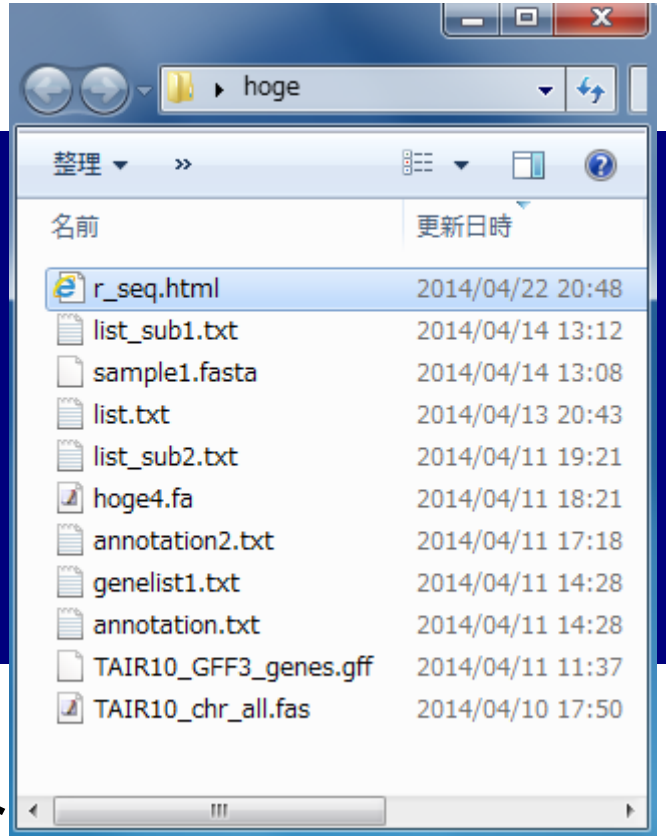
講義室後ろにあるUSBメモリ中のhogeフォルダをデスクトップにコピーしておいてください。



ゲノム情報解析基礎

～ Rで塩基配列解析 ～

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）

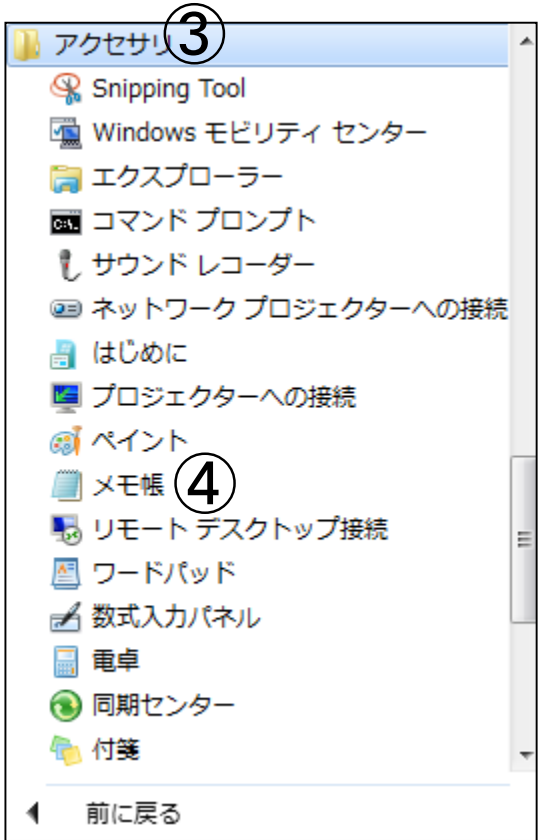
http://www.iu.a.u-tokyo.ac.jp/~kadota/

kadota@iu.a.u-tokyo.ac.jp

前回のhogeフォルダがデスクトップに残っているかもしれないのでご注意ください。

各種ソフトの場所

①



スタート ― すべてのプログラム ― …

ゲノム情報解析

- 例：シロイヌナズナ（*Arabidopsis thaliana*）
  - ゲノム配列決定（Chr1-5, ChrC, ChrM）
    - 1番染色体：Theologis et al., *Nature*, **408**: 816-820, 2000
    - 2番染色体：Lin et al., *Nature*, **402**: 761-768, 1999
    - 3番染色体：Salanoubat et al., *Nature*, **408**: 820-822, 2000
    - …
  - トランスクリプトーム配列（cDNA配列）決定
    - アノテーション：Seki et al., *Science*, **296**: 141-145, 2002
    - …
  - まとめサイト
    - The Arabidopsis Information Resource (TAIR)
    - Lamesch et al., *Nucleic Acids Res*., **40**(Database issue): D1202-1210, 2011
    - http://www.arabidopsis.org/

	Length	GC contents
chr1	28.76MB	35.80%
chr2	19.60MB	35.80%
chr3	23.17MB	35.40%
chr4	17.40MB	36.02%
chr5	25.95MB	34.50%

Rは論文中に書かれている
ことの一部を再現?!できます

(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 2014/04/10, since 2010)

What's ne

- 2014年9
  農で開催
- 門田幸二
  解析を仮
  向けに、
  置いた構
- 参考資料
- 私の所属
  フォ関連
  東大以外
  ノム情報
  興味ある
- 機能解析
  解析のみ

- 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 2014/02/07)
- 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 2014/04/10) NEW
- イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome(last modified 2014/04/01) 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) NEW
- イントロ | N
- イントロ | N
- イントロ | N
- イントロ | N
- イントロ | N
- イントロ | N
- イントロ | N
- イントロ | N
- イントロ | N
- イントロ | N

