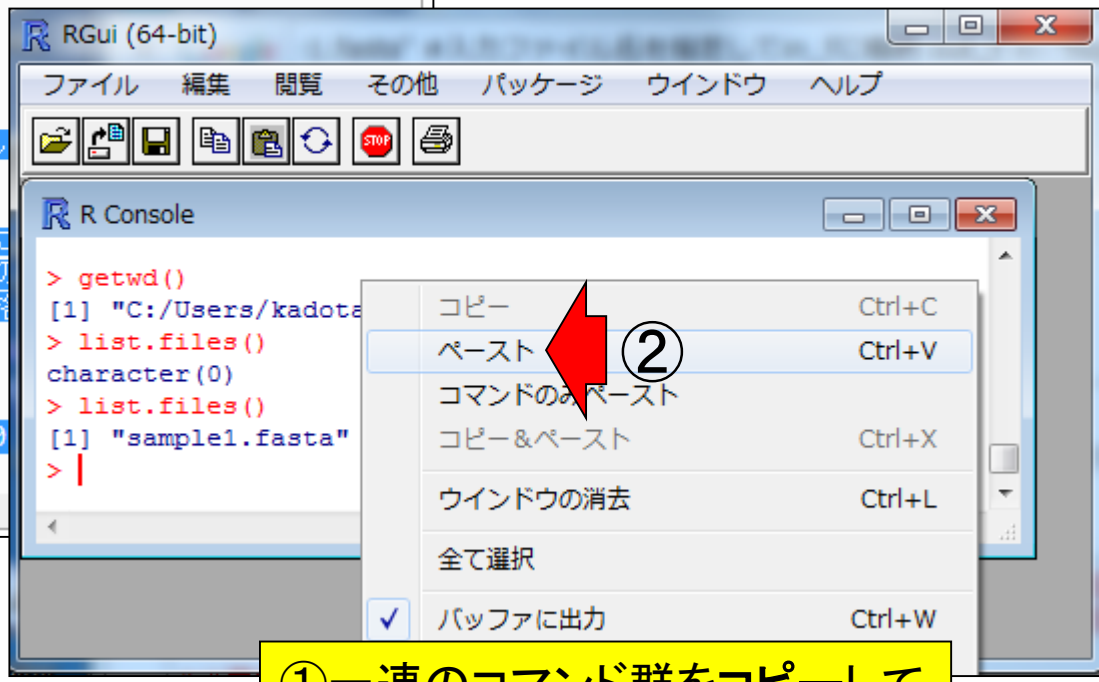
イントロ | 一般 | 翻訳配列(translate)を取得 NEW

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

#必要なパッケージをロード
library(Biostrings)

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

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

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





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**

R Console

```
> 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
 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$
> fasta <- hoge                              #hogeの中身をfastaに格納
> fasta                                      #確認してるだけです
  A AAStringSet instance of length 1
    width seq                                               names
[1]     4 SDGL                                              kadota
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=50)#fastaの中身を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

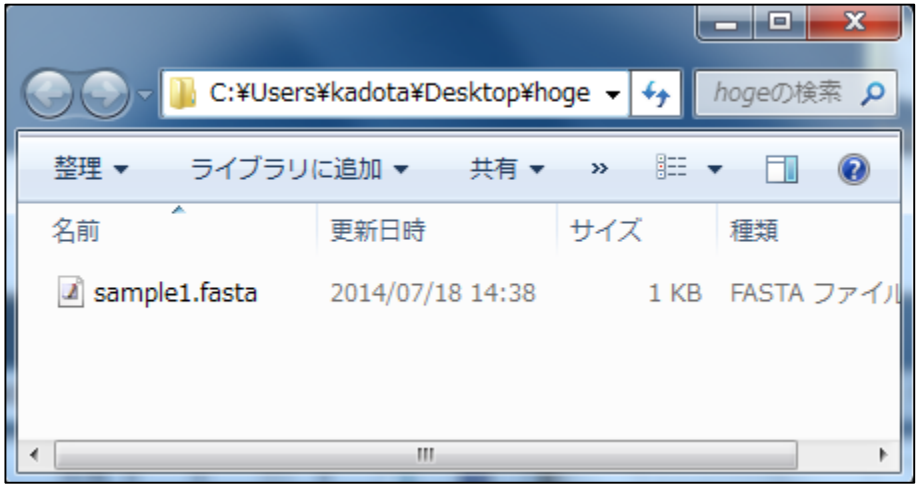
実行結果

R Console

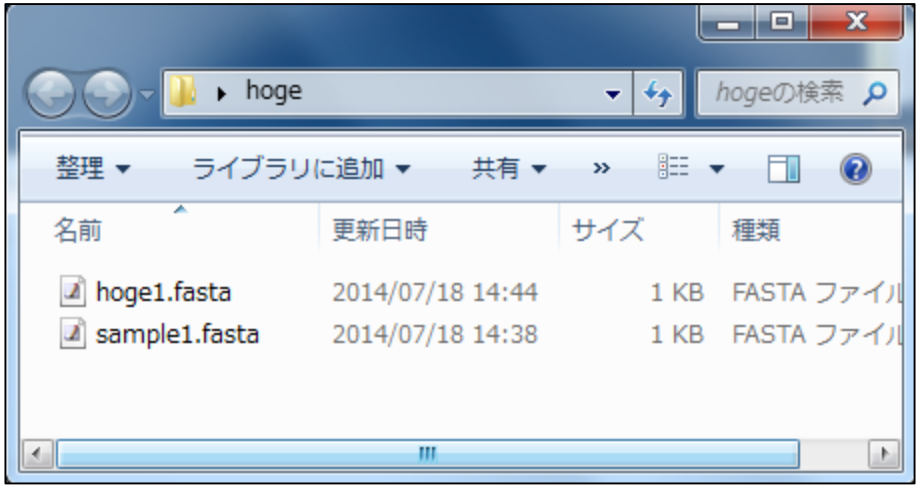
```
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fast$
>
> #本番
> hoge <- translate(fasta)                    #fast$
> names(hoge) <- names(fasta)                 #現状$
> fasta <- hoge                               #hoge$
> fasta                                       #確認$
  A AAStringSet instance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

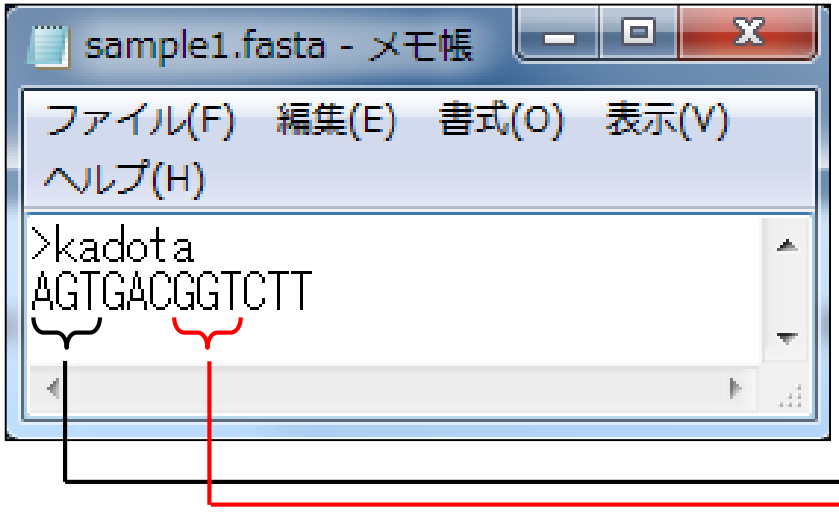


実行後のhogeフォルダ

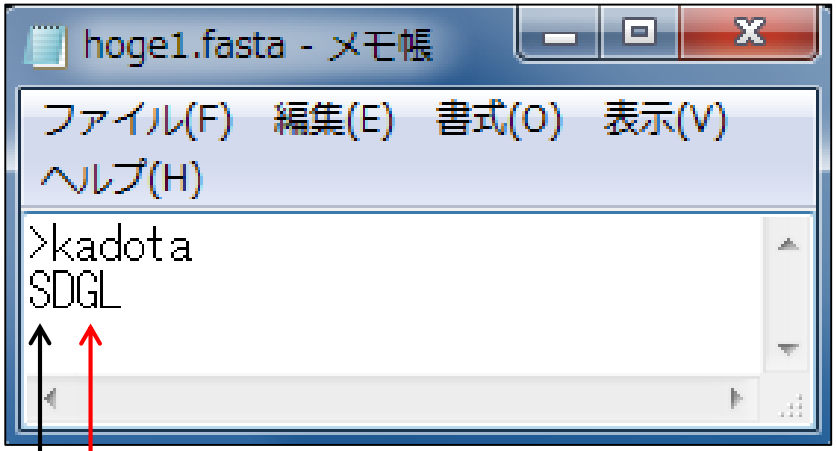


実行結果

入力：塩基配列ファイル（sample1.fasta）



出力：アミノ酸配列ファイル（hoge1.fasta）



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

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descri…en)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(…modified 2013/06/14)
- イントロ | 一般 | 相補鎖(complement)を取得(…st 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:

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(ショウgene_4)の17番目のポンションが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
fasta                                      #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descript
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
```

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。

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

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

#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
fasta

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

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

R Console