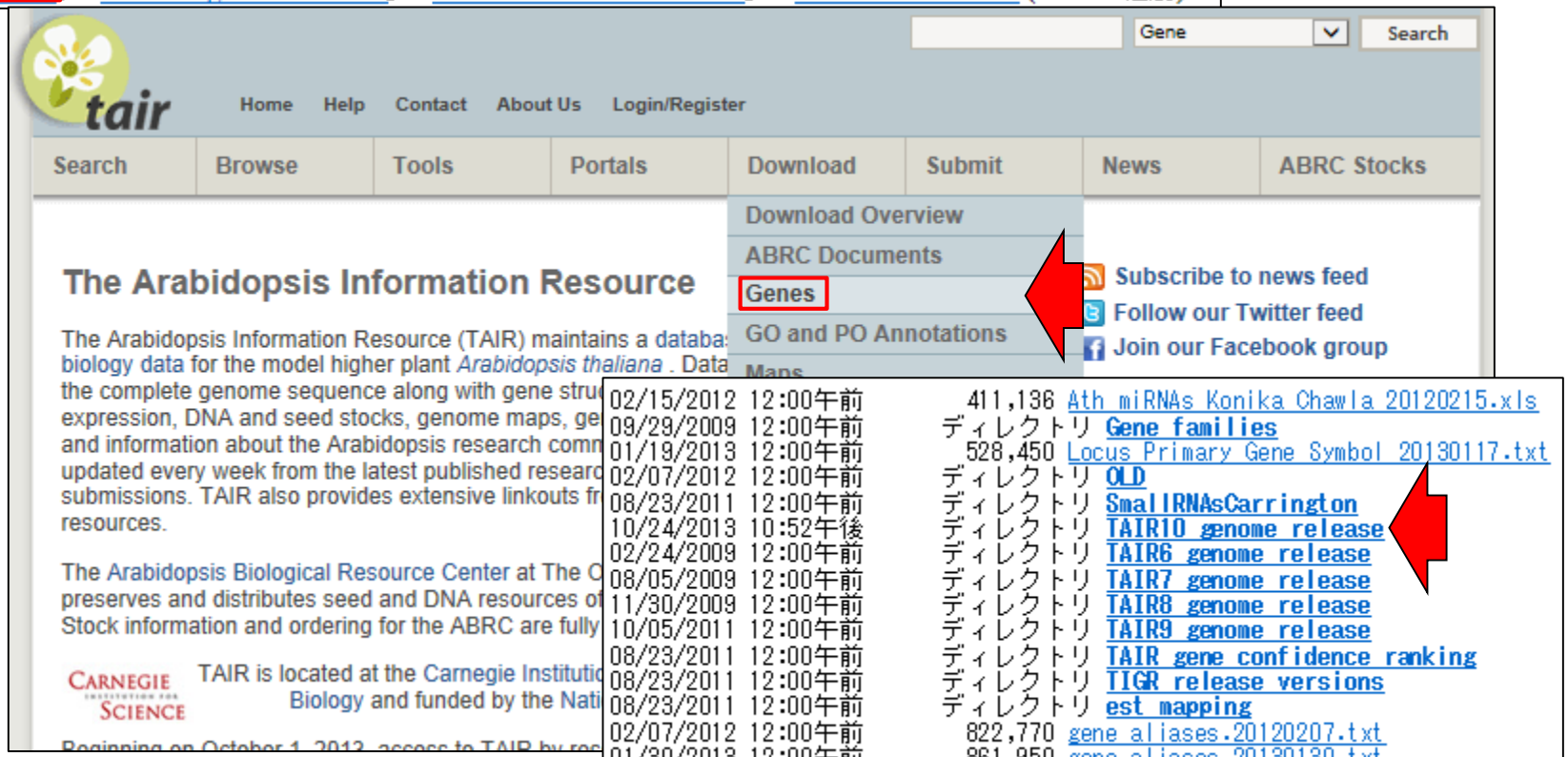
イントロ | 一般 | 配列取得 | ゲノム配列 | 公共DBから NEW

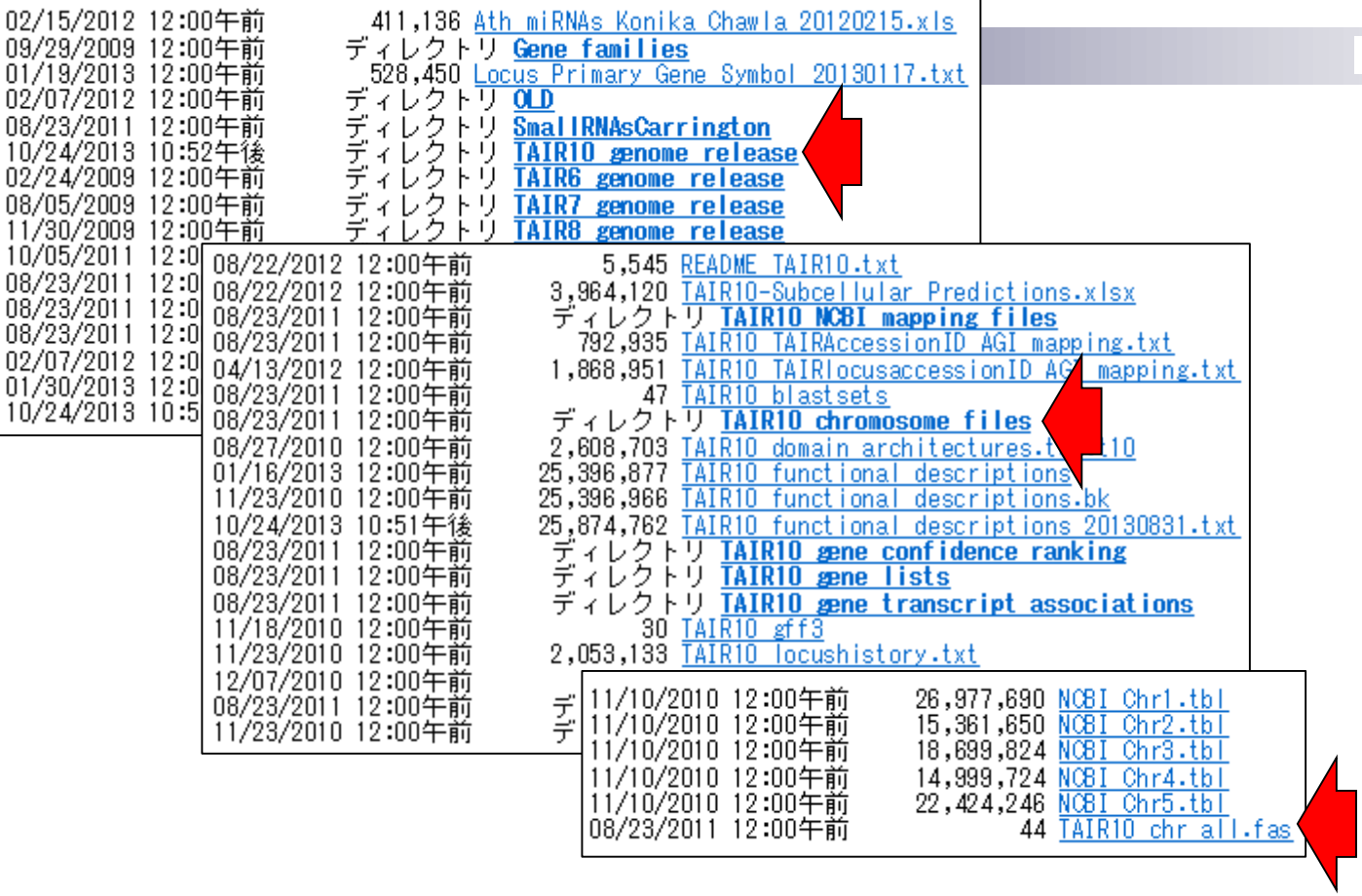
- UCSCの Sequence and Annotation Downloads (Karolchik et al., Nucleic Acids Res., 2014)
  - ヒト; Human(H.sapiens)
  - ラット; Rat(R.norvegicus)
  - ネコ; Cat(F.catus)
  - ウサギ; Rabbit(O.cuniculus)
  - ニワトリ; Chicken(G.gallus)
  - イヌ; Dog(C.familiaris)
  - ウマ; Horse(E.caballus)
  - ...
- Helix Systems Scientific Databases (アップデートの日付順になっている。RefSeqやESTなど様々なデータベースを一度にみられる)
- イネ:RAP-DB (Sakai et al., Plant Cell Physiol., 2013)
  - 「ダウンロード」-「Genome assembliesのところのDownload」。IRGSP-1.0_genome.fasta.gz (116MB程度)の圧縮ファイル。
- シロイヌナズナ:The Arabidopsis Information Resource (TAIR) (Lamesch et al., Nucleic Acids Res., 2012)
  - 「ダウンロード」-「Genes」-「TAIR10_genome_release」-「TAIR10_chromosome_files」の TAIR10_chr_all.fas (120MB程度)

TAIR10のゲノム配列ファイル(TAIR10_chr_all.fas)はここからダウンロードしました

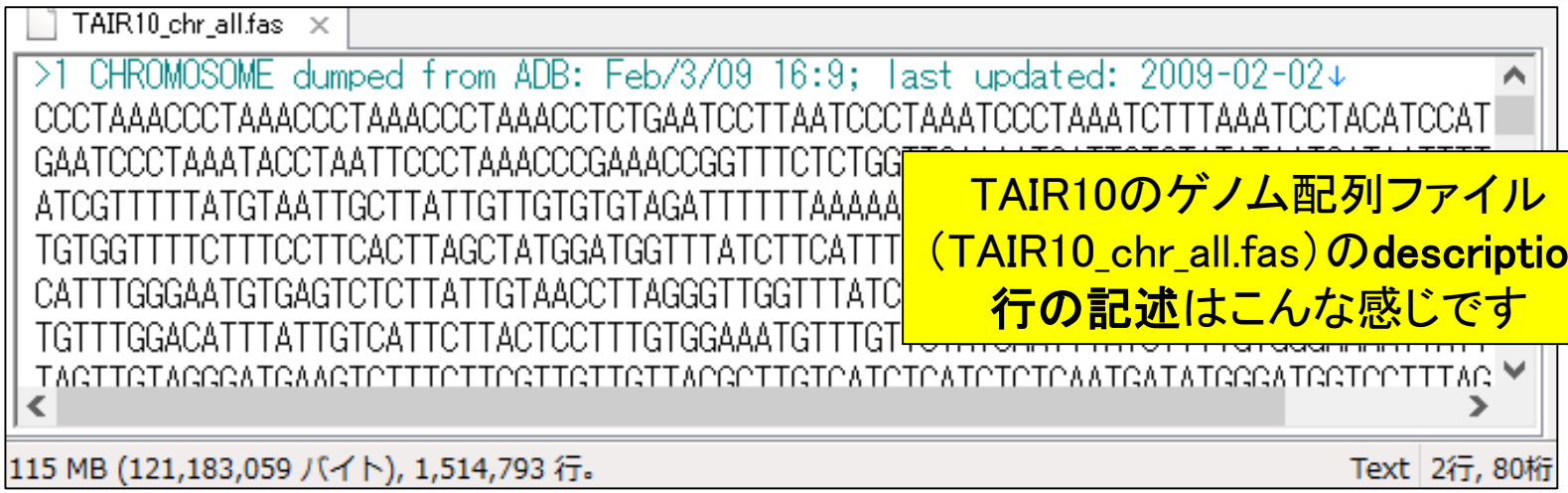
- UCSCの Sequence and Annotation Downloads (Karolchik et al., Nucleic Acids Res., 2014)
  - ヒト ; Human(H.sapiens)
  - ラット ; Rat(R.norvegicus)
  - ネコ; Cat(F.catus)
  - ウサギ ; Rabbit(O.cuniculus)
  - ニワトリ; Chicken(G.gallus)
  - イヌ; Dog(C.familiaris)
  - ウマ; Horse(E.caballus)
  - ...
- Helix Systems Scientific Databases (アップデートの日付順になっている。RefSeqやESTなど様々なデータベースを一度にみられる)
- イネ : RAP-DB (Sakai et al., Plant Cell Physiol., 2013)
  - 「ダウンロード」-「Genome a... nesのところの Download」。IRGSP-1.0_genome.fasta.gz (116MB程度)の圧縮ファイル。
- シロイヌナズナ : The Arabidopsis Information Resource (TAIR) (Lamesch et al., Nucleic Acids Res., 2012)
  - 「ダウンロード」-「Genes」-「TAIR10_genome_release」-「TAIR10_chromosome_files」の TAIR10_chr_all.fas (120MB程度)