```
> #本番
> hoge <- translate(fasta)                 #fastaをアミノ酸配列に翻$
 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$
 以下にエラー names(hoge) <- names(fasta) : オブジェクト 'hoge' $
> fasta <- hoge                            #hogeの中身をfastaに格納
 エラー: オブジェクト 'hoge' がありません
> fasta                                    #確認してるだけです
  A DNAStringSet instance of length 5
    width seq                                        names
[1]    21 CGACAGCTCCTCGGCATCCGA                      gene_1
[2]    27 GTCTGCCTCAAGCGCCCCAAGTGGGTT                gene_2
[3]    21 TGTAGGAGAAGGGCGTAATCT                      gene_3
[4]    30 CGTGCTGATTCCACACNGCAGTAAACGCGG             gene_4
[5]    30 CGTGCTGATTCCACACAGCAGTAAACGCGG             gene_5
>
> #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↓
TGTAGGAGAAGGGCGTAATCT↓
>gene_4↓
CGTGCTGATTCCACACNGCAGTAAACGCGG↓
>gene_5↓
CGTGCTGATTCCACACAGCAGTAAACGCGG↓
←
```

出力：hoge4.fasta

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

この結果がおかしいと思えるようになりましょう。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↓
CGTGCTGATTCCACACNGCAGTAAACGCGG↓
>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$
 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$
 以下にエラー names(hoge) <- names(fasta) :  オブジェクト 'hoge' $
> fasta <- hoge                               #hogeの中身をfastaに格納
 エラー:  オブジェクト 'hoge' がありません
> fasta                                       #確認してるだけです
  A DNAStringSet instance of length 5
    width seq                                       names
[1]    21 CGACAGCTCCTCGGCATCCGA                    gene_1
[2]    27 GTCTGCCTCAAGCGCCCCAAGTGGGTT              gene_2
[3]    21 TGTAGGAGAAGGGCGTAATCT                    gene_3
[4]    30 CGTGCTGATTCCACACNGCAGTAAACGCGG           gene_4
[5]    30 CGTGCTGATTCCACACAGCAGTAAACGCGG           gene_5
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=50)#fa$
>
```

- 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descri…en)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(…modified 2013/06/14)
- イントロ | 一般 | 相補鎖(complement)を取得(…st 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

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。

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」ー「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. FAS…

```
in_f <
out_f

#必要な
librar

#入力フ
fasta

#本番
hoge <
names(
fasta
fasta

#ファイ
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
fasta                                      #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
fasta <- fasta[obj]                        #objがTRUEとなる要素のみ抽出した結果をfastaに格
fasta                                      #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
fasta <- hoge                              #hogeの中身をfastaに格納
fasta                                      #確認してるだけです
```

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
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
out_f <- "hoge5.fasta"

#必要なパッケージをロー
library(Biostrings)

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

#前処理(ACGTのみからな
hoge <- rowSums(alphab
obj <- (width(fasta) =
fasta <- fasta[obj]
fasta

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

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。

R Console

```
> #前処理(ACGTのみからなる配列を抽出)
> hoge <- rowSums(alphabetFrequency(DNAStringSet(fasta))[,1:4])#A$
> obj <- (width(fasta) == hoge)              #条件を満たすかどうかを$
> fasta <- fasta[obj]                        #objがTRUEとなる要素のみ$
> fasta                                      #確認してるだけです
  A DNAStringSet instance of length 4
    width seq                                      names
[1]    21 CGACAGCTCCTCGGCATCCGA                    gene_1
[2]    27 GTCTGCCTCAAGCGCCCCAAGTGGGTT              gene_2
[3]    21 TGTAGGAGAAGGGCGTAATCT                    gene_3
[4]    30 CGTGCTGATTCCACACAGCAGTAAACGCGG           gene_5
>
> #本番
> hoge <- translate(fasta)                   #fastaをアミノ酸配列に翻$
> names(hoge) <- names(fasta)                #現状では翻訳した結果の$
> fasta <- hoge                              #hogeの中身をfastaに格納
> fasta                                      #確認してるだけです
  A AAStringSet instance of length 4
    width seq                                      names
[1]     7 RQLLGIR                                  gene_1
[2]     9 VCLKRPKWV                                gene_2
[3]     7 CRRRA*S                                  gene_3
[4]    10 RADSTQQ*TR                               gene_5
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=50)#fa$
>
```

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
fasta                                      #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,
```

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
fasta                                      #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,..の数を配列こ
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
fasta                                      #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
fasta <- hoge                              #hogeの中身をfastaに格納
fasta                                      #確認してるだけです
```

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
out_f <- "hoge5.fasta"

#必要なパッケージをロー
library(Biostrings)

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

#前処理(ACGTのみからなる
hoge <- rowSums(alphab
obj <- (width(fasta) =
fasta <- fasta[obj]
fasta

#本番
hoge <- translate(fast
names(hoge) <- names(f
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])#A$
> obj <- (width(fasta) == hoge)          #条件を満たすかどうかを$
> fasta <- fasta[obj]                    #objがTRUEとなる要素のみ$
> fasta                                  #確認してるだけです
  A DNAStringSet instance of length 4
    width seq                                          names
[1]    21 CGACAGCTCCTCGGCATCCGA                        gene_1
[2]    27 GTCTGCCTCAAGCGCCCCAAGTGGGTT                  gene_2
[3]    21 TGTAGGAGAAGGGCGTAATCT                        gene_3
[4]    30 CGTGCTGATTCCACACAGCAGTAAACGCGG               gene_5
>
> #本番
> hoge <- translate(fasta)
> names(hoge) <- names(fasta)
> fasta <- hoge
> fasta
  A AAStringSet instance of le
    width seq
[1]     7 RQLLGIR                                      gene_1
[2]     9 VCLKRPKWV                                    gene_2
[3]     7 CRRRA*S                                      gene_3
[4]    10 RADSTQQ*TR                                   gene_5
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=50)#fa$
>
```

「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

UAU (Tyr/Y)チロシン
UAC (Tyr/Y)チロシン
UAA Ochre (*終止*)
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

入力：sample4.fasta

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

出力：hoge5.fasta

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

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、トランスクリプトーム、正規化、発現変動、統計、モデル、バイオインフォマティクス～

(last modified 2014/08/03, 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(講

- 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 | 一般 | 配列取得 |
- イントロ | 一般 | 配列取得 |
- イントロ | 一般 | 配列取得 |
- イントロ | 一般 | 配列取得 |
- イントロ | 一般 | 配列取得 |
- 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 | qrqc (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 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/

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
library(Bio
#入力ファイルの
fasta <- readD
fasta

#本番
hoge <- rowSum
obj <- (width(
fasta <- fasta
fasta

#ファイルに保存
writeXStringSe
```


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。

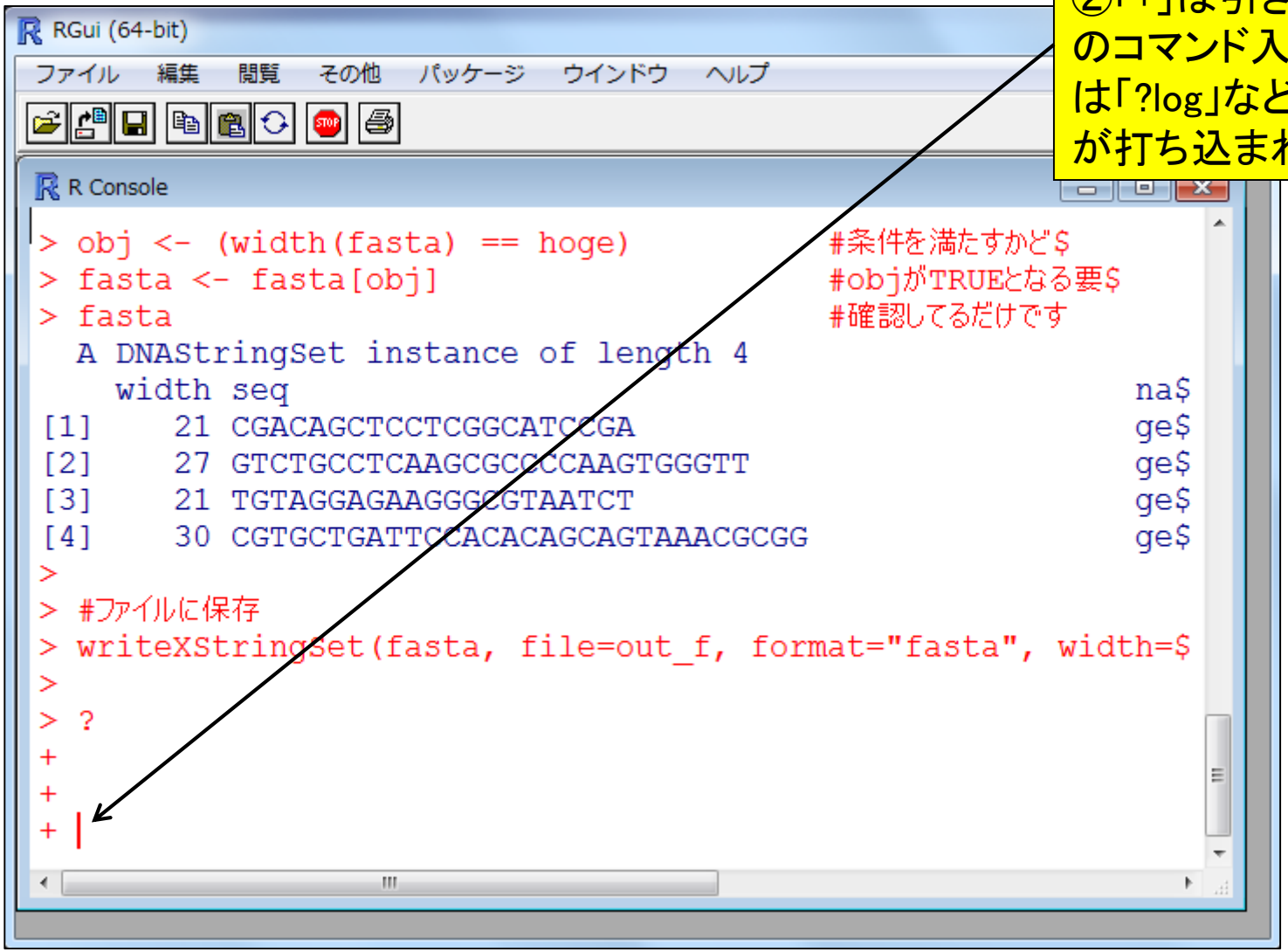
6. (multi-)FASTA形式ファイル(sample4.fasta)の場合:

gene_4が消えていることが分かります。

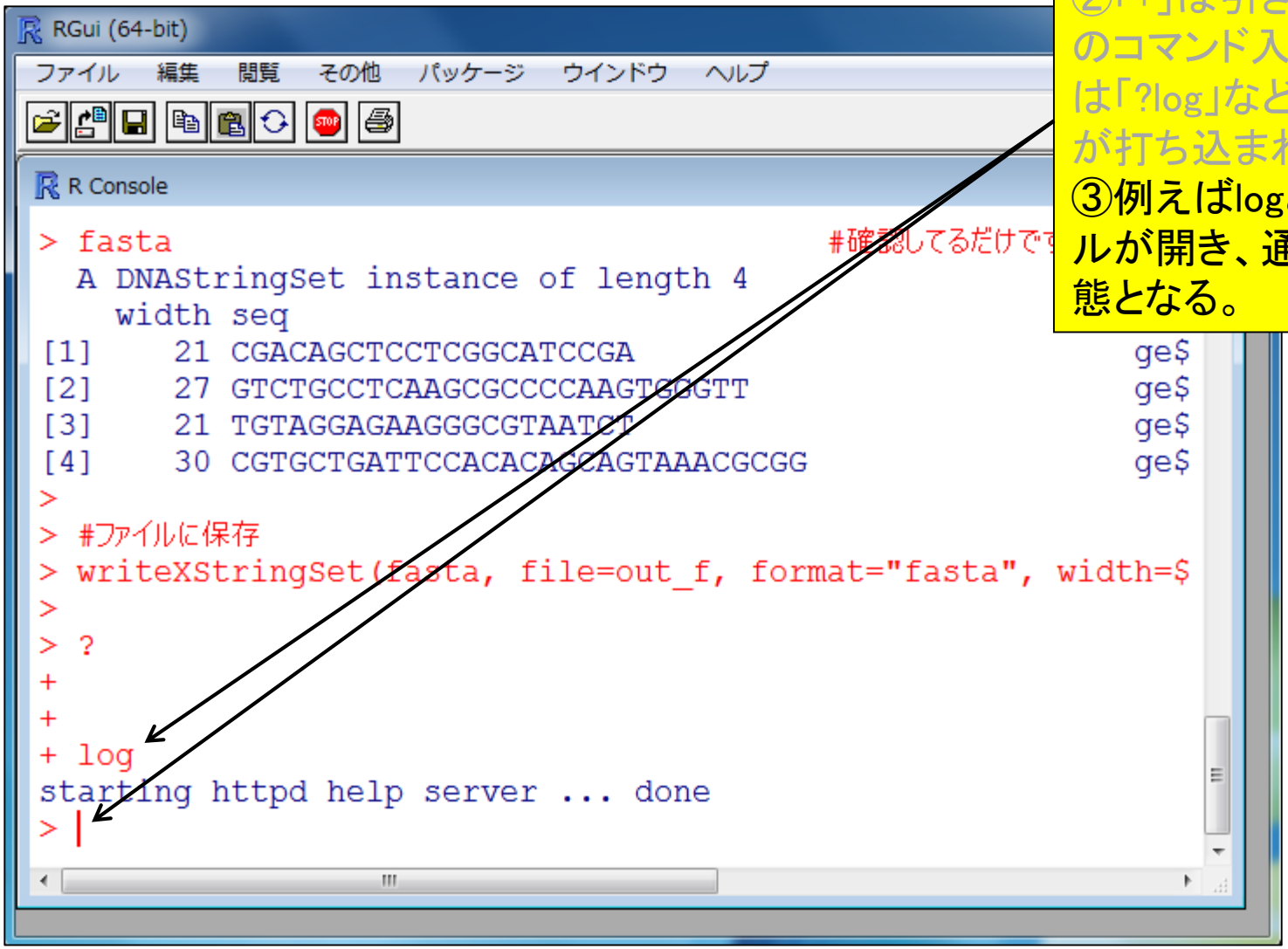
```
in_f <- "sample4.fasta"                  #入力ファイル名を指定してin_fに格納
out_f <- "hoge6.fasta"                   #出力ファイル名を指定してout_fに格納

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

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

#本番
hoge <- rowSums(alphabetFrequency(DNAStringSet(fasta))[,1:4])#A,C,G,T,..の数を配列ごとに力
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
fasta                                    #確認してるだけです

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

Tips

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。

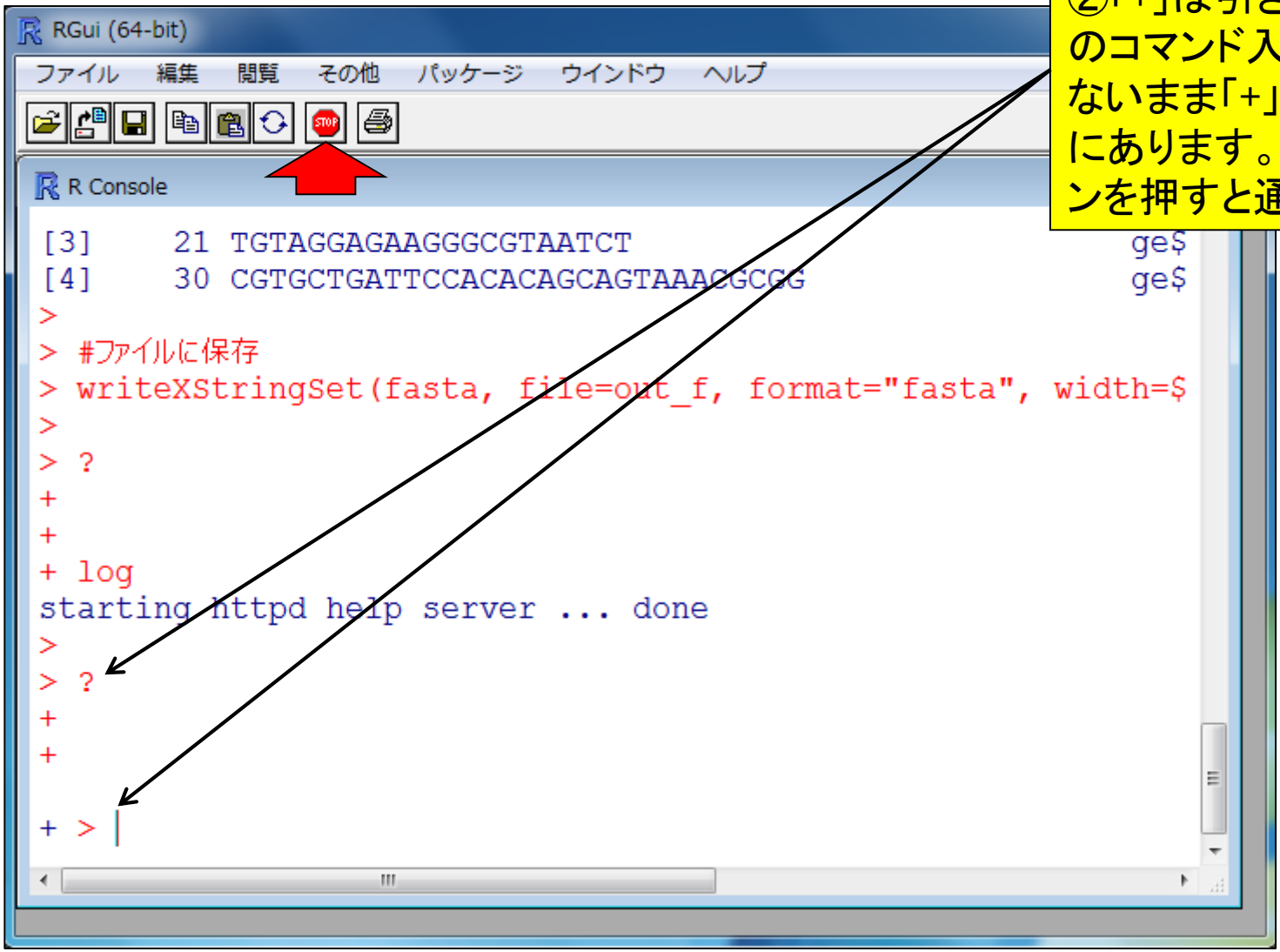


Tips



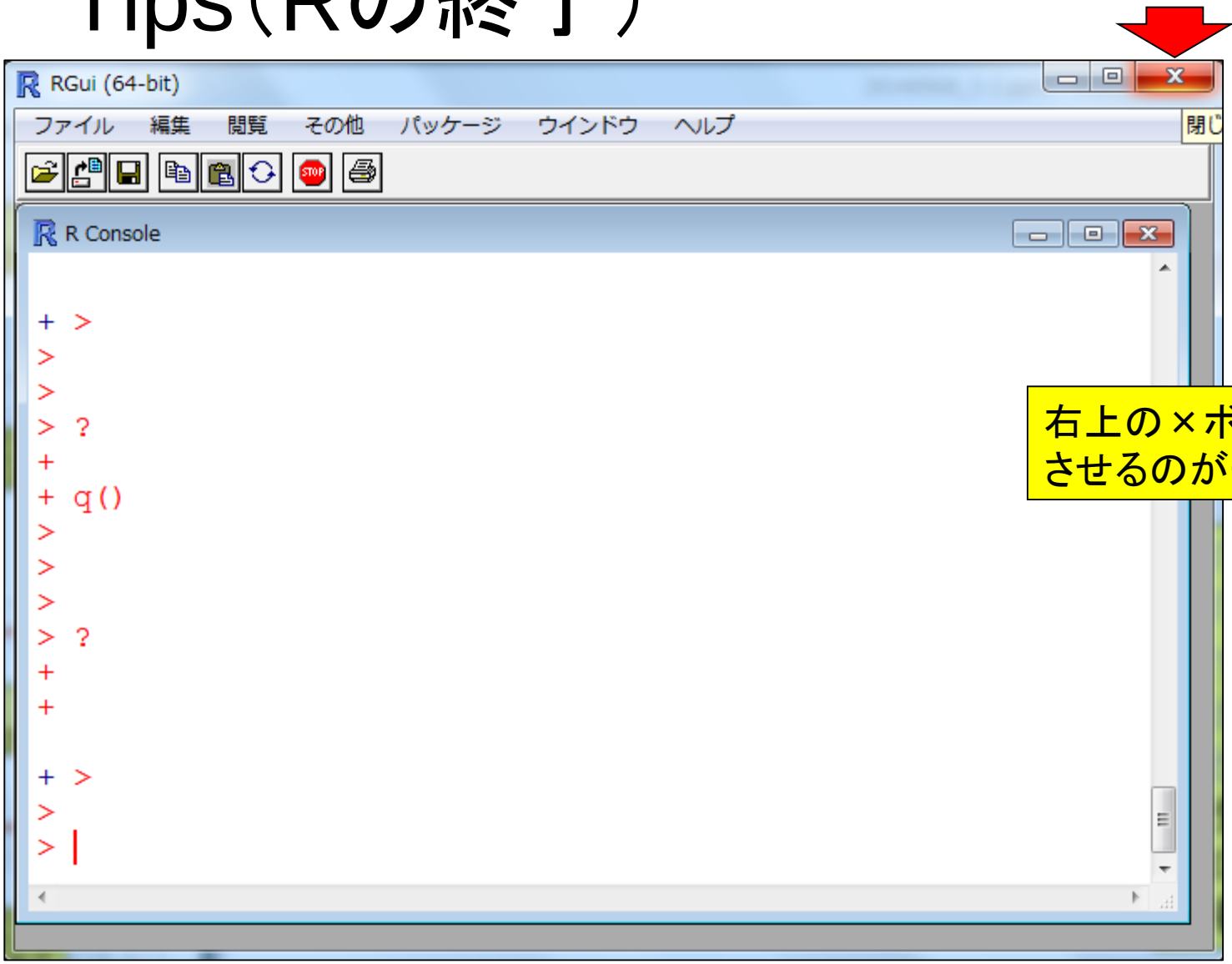
①「?」のみを打ち込んでリターン。
②「+」は引き続いて打ち込まれるはずのコマンド入力待ち状態。例えば通常は「?log」などと?に引き続いて関数名が打ち込まれるはず。
③例えばlogと打ち込むとhtmlマニュアルが開き、通常のコマンド入力待ち状態となる。

Tips



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。

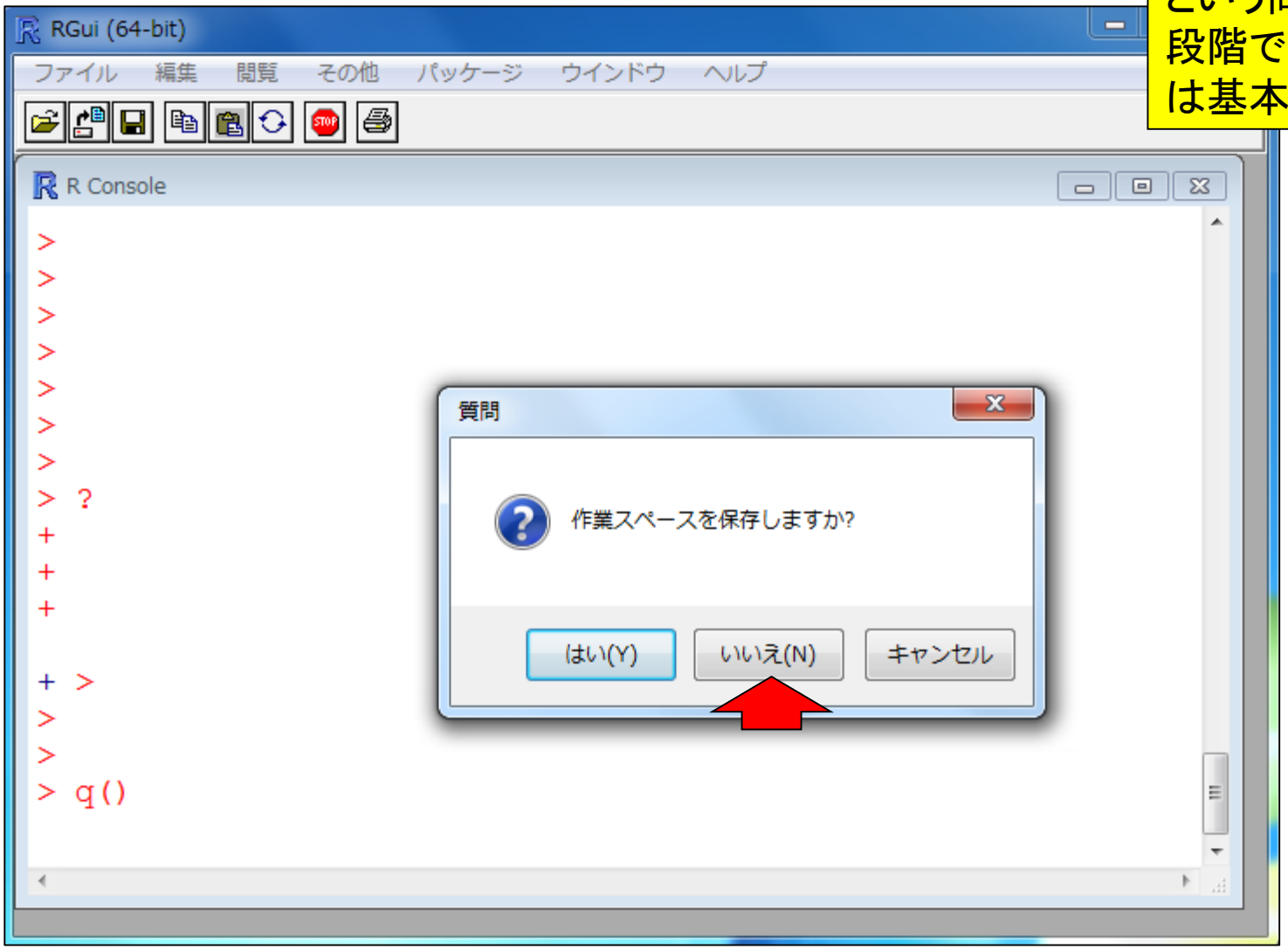
Tips（Rの終了）



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

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、トランスクリプトーム、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
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
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
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
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
```

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

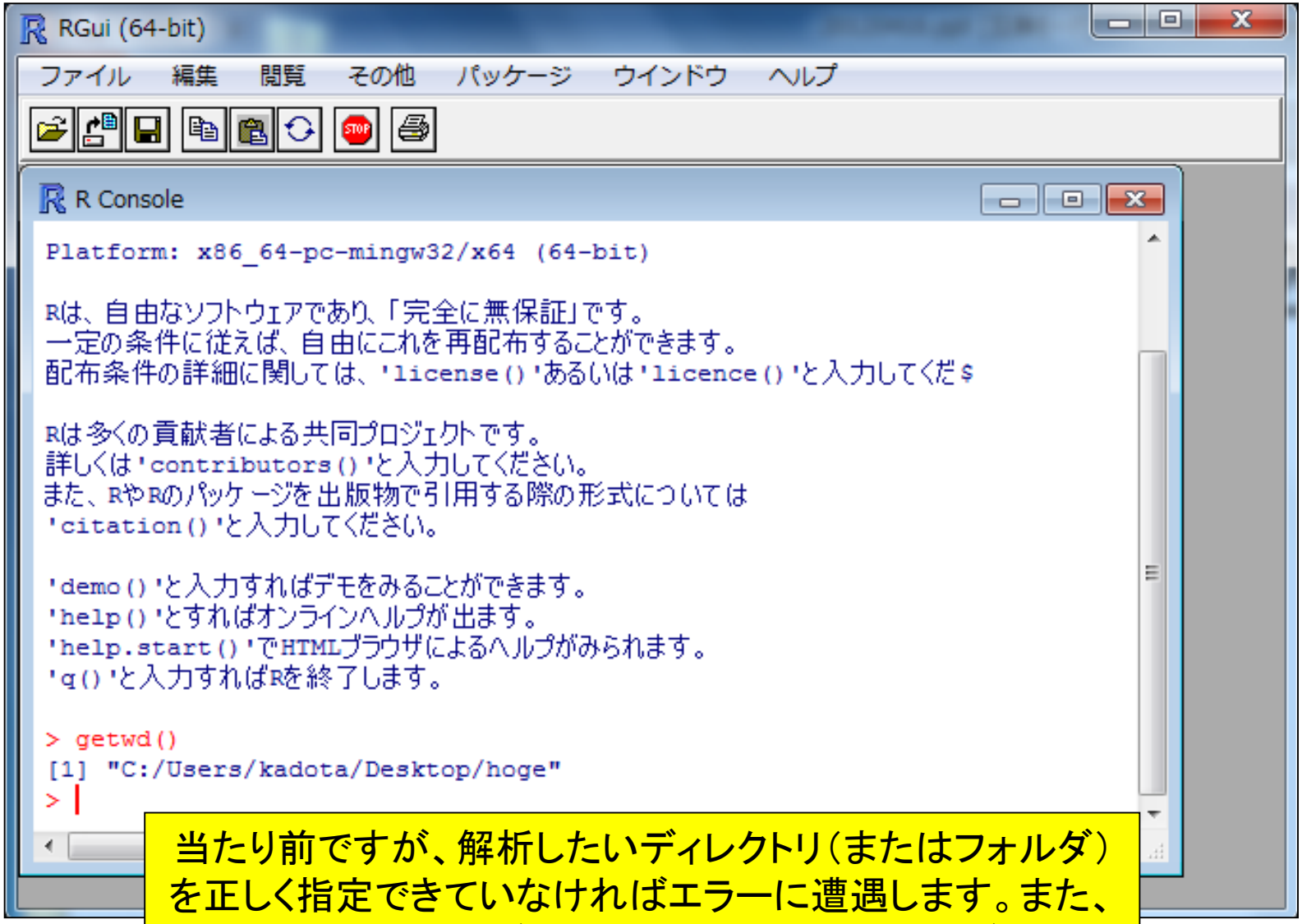
出力: hoge1.txt

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



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

基本はコピペ

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。

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。

1. 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:

```
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, he
keywords <- readLines(in f2)
dim(data)

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

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

切り取り(T)
コピー(C)
貼り付け
すべて選択(A)
印刷(I)...
印刷プレビュー
Bing でマップ
Bing で翻訳
Google で検索
電子メール (W
すべてのアク
Send to One

RGui (64-bit)

ファイル 編集 閲覧 その他 パッケージ ウインドウ ヘルプ

R Console

```
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

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
また、RやRのパッケージを出版物で引用する
'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
'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
'q()'と入力すればRを終了します。

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

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
ウィンドウを常にトップに置く

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

実行結果

RGui (64-bit)