TAIRウェブインターフェースから取得する際のイメージ





	Length	GC contents
chr1	28.76MB	35.80%
chr2	19.60MB	35.80%
chr3	23.17MB	35.40%
chr4	17.40MB	36.02%
chr5	25.95MB	34.50%



TAIR10のゲノム配列ファイル（TAIR10_chr_all.fas）のdescription行の記述はこんな感じです

(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 2014/04/10, since 2010)

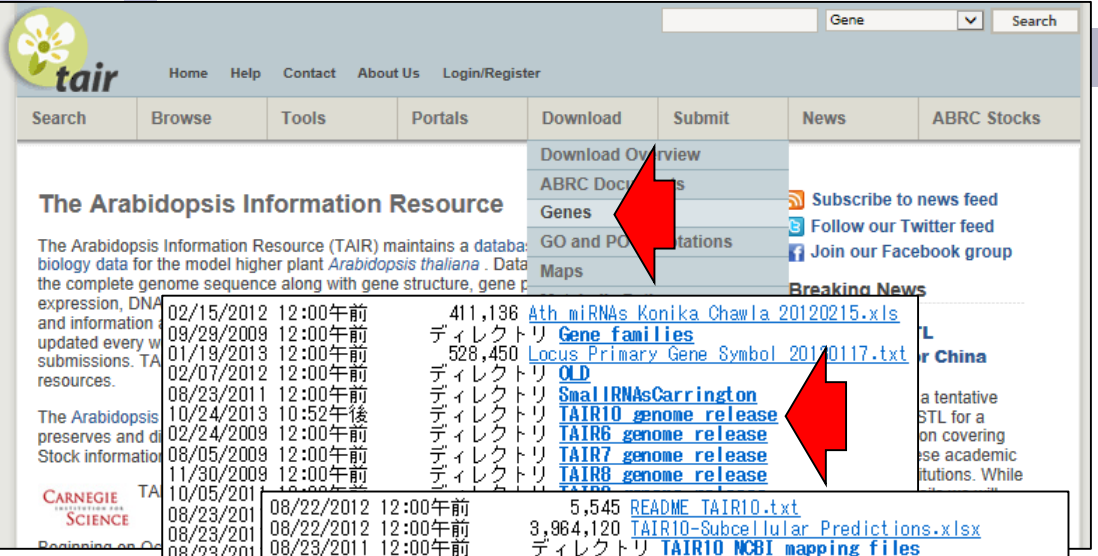
- イントロ | NGS | 様々なプラットフォーム (last modified 2013/06/12)
- イントロ | NGS | qPCRやmicroarrayなどとの比較 (last modified 2010/12/16)
- イントロ | NGS | Viewer (last modified 2014/01/29)
- イントロ | NGS | 配列取得 | FASTQ or SRALite | 公共DBから (last modified 2014/03/27) NEW
- イントロ | NGS | 配列取得 | FASTQ or SRALite | SRAdb(Zhu 2013) (last modified 2014/04/01) NEW
- イントロ | NGS | アノテーション情報取得 | について (last modified 2014/03/26) NEW
- イントロ | NGS | アノテーション情報取得 | GFF/GTF形式ファイル (last modified 2014/04/10) NEW
- イントロ | NGS | アノテーション情報取得 | refFlat形式ファイル (last modified 2013/09/25)
- イントロ | NGS | アノテーション情報取得 | biomaRt(Durinck 2009) (last modified 2013/09/26)

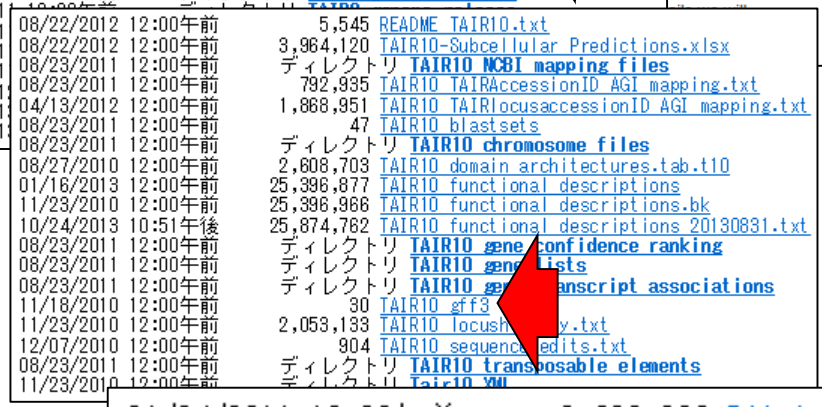
イントロ | NGS | アノテーション情報取得 | GFF/GTF形式ファイル NEW

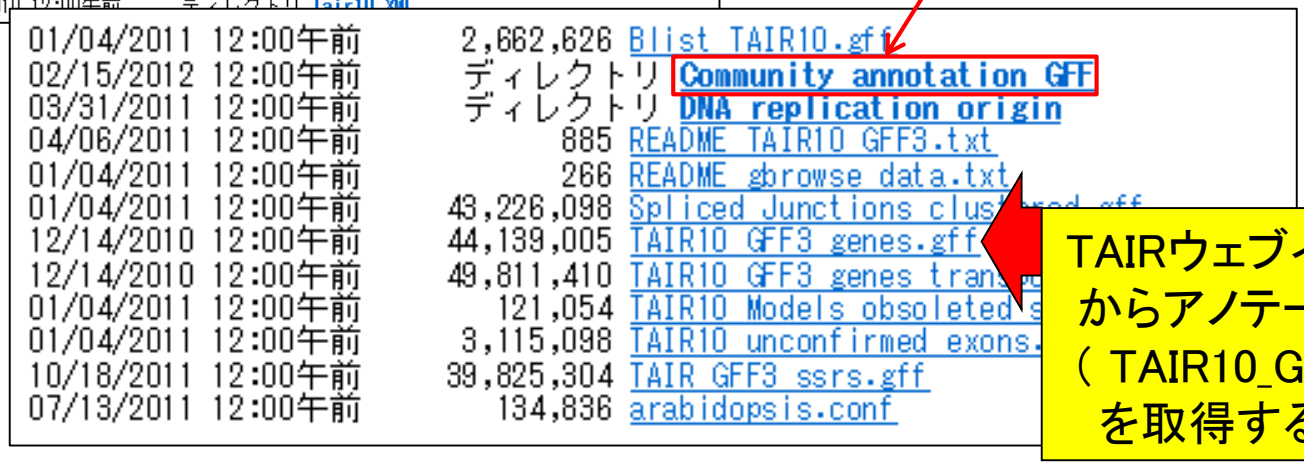
多くの生物種についてEnsembl (Flicek et al., Nucleic Acids Res., 2013) のFTPサイトからGTF形式(GFF ver. 2)の遺伝子アノテーションファイルを得ることができます。refFlat形式同様、どの領域にどの遺伝子があるのかという座標(Coordinates)情報を含みます。ゲノム配列のバージョンと同じであることを確認した上で用いましょう。