ファイル 編集 閲覧 その他 パッケージ ウインドウ ヘルプ

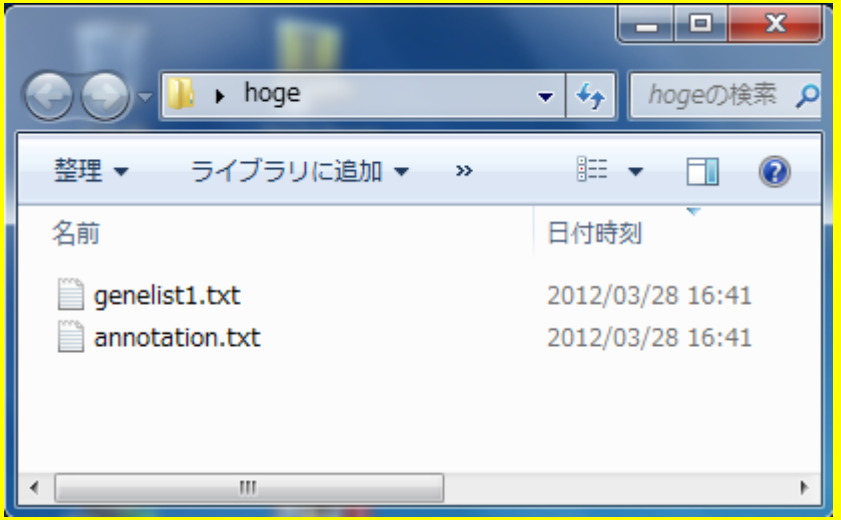
R Console

```
> 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)#$
>
>
```

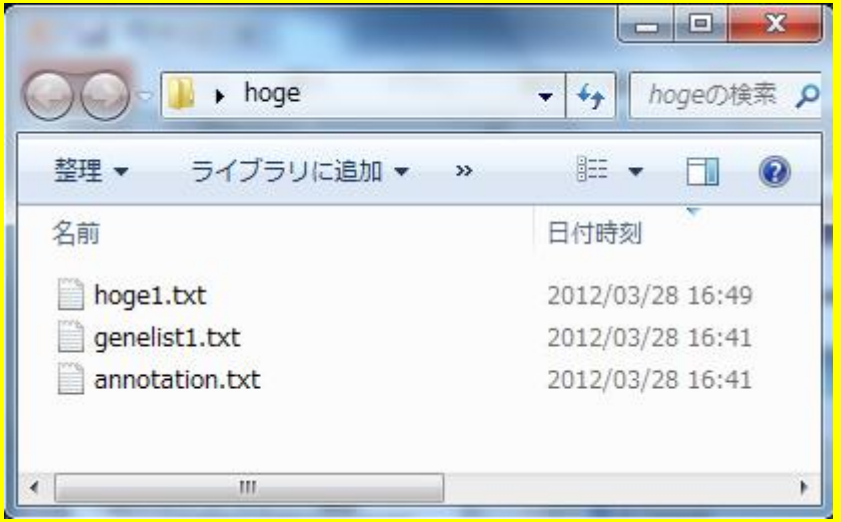
hoge1.txt

	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

実行前のhogeフォルダ



実行後のhogeフォルダ



色についての説明

(Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランスクリプトーム、正規化、発現変動、統計、モデル、バイオインフォマティクス～
(last modified 2014/08/22, 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)
- 門田幸二 著シリーズ Useful R 第7巻トランスクリプトーム解析刊行(共立出版)

- 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)

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

色についての説明

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

```
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
```

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

1. 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）
2. 該当箇所を変更し、R Console画面上でコピペ

list.txt - メモ帳

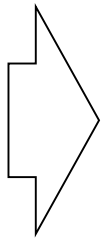
ファイル(F) 編集(E)

```
nuclear
membrane
```

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 <- "hoge1.txt"
param <- 1

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

#本番
obj <- is.element(as.chara
out <- data[obj,]
dim(out)
write.table(out, out_f, se
```



run1.txt - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

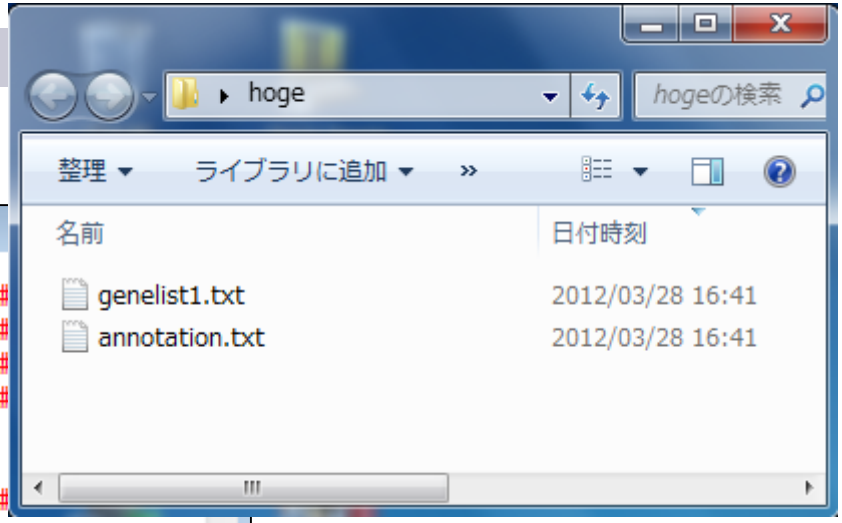
```
in_f1 <- "annotation.txt"
in_f2 <- "list.txt"
out_f <- "hoge1.txt"
param <- 4

#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=
```

一連の作業手順を記述した**スクリプト**を1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

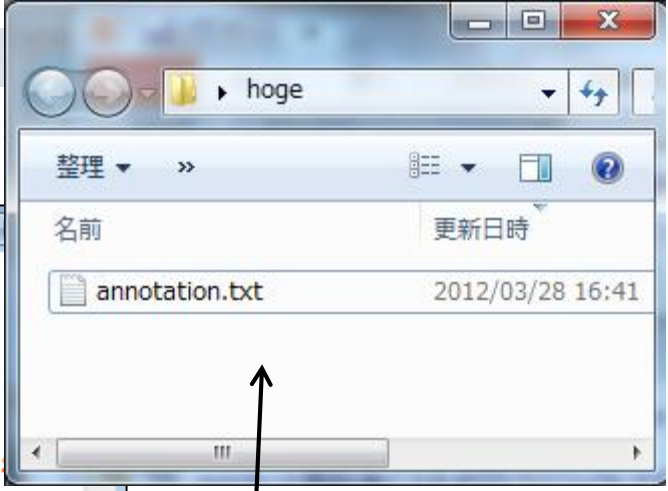
```
> in_f1 <- "annotation.txt"                                          #
> in_f2 <- "genelist1.txt"                                           #
> out_f <- "hoge1.txt"                                               #
> param <- 1                                                         #
>
> #ファイルの読み込み
> 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 director$
> keywords <- readLines(in_f2)                                       #入力ファ$
 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