- Ensembl (Flicek et al., Nucleic Acids Res., 2013)
  圧縮(gzip)ファイル形式です。基本はFTPサイトです。代表的なものを以下にリストアップしています。
  - ヒト; Human(H.sapiens)
  - ラット; Rat(R.norvegicus)
  - ネコ; Cat(F.catus)
  - ウサギ; Rabbit(O.cuniculus)
  - ニワトリ; Chicken(G.gallus)
  - イヌ; Dog(C.familiaris)
  - ウマ; Horse(E.caballus)
  - ゼブラフィッシュ; Zebrafish (D.rerio)
  - ...
- イネ: RAP-DB (Sakai et al., Plant Cell Physiol., 2013)
  - 「ダウンロード」-「Gene set」-「Gene structure and function inf
    IRGSP-1.0_representative_2014-03-05.tar.gz (12.4MB程度)の圧縮ファイルが得られ
- シロイヌナズナ: The Arabidopsis Information Resource (TAIR) (Lamesch et al., Nucleic Ac s., 2012)
  - 「ダウンロード」-「Genes」-「TAIR10_genome_release」-「TAIR10_gff3」の TAIR10_GFF3_genes.gff (42MB程度)

TAIR10のアノテーションファイル(TAIR10_GFF3_genes.gff)はここからダウンロードしました







一般的には、Community annotationのほうがより情報量も多くていいらしい

TAIRウェブインターフェースからアノテーションファイル（TAIR10_GFF3_genes.gff）を取得する際のイメージ

GFF/GTF形式ファイルの例

GFF3形式ファイルの例（シロイヌナズナ; TAIR10_GFF3_genes.gff）

	A	B	C	D	E	F	G	H	I
1	Chr1	TAIR10	chromosome	1	30427671	.	.	.	ID=Chr1;Name=Chr1
2	Chr1	TAIR10	gene	3631	5899	.	+	.	ID=AT1G01010;Note=protein_coding_gene;Name=AT1G01010
3	Chr1	TAIR10	mRNA	3631	5899	.	+	.	ID=AT1G01010.1;Parent=AT1G01010;Name=AT1G01010.1;Index=1
4	Chr1	TAIR10	protein	3760	5630	.	+	.	ID=AT1G01010.1-Protein;Name=AT1G01010.1;Derives_from=AT1G01010.1
5	Chr1	TAIR10	exon	3631	3913	.	+	.	Parent=AT1G01010.1
6	Chr1	TAIR10	five_prime_UTR	3631	3759	.	+	.	Parent=AT1G01010.1
7	Chr1	TAIR10	CDS	3760	3913	.	+	0	Parent=AT1G01010.1,AT1G01010.1-Protein;
8	Chr1	TAIR10	exon	3996	4276	.	+	.	Parent=AT1G01010.1
9	Chr1	TAIR10	CDS	3996	4276	.	+	2	Parent=AT1G01010.1,AT1G01010.1-Protein;
10	Chr1	TAIR10	exon	4486	4605	.	+	.	Parent=AT1G01010.1
11	Chr1	TAIR10	CDS	4486	4605	.	+	0	Parent=AT1G01010.1,AT1G01010.1-Protein;
12	Chr1	TAIR10	exon	4706	5095	.	+	.	Parent=AT1G01010.1

遺伝子ごとに、どの染色体のどの座標上に存在するのかなどの情報を含むタブ区切りテキストファイル

GTF形式ファイルの例（ゼブラフィッシュ; Danio_rerio.Zv9.75.gtf）

	A	B	C	D	E	F	G	H	I
1	#!genome-build Zv9								
2	#!genome-version Zv9								
3	#!genome-date 2010-04								
4	#!genome-build-accession NCBI:GCA_000002035.2								
5	#!genebuild-last-updated 2014-02								
6	7	protein_coding	gene	100958	101715	.	+	.	gene_id "ENSDARG00000076051"; gene_name "CABZ01062994.1"; gene
7	7	protein_coding	transcript	100958	101715	.	+	.	gene_id "ENSDARG00000076051"; transcript_id "ENSDART00000113409
8	7	protein_coding	exon	100958	100975	.	+	.	gene_id "ENSDARG00000076051"; transcript_id "ENSDART00000113409
9	7	protein_coding	CDS	100958	100975	.	+	0	gene_id "ENSDARG00000076051"; transcript_id "ENSDART00000113409
10	7	protein_coding	exon	101077	101715	.	+	.	gene_id "ENSDARG00000076051"; transcript_id "ENSDART00000113409
11	7	protein_coding	CDS	101077	101715	.	+	0	gene_id "ENSDARG00000076051"; transcript_id "ENSDART00000113409
12	7	protein_coding	gene	116160	117573	.	+	.	gene_id "ENSDARG00000088691"; gene_name "BX511027.1"; gene_sour
13	7	protein_coding	transcript	116160	117573	.	+	.	gene_id "ENSDARG00000088691"; transcript_id "ENSDART00000129330

タブ区切りテキストファイルからの情報抽出

入力：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

入力：リストファイル（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

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

(Rで)塩基配列解析(主にNGSやRNA-seq解析)

(last modified 2014/02/06, since 2010)

What's new?

- 項目名の整理を行っています。3CやBS-seq周辺についても少し言及してあります。(2014/02/06) NEW
- 一連の解析パイプライン(RNA-seqデータ取得 -> マッピング -> カウントデータやRPKMデータ取得 -> サンプル間クラスタリングや発現変動解析およびM-A plot描画まで)をアップデートしました。項目名の一番下のほうです。(2013/10/19)
- 発現変動解析用Rパッケージ TCC (ver. 1.2.0; Sun et al., BMC Bioinformatics, 2013)がBioconductorよりリリースされました。最新版を利用したい方は、R (ver. 3.0.2)をインストールしたのち、Bioconductor (ver. 2.13)をインストールしてください。(2013/10/17)
- どのブラウザからでもエラーなく見られる(W3C validation)ように((Rで)マイクロアレイデータ解析も含めて)リニューアルしました。(2013/07/30)
- 2013年7月29日まで公開していた以前の「(Rで)塩基配列解析」のウェブページや関連ファイルはRdeennki.zipからダウンロード可能です(110MB程度)。(2013/07/30)

- イントロ | 一般 | 任意のキーワードを含む行を抽出(基礎)