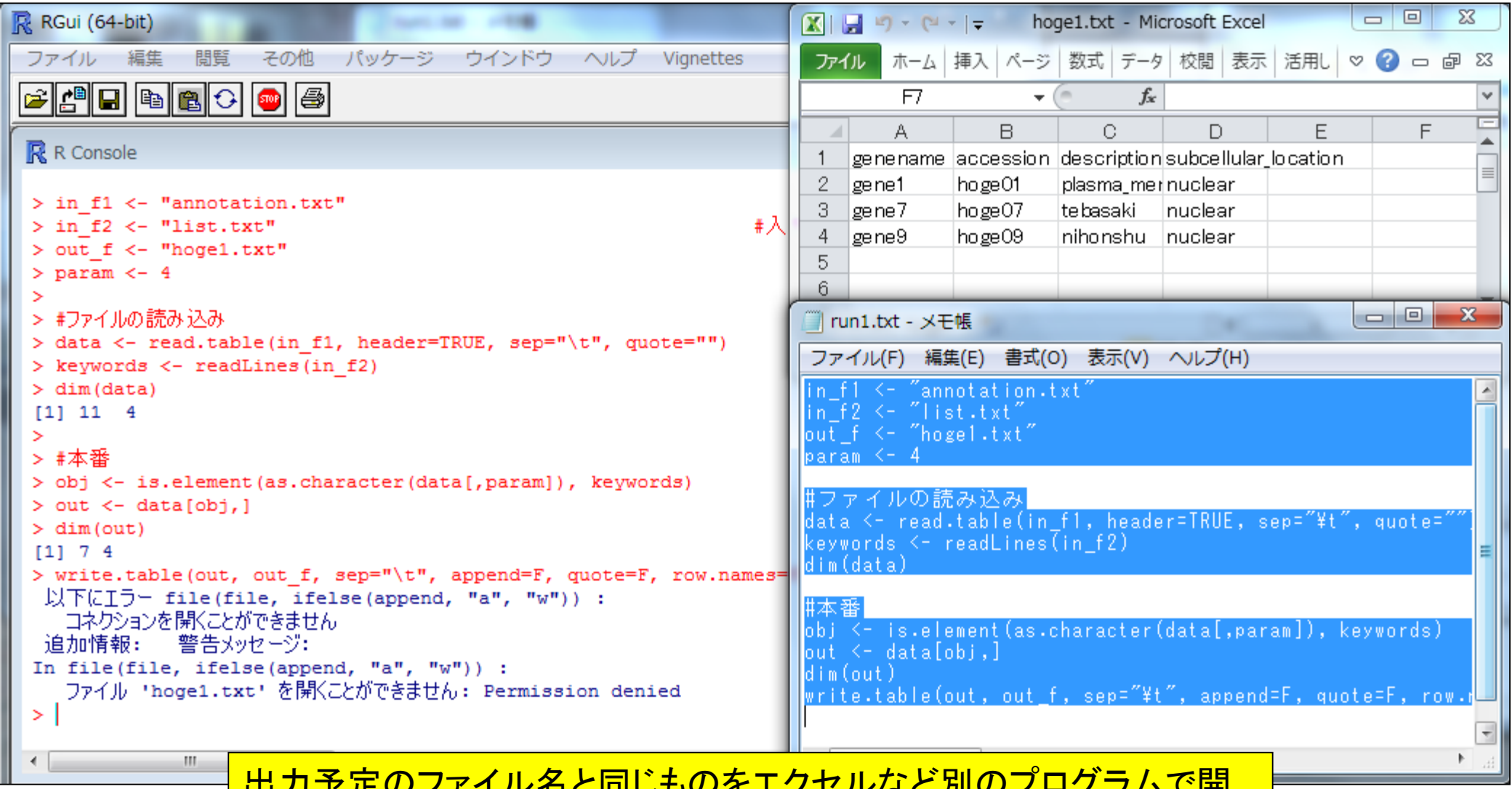
R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> in_f1 <- "annotation.txt"                                    #入力ファイル$
> in_f2 <- "genelist1.txt"                                     #入力ファイル$
> out_f <- "hoge1.txt"                                         #出力ファイル$
> param <- 1                                                   #in_f1で読み
>
> #ファイルの読み込み
> data <- read.table(in_f1, header=TRUE, sep="\t", quote="")   #入力ファイル$
> keywords <- readLines(in_f2)                                 #入力ファイル$
 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)                                                    #オブジェクト$
[1] 11  4
>
> #本番
> obj <- is.element(as.character(data[,param]), keywords)      #in_f1で読み$
 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,]                                            #行列dataから$
 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
>
```



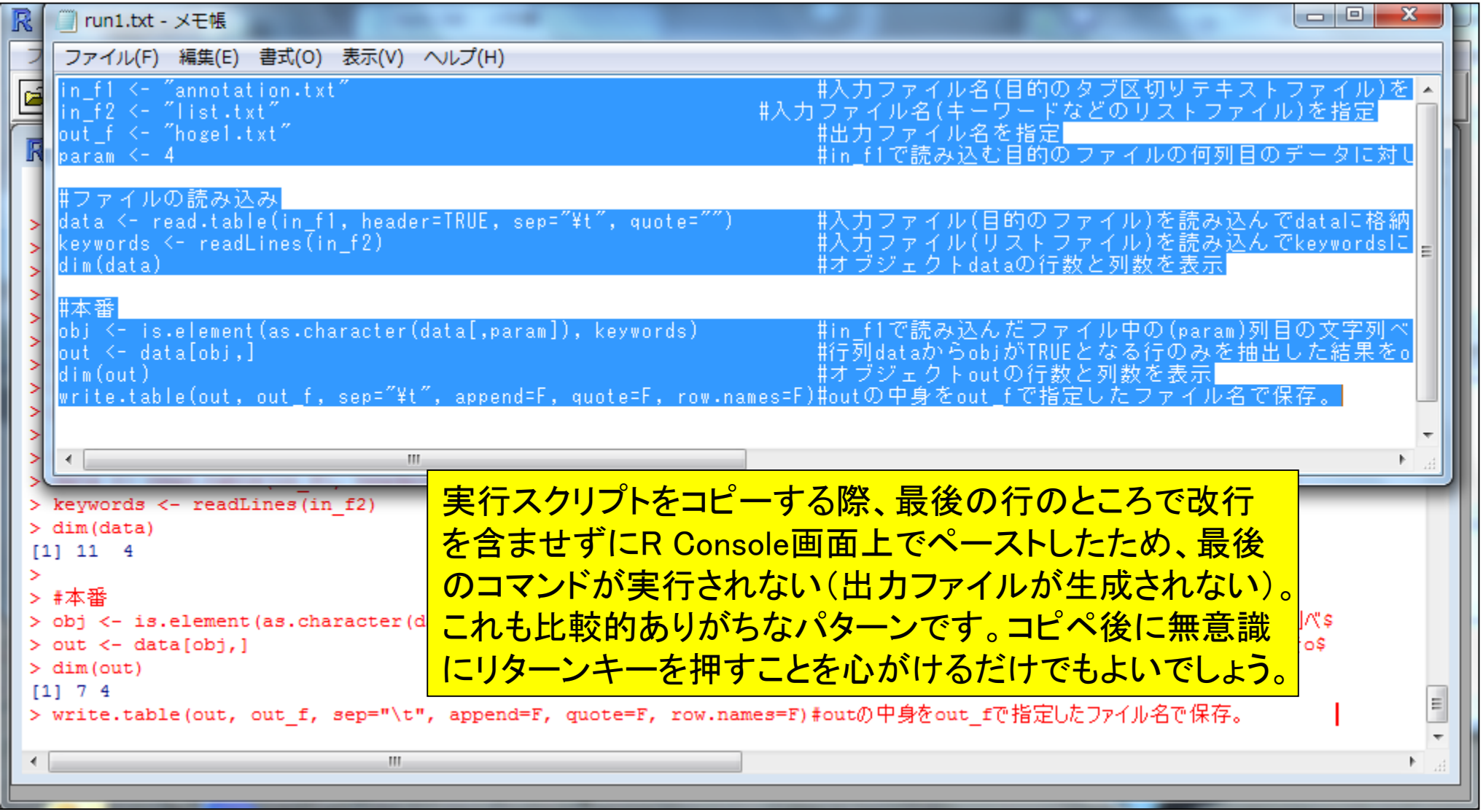
必要な入力ファイルが作業ディレクトリ中に存在しない。この場合、in_f2で指定したgenelist1.txtが存在しないため、それの読み込み段階でエラーが出ている。それゆえ、その情報を用いているコマンド部分でエラーが出ている。

ありがちなミス3



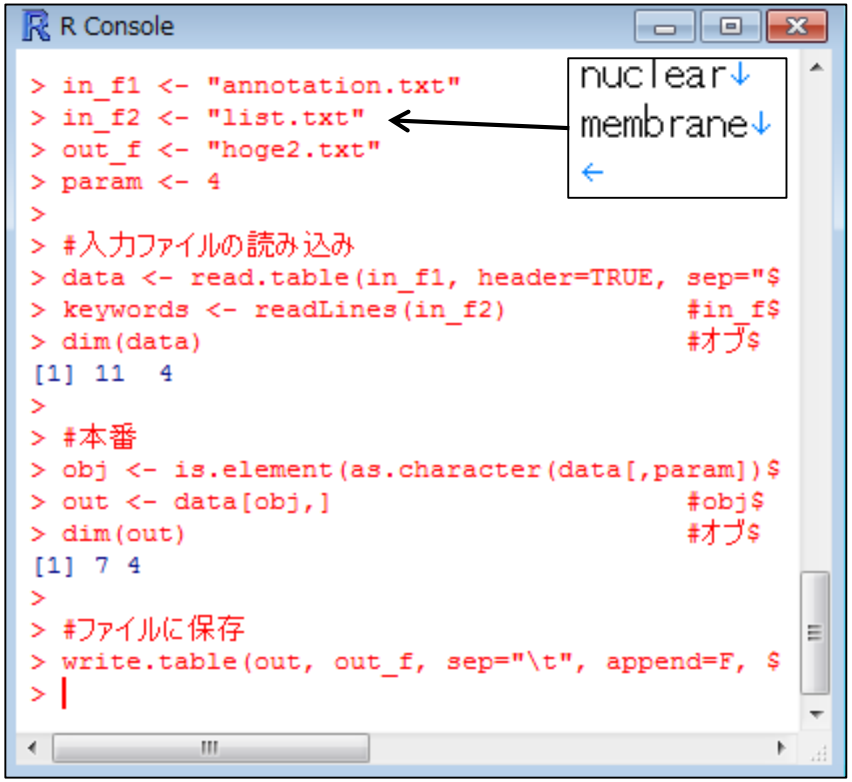
出力予定のファイル名と同じものをエクセルなど別のプログラムで開いているため、最後のwrite.table関数のところでエラーが出る。対処法は、出力ファイル名を変更するか、開いている別のプログラムを閉じる。

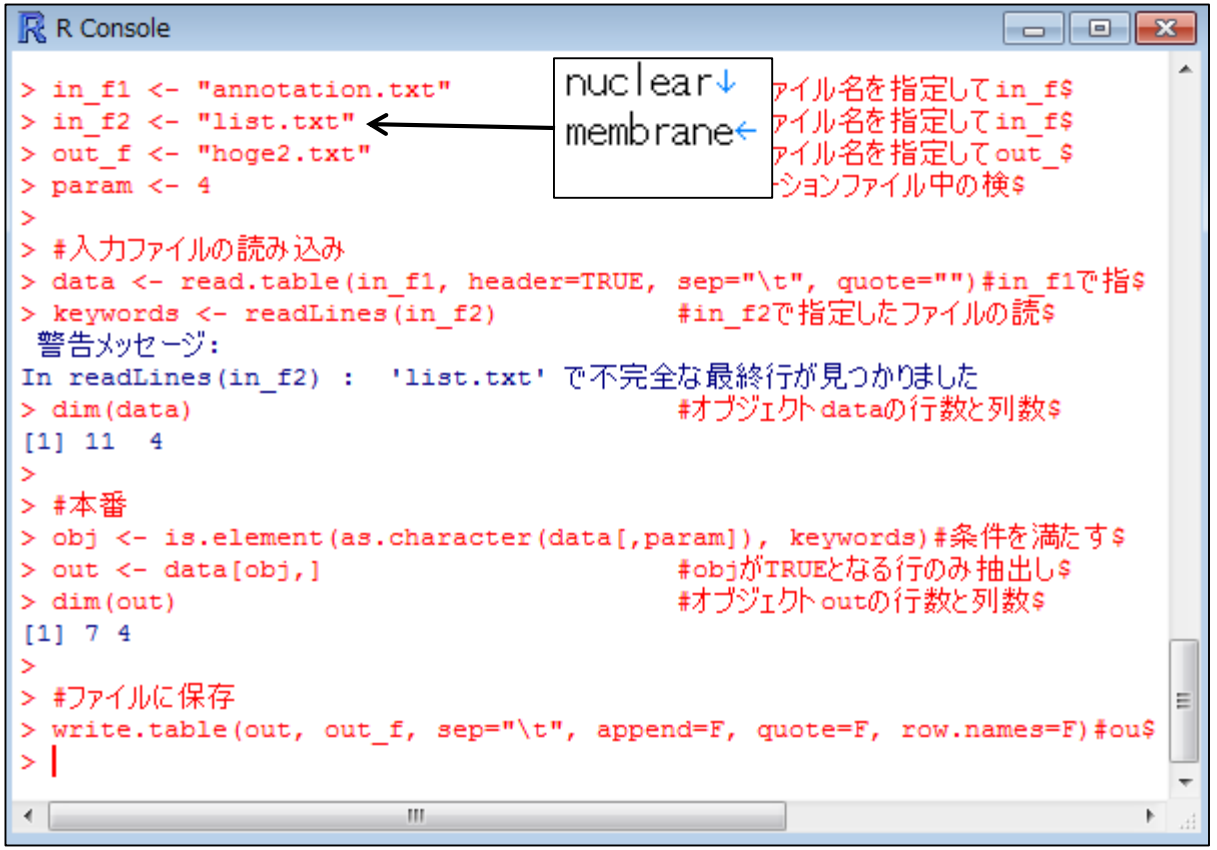
ありがちなミス4



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





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
data <- read.table(in_f1, header=TRUE, sep="\t", quote="")#in_f1で指定した
```