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のannotation.txtが手元にあり、この中からgenelist1.txtのような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す。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
```

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

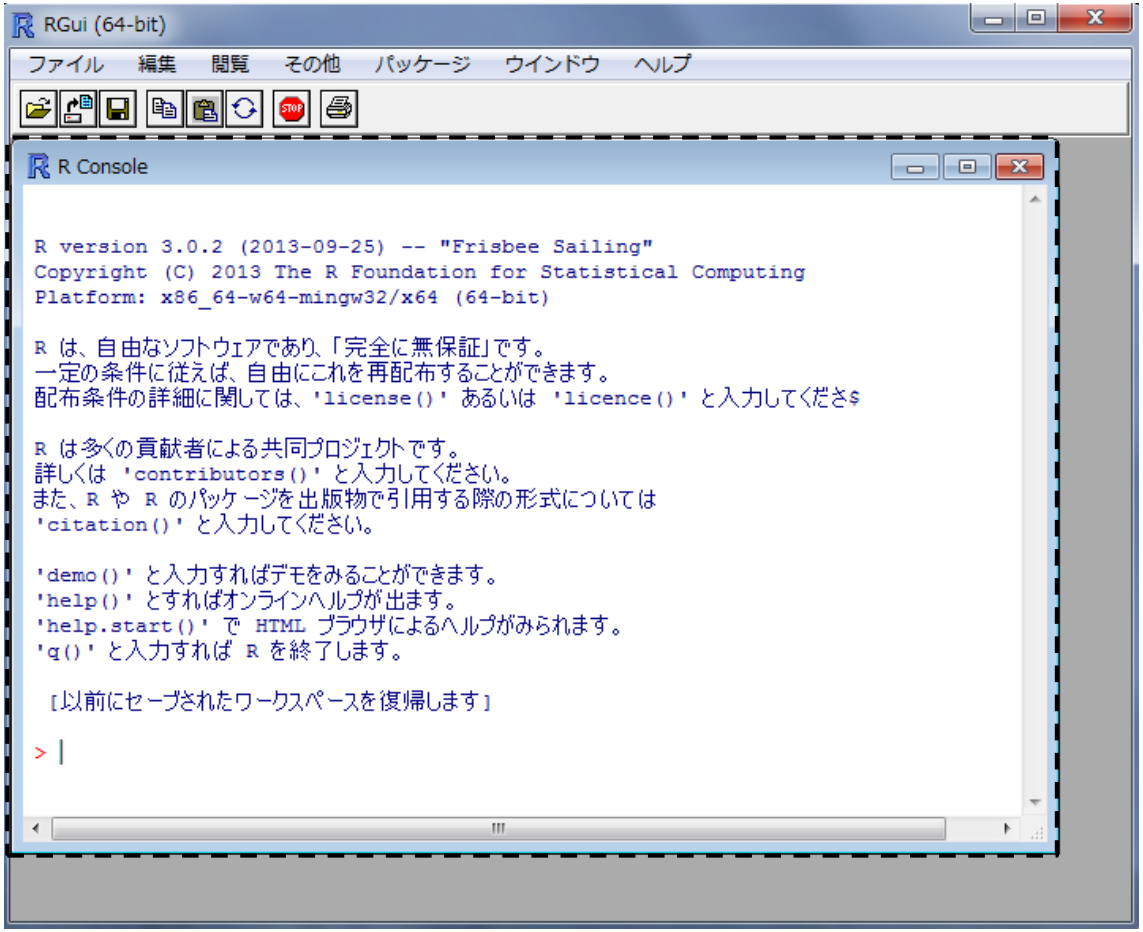
出力: hoge1.txt

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

Rの起動

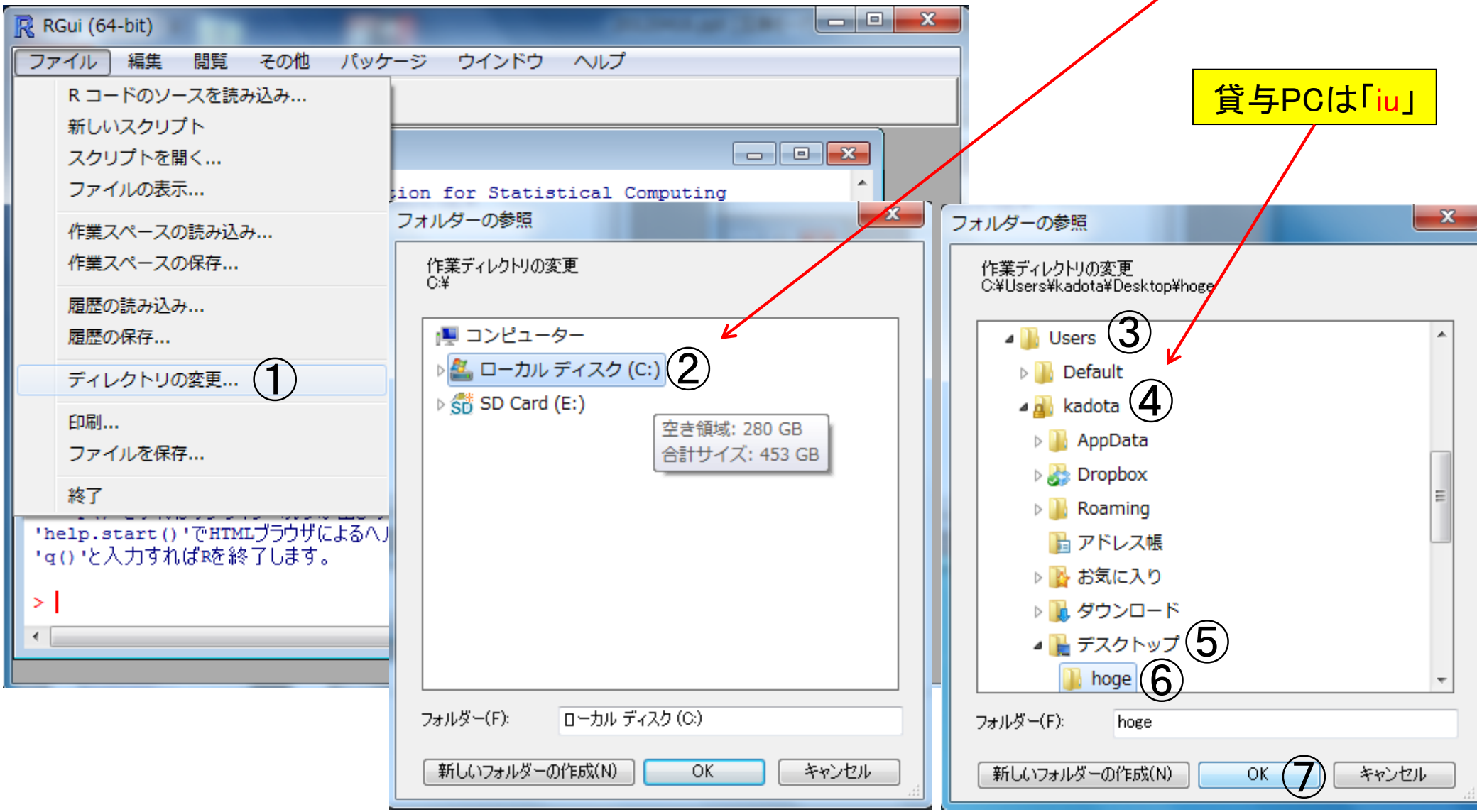


デスクトップにあるhogeフォルダ中のファイルを解析

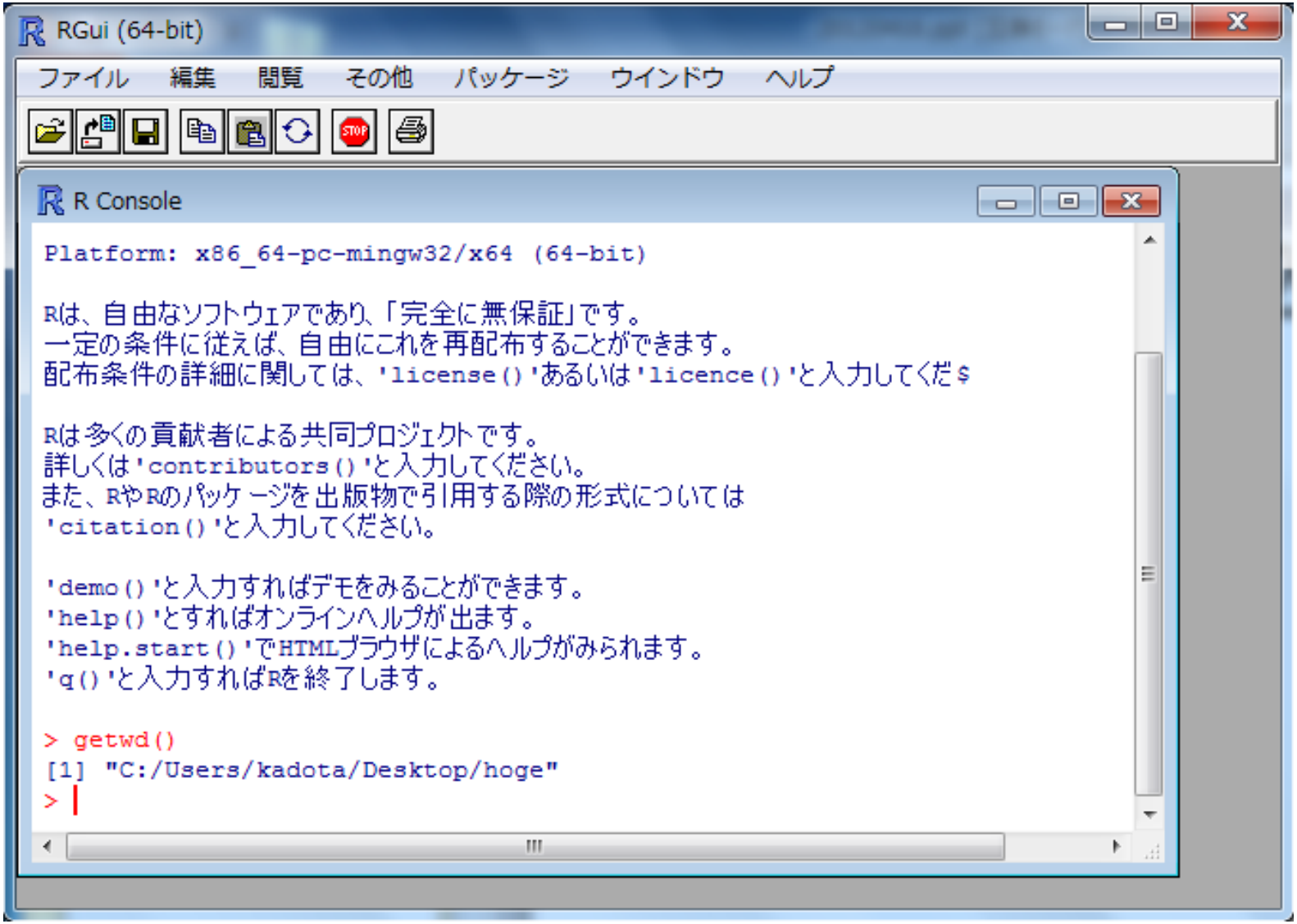
作業ディレクトリの変更

「Windows(C:)」となっている場合もあるが、気にしない

貸与PCは「iu」



getwd() と打ち込んで確認



基本はコピペ

2013年7月以降のリニューアルで、コードのコピーがやりずらくなっています。**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。

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
(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
すべてのアクセ
Send to OneN

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()'と入力してくだ
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

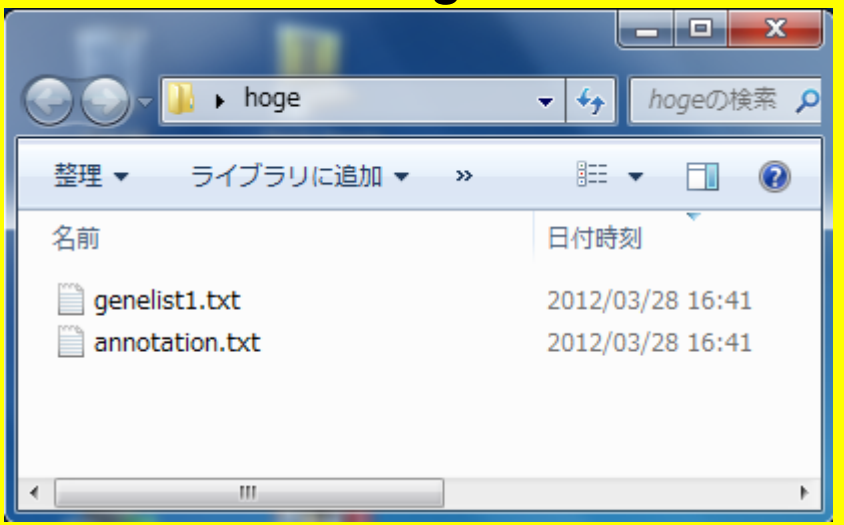
R Console