①

②

③

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

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

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

RGui (64-bit)

ファイル 編集 閲覧 その他 パッケージ ウインドウ ヘルプ

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      iyaaaa           endoplasmic
>
```

入力ファイルの中身を正しく
読み込めていることがわかる

```
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,
```

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

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      genesi
5     gene5     hoge05         kam
6     gene6     hoge06      netteb
7     gene7     hoge07     tebasak
8     gene8     hoge08        biir
9     gene9     hoge09    nihonshu          nuclear
10   gene10     hoge10      agene1         membrane
11   gene11     hoge11      iyaaaa       endoplasmic
> dim(data)
[1] 11  4
>
```

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。

行列の要素へのアクセス

R Console

```
10    gene10     hoge10       agene1              me
11    gene11     hoge11       iyaaaa            endop
> 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
>
```

data[行，列]

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"                    #入力ファイル名を指定してin_f1に格
in_f2 <- "genelist1.txt"                     #入力ファイル名を指定してin_f2に格
out_f <- "hoge1.txt"                         #出力ファイル名を指定してout_fに格
param <- 1

#入力ファイルの読み込み
data <- read.table(in_f
keywords <- readLines(i
dim(data)

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

#ファイルに保存
write.table(out, out_f,
```

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
>
```

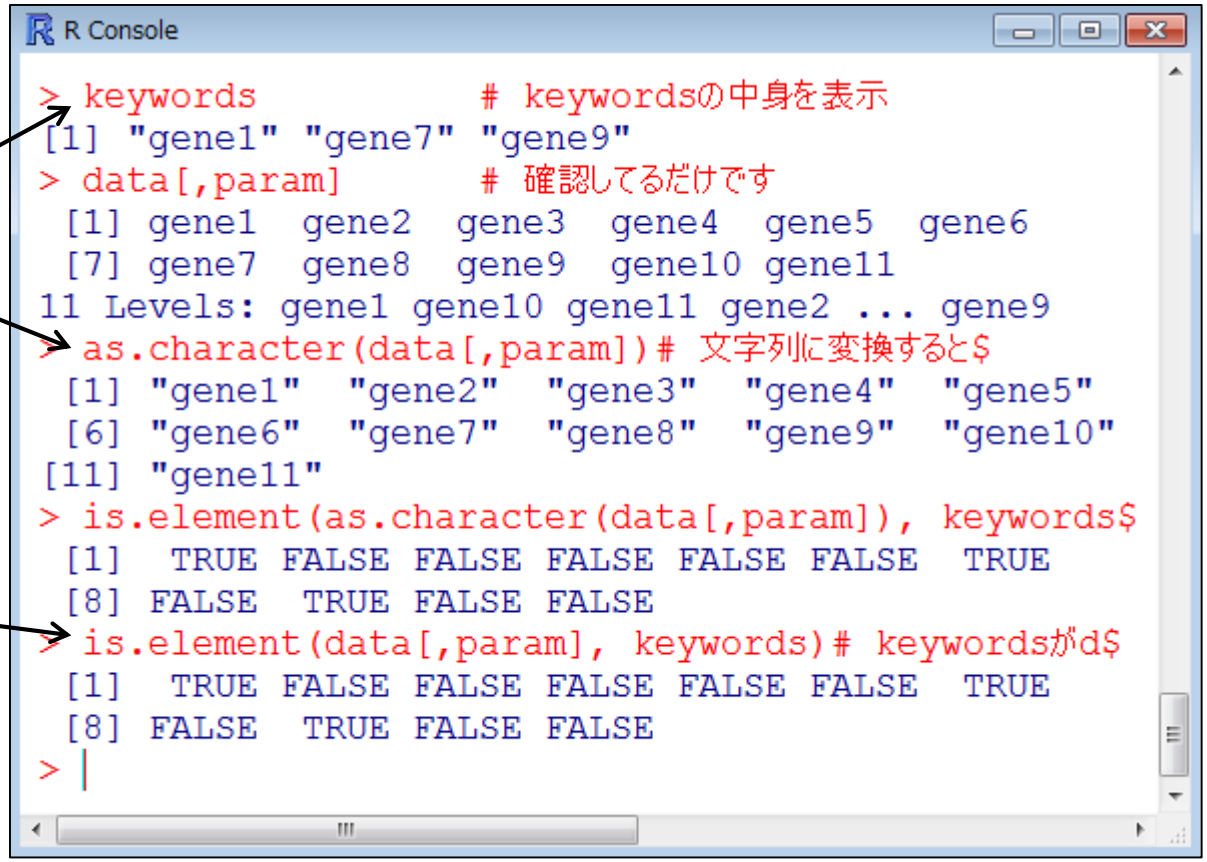
論理値ベクトルobjを用いてTRUEの要素に対応する行を抽出している

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
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。

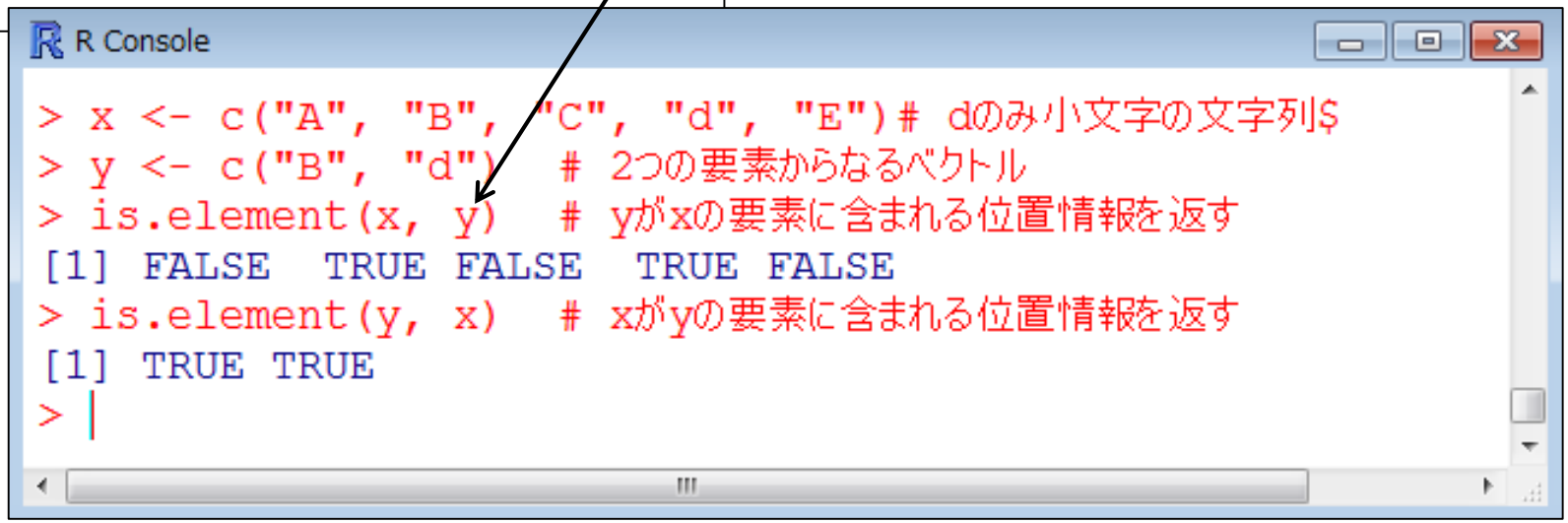


論理値ベクトルを理解

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。

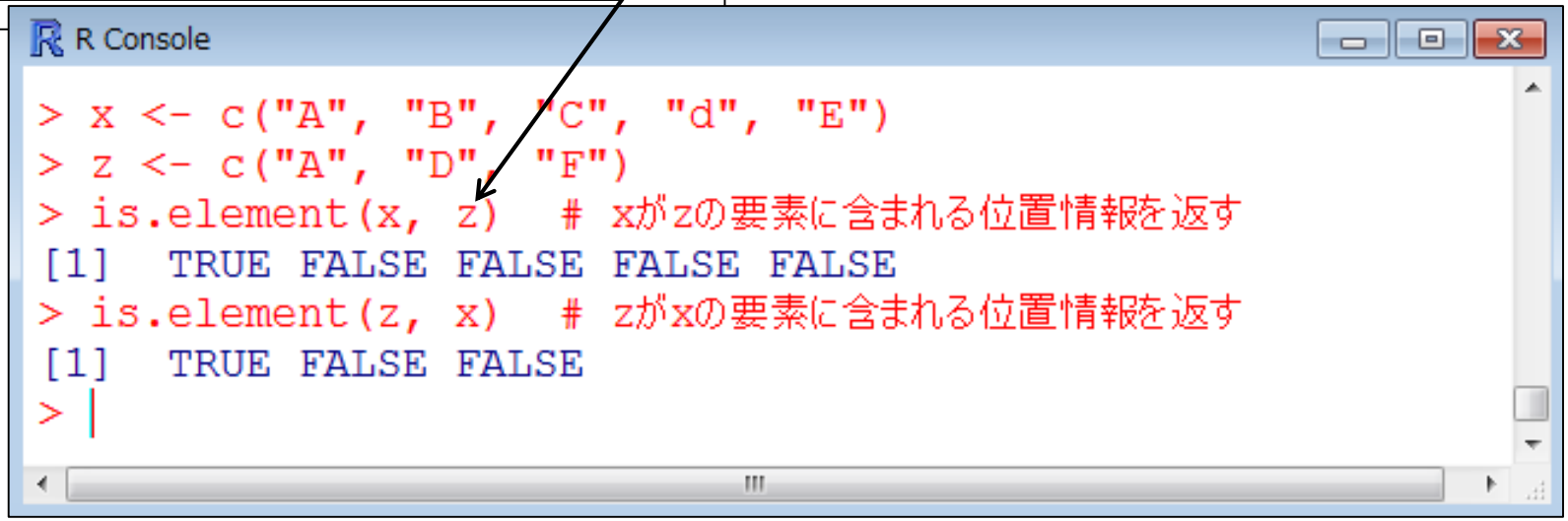


論理値ベクトルを理解

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。**



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 <-
dim(out

#ファイ
write.t
```

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

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
```