```
> 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)
>
>
```

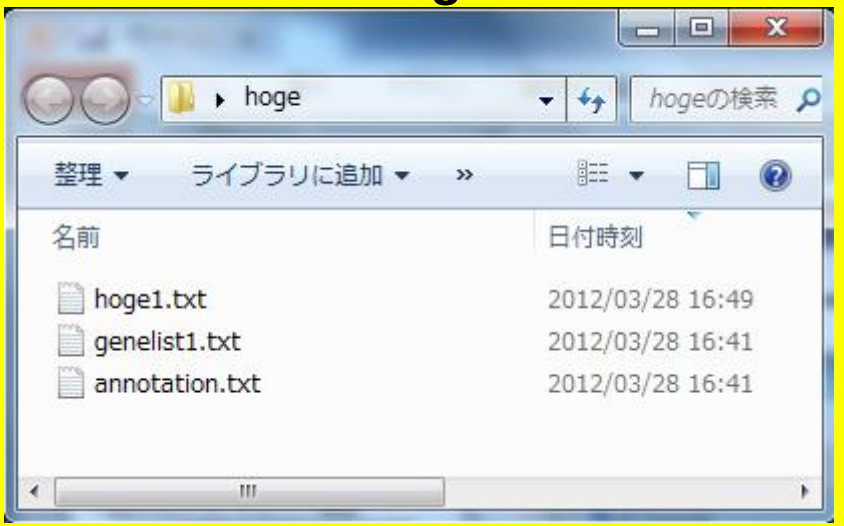
hoge1.txt

	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



(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 2014/04/10, since 2010)

What's new?

- 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」を東大農で開催します。近いうちに詳細を公開しますので興味ある方は予定を開けといてください。(2014/04/05) NEW
- 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が共立出版から出ました。マイクロアレイとRNA-seq解析を例としてRを用いてトランスクリプトーム解析を行うための体系的な本としてまとめました。数式が苦手なヒト向けに、重みつき平均の具体的な計算例などを挙げてオプションの意味などがわかるような中身の理解に重点を置いた構成にしてあります。(2014/04/10) NEW
- 参考資料(講義、講習会、本など)の項目を追加しました。(2014/04/10) NEW
- 私の所属するアグリバイオインフォマティクス教育研究プログラムでは、平成26年度も(**東大生に限らず**)バイオインフォ関連講義を行います。4/9に私の第一回目の講義がありましたが、過去最高の122名の出席がありました。例年東大以外の企業の方、研究員、学生が二割程度は受講しております。このウェブページと直接関連する講義は「ゲノム情報解析基礎」と「農学生命情報科学特論I」ですが、背景理論の説明などは「機能ゲノム学」でも行います。興味ある科目のみの受講も可能ですので、お気軽にどうぞ。(2014/04/10) NEW
- 機能解析の遺伝子オントロジー(GO)解析とパスウェイ(Pathway)解析周辺を更新し、SeqGSEAパッケージを用いた解析のみですが一通りできるようにしました。(2014/03/30) NEW

- はじめに (last modified 2014/01/30)
- 参考資料(講義、講習会、本など) (last modified 2014/04/10) NEW
- 過去のお知らせ (last modified 2014/03/23) NEW
- Rのインストールと起動 (last modified 2013/09/27)
- サンプルデータ (last modified 2014/04/01) NEW
- イントロ | 一般 | ランダムに行を抽出 (last modified 2013/10/10
- イントロ | 一般 | 任意の文字列を行の最初に挿入 (last modifie
- イントロ | 一般 | 任意のキーワードを含む行を抽出(基礎) (last
- イントロ | 一般 | ランダムな塩基配列を生成 (last modified 201
- 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 (
- イントロ | 一般 | 任意の位置の塩基を置換 (last modified 2013
- イントロ | 一般 | 指定した範囲の配列を取得 (last modified 201
- 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)

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
```