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
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_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
```

このコードはヘッダー行がある場合のものです

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

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。

このコードはヘッダー行がない場合のものです

```
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
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
```

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:

```
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_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
```

ヘッダー行がある場合

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

#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
```

ヘッダー行がない場合

Contents

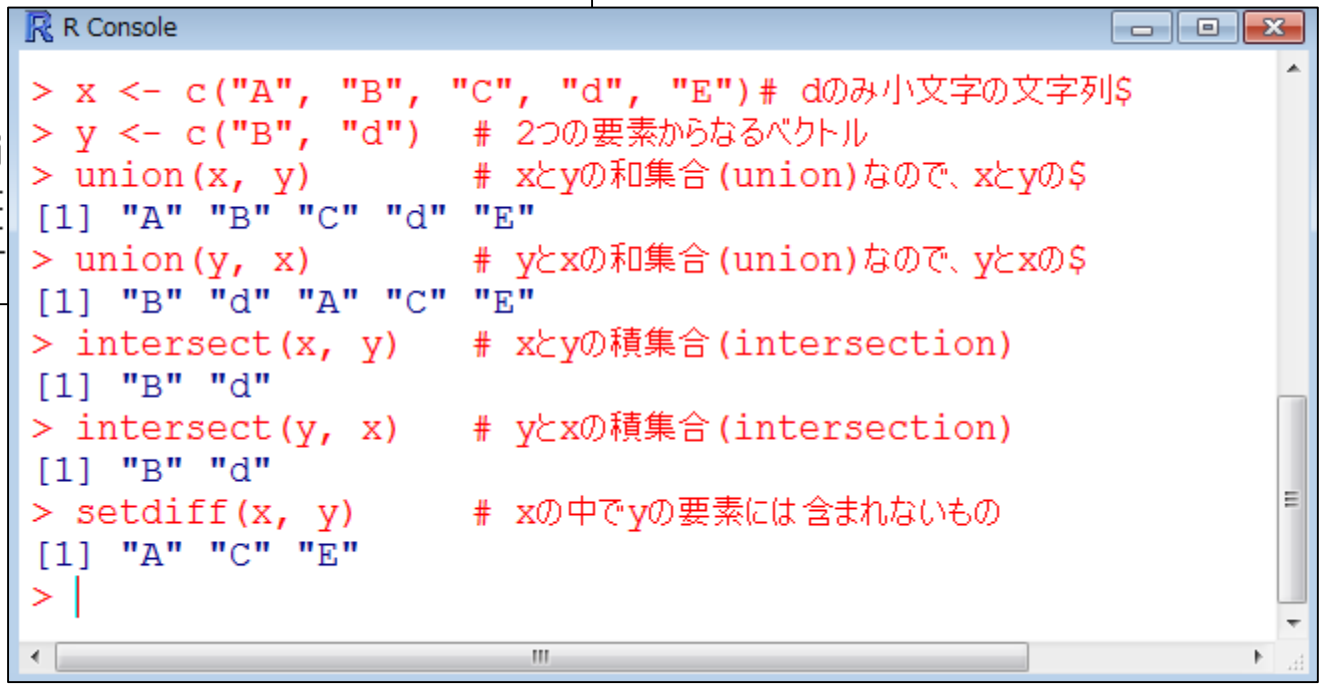
- 3-2. R 基礎2、2014/09/08 13:15-14:45、初級、実習
  - (Rで)**塩基配列解析**の基本的な利用法(翻訳配列の取得を例に)
    - 入力ファイル取得、作業ディレクトリの変更、本番(基本はコピペ)、出力結果の確認
    - 1つの項目に多数の例題(異なる入力ファイル、エラーへの対処例)
  - 行列形式ファイルの解析基礎(アノテーションファイルを例に)
    - 例題をテンプレートとして任意の解析を行う基本手順
    - ありがちなミスとエラーメッセージ
    - 入力ファイルの最後の改行の有無
    - プログラム内部の説明(行列演算の基礎)
  - 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。

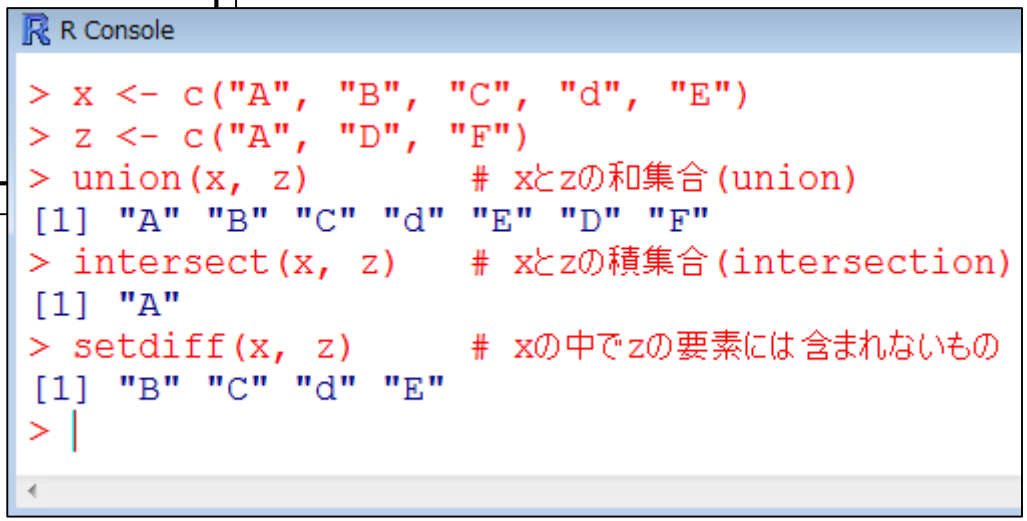


Tips（集合演算）

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。



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 Console

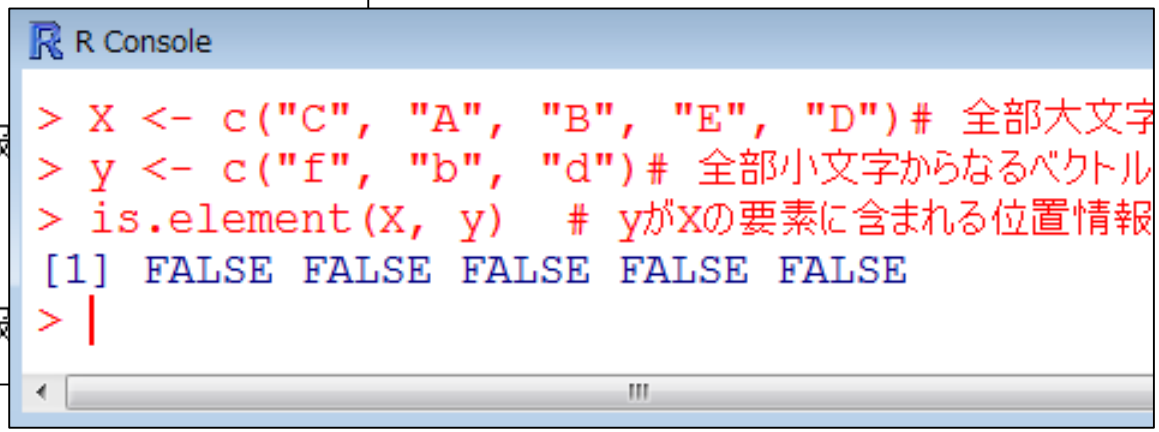
```
> hoge <- c(x, z)    # ベクトルxとzを結合。cは結
> hoge               # hogeの中身を表示
[1] "A" "B" "C" "d" "E" "A" "D" "F"
> sorted <- sort(hoge) # アルファベット順にソート
> sorted             # sortedの中身を表示
[1] "A" "A" "B" "C" "d" "D" "E" "F"
> table(sorted)      # 要素ごとの出現回数
sorted
A B C d D E F
2 1 1 1 1 1 1
> table(hoge)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

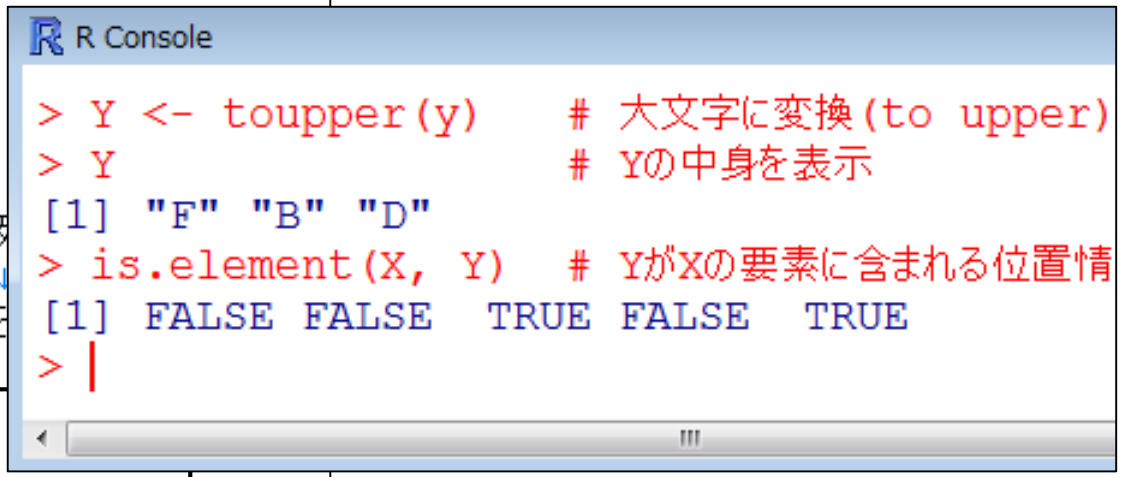


Tips(その他)

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

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。



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

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。