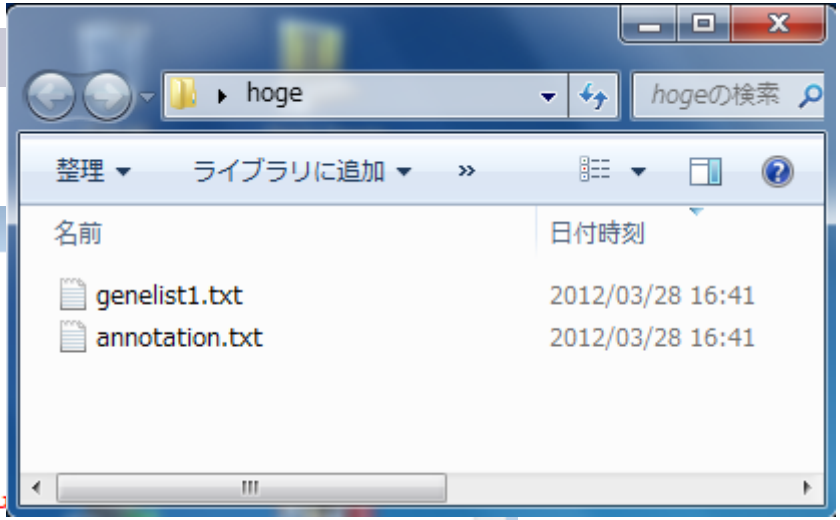
run1.txt - メモ帳

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

ありがちなミス1

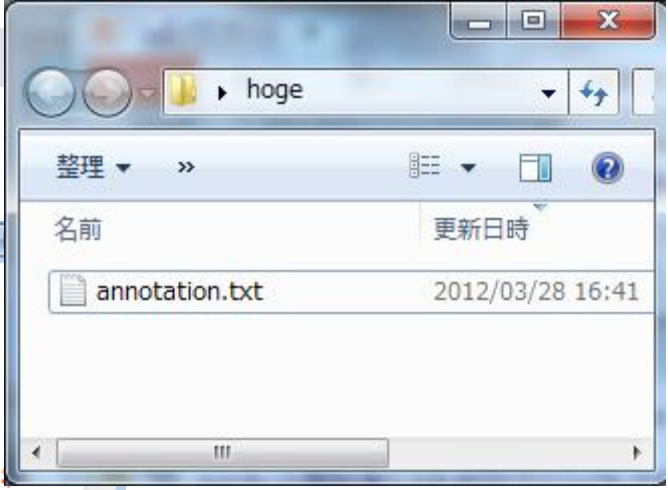


R Console

```
> 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", qu
 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)                                                       #オブジェ$
NULL
>
> #本番
> obj <- is.element(as.character(data[,param]), keywords)         #in_f1で読$
 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,]                                               #行列data$
 エラー:  オブジェクト 'obj' がありません
> dim(out)                                                        #オブジェ$
 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…

ありがちなミス2

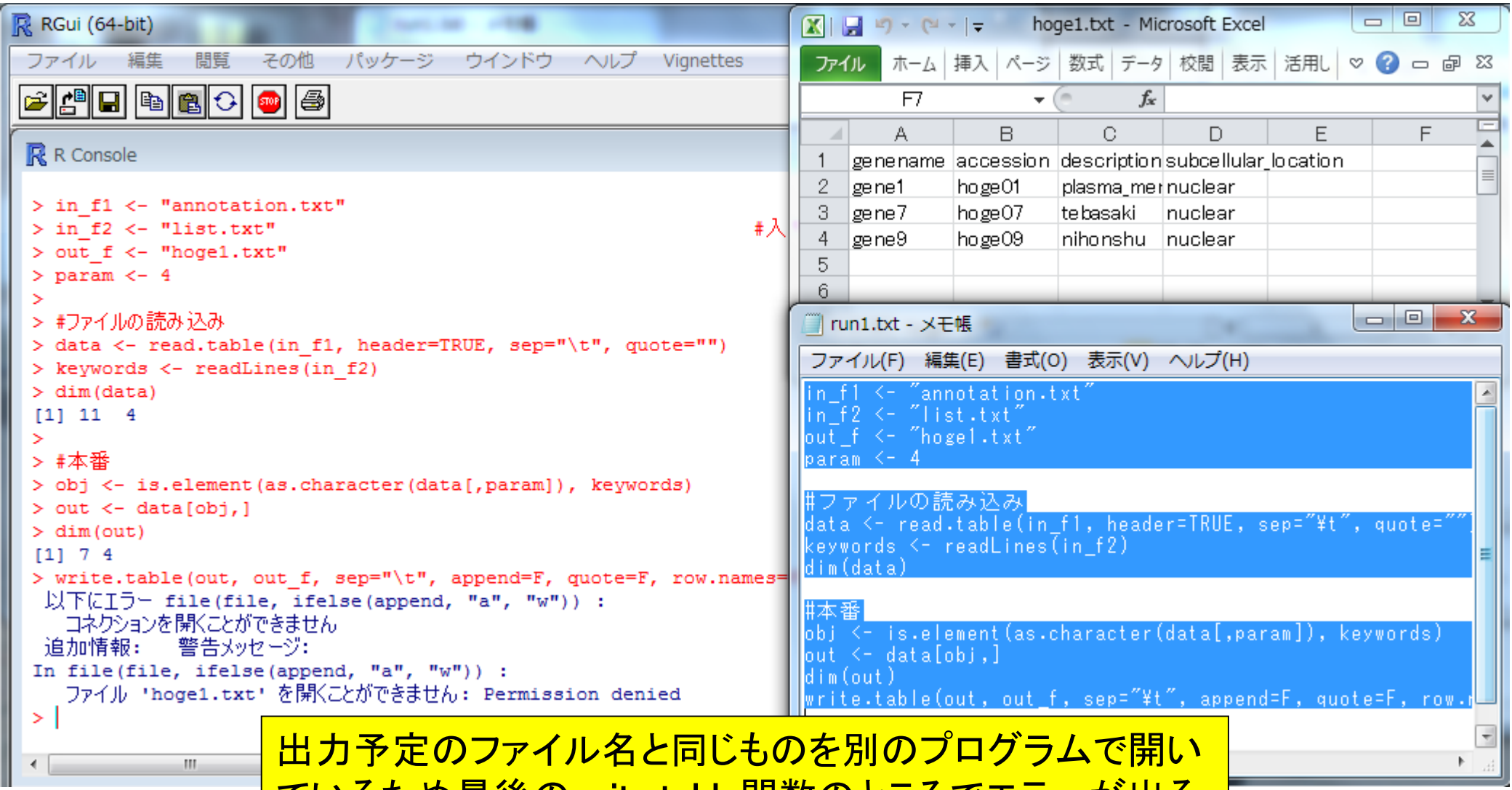


R Console

```
> 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)                                                     #オブジェクト$
 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の中身をo$
 以下にエラー is.data.frame(x) :  オブジェクト 'out' がありません
>
```

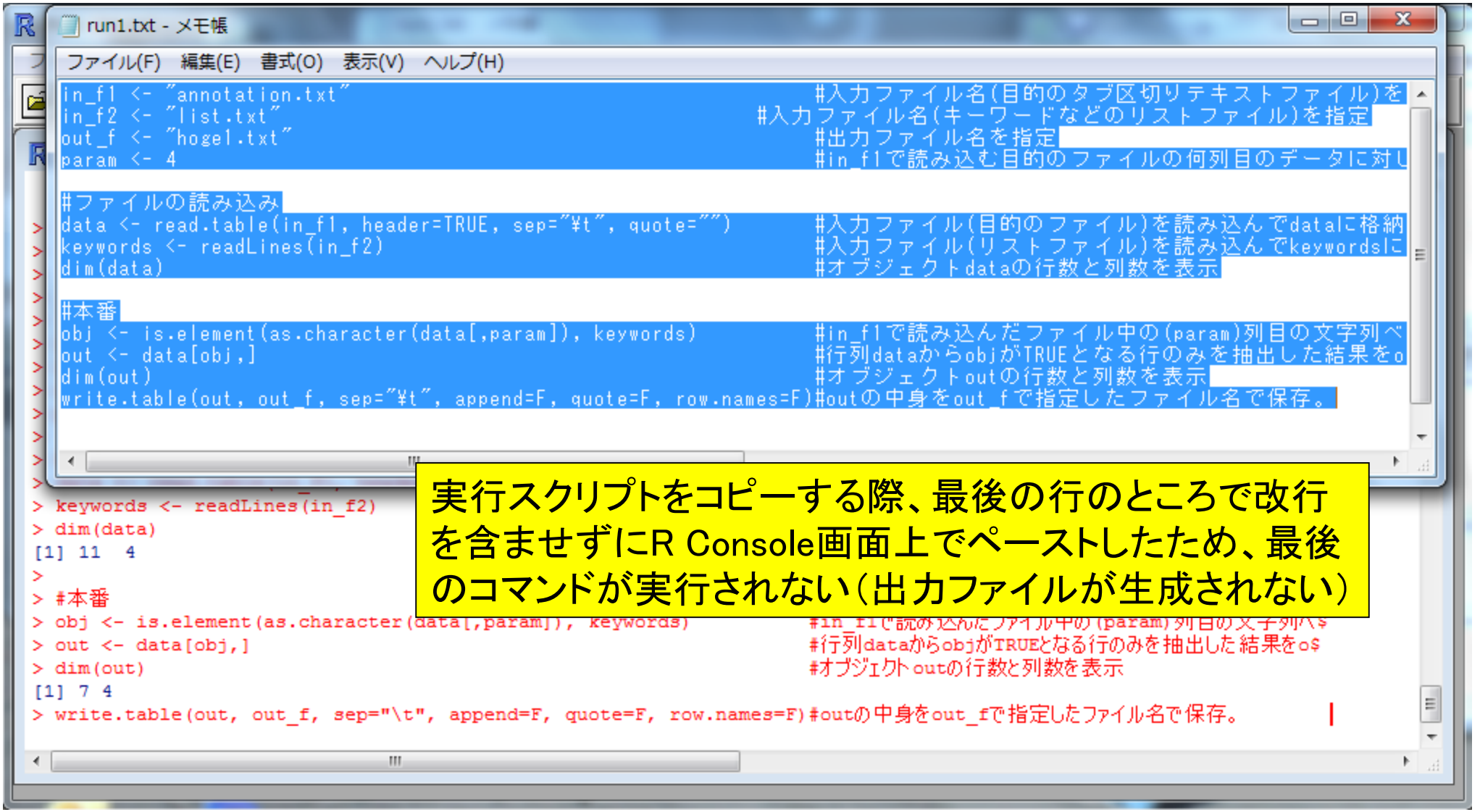
必要な入力ファイルが作業ディレクトリ中に存在しない…

ありがちなミス3



出力予定のファイル名と同じものを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

ありがちなミス4


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" #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 <- "hoge1.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="") #入力ファイル(目的のファイル)を読み込んでdataに格納
keywords <- readLines(in_f2) #入力ファイル(リストファイル)を読み込んでkeywordsに
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) #in_f1で読み込んだファイル中の(param)列目の文字列べ
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。
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)
> 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=F)#outの中身をout_fで指定したファイル名で保存。

改行を入れておいたほうが警告が出ない

list.txt

```
nuclear
membrane
```

list.txt *

```
nuclear
membrane
```

R Console

```
> in_f1 <- "annotation.txt"                    #入力$
> in_f2 <- "list.txt"                          #入力$
> out_f <- "hoge2.txt"                         #出力$
> param <- 4                                   #アノ$
>
> #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"                    #入力ファイル名を指定してin_f$
> in_f2 <- "list.txt"                          #入力ファイル名を指定してin_f$
> out_f <- "hoge2.txt"                         #出力ファイル名を指定してout_$
> param <- 4                                   #アノテーションファイル中の検$
>
> #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。

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
2     gene2     hoge02      hohinu
3     gene3     hoge03     agribio
4     gene4     hoge04     genesi
5     gene5     hoge05        kam
6     gene6     hoge06     netteb
7     gene7     hoge07    tebasak
8     gene8     hoge08       biir
9     gene9     hoge09    nihonshu
10   gene10     hoge10      agene1
11   gene11     hoge11      iyaaaa
> 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

genelist1.txt

	A
1	gene1
2	gene7
3	gene9

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

#入力ファイルの読
data <- read.tab
keywords <- read
dim(data)

#本番
obj <- is.elemen
out <- data[obj,
dim(out)

#ファイルに保存
write.table(out,
```

R Console

```
> obj
 [1]  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE
> 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
10   gene10     hoge10      agene1
11   gene11     hoge11      iyaaaa
> obj
 [1]  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  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

genelist1.txt

	A
1	gene1
2	gene7
3	gene9

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

#本番
obj <- is.element(as.character(data[,param]), keywords)#条件を満たすかどう
out <- data[obj,]                                      #obj
dim(out)                                               #オ

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

R Console

```
>
> keywords                          #keywordsの中身を表示
[1] "gene1" "gene7" "gene9"
> length(keywords)                  #keywordsベクトルの要素数を表示
[1] 3
> keywords[3]                       #3番目の要素を表示
[1] "gene9"
> keywords[4]                       #4番目の要素は...ない
[1] NA
> keywords == "gene7"               #"gene7"という文字の位置情報を表示
[1] FALSE  TRUE FALSE
> obj <- keywords == "gene7" #上記結果をobjに格納
> keywords[obj]                     #objがTRUEとなる要素のみを表示
[1] "gene7"
>
```

ここも分かりづらいところではある…

論理値ベクトルを理解

genelist1.txt

	A
1	gene1
2	gene7
3	gene9

R Console

```
> hoge <- c("gene7", "gene9")#二つの要素からなるベクトルhogeを作成
> hoge                       #hogeの中身を表示させてるだけ
[1] "gene7" "gene9"
> keywords == hoge           #FALSE TRUE TRUEになることを期待したが...
[1] FALSE FALSE FALSE
 警告メッセージ:
In keywords == hoge :
    長いオブジェクトの長さが短いオブジェクトの長さの倍数になっていません
> is.element(keywords, hoge) #keywords中の各要素は集合hogeに含まれるか否か
[1] FALSE  TRUE  TRUE
>
```

疑問に思ったら、自分の理解できるところから試す

```
#本番
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
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
```

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
```

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。


```
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
```

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:


```
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

入力ファイルはhoge
フォルダ内にあります

課題1

- シロイヌナズナのGFF3形式ファイル（TAIR10_GFF3_genes.gff）を入力として下記の解析結果を示せ。また、3と4の結果の違いについて論ぜよ
  1. 染色体数（3列目が"chromosome"となっている行数）
  2. mRNAの個数（3列目が"mRNA"となっている行数）
  3. 1列目が"Chr1"および"ChrM"となっているトータルの行数
  4. 1列目が"chromosome 1"および"ChrM "となっているトータルの行数

	A	B	C	D	E	F	G	H	I
1	Chr1	TAIR10	chromosome	1	30427671	.	.	.	ID=Chr1;Name=Chr1
2	Chr1	TAIR10	gene	3631	5899	.	+	.	ID=AT1G01010;Note=protein_coding_gene;Name=AT1G01010
3	Chr1	TAIR10	mRNA	3631	5899	.	+	.	ID=AT1G01010.1;Parent=AT1G01010;Name=AT1G01010.1;Index=1
4	Chr1	TAIR10	protein	3760	5630	.	+	.	ID=AT1G01010.1-Protein;Name=AT1G01010.1;Derives_from=AT1G01010.1
5	Chr1	TAIR10	exon	3631	3913	.	+	.	Parent=AT1G01010.1
6	Chr1	TAIR10	five_prime_UTR	3631	3759	.	+	.	Parent=AT1G01010.1
7	Chr1	TAIR10	CDS	3760	3913	.	+	0	Parent=AT1G01010.1,AT1G01010.1-Protein;
8	Chr1	TAIR10	exon	3996	4276	.	+	.	Parent=AT1G01010.1
9	Chr1	TAIR10	CDS	3996	4276	.	+	2	Parent=AT1G01010.1,AT1G01010.1-Protein;
10	Chr1	TAIR10	exon	4486	4605	.	+	.	Parent=AT1G01010.1
11	Chr1	TAIR10	CDS	4486	4605	.	+	0	Parent=AT1G01010.1,AT1G01010.1-Protein;
12	Chr1	TAIR10	exon	4706	5095	.	+	.	Parent=AT1G01010.1

入力ファイルのヘッダー行の有無に気をつけよう

課題1の3.と4.のヒント

R Console

```
> in_f <- "TAIR10_GFF3_genes.gff"          #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)
> 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
[1] 590264      9
> head(data)                               #最初の6行分を表示
    V1     V2              V3   V4       V5 V6 V7 V8                                                                  V9
1 Chr1 TAIR10     chromosome    1 30427671  .  .  .                                                        ID=Chr1;Name=Chr1
2 Chr1 TAIR10           gene 3631     5899  .  +  .            ID=AT1G01010;Note=protein_coding_gene;Name=AT1G01010
3 Chr1 TAIR10           mRNA 3631     5899  .  +  .         ID=AT1G01010.1;Parent=AT1G01010;Name=AT1G01010.1;Index=1
4 Chr1 TAIR10        protein 3760     5630  .  +  . ID=AT1G01010.1-Protein;Name=AT1G01010.1;Derives_from=AT1G01010.1
5 Chr1 TAIR10           exon 3631     3913  .  +  .                                                      Parent=AT1G01010.1
6 Chr1 TAIR10 five_prime_UTR 3631     3759  .  +  .                                                      Parent=AT1G01010.1
> head(data[,1])
[1] Chr1 Chr1 Chr1 Chr1 Chr1 Chr1
Levels: Chr1 Chr2 Chr3 Chr4 Chr5 ChrC ChrM
> table(data[,1])

  Chr1   Chr2   Chr3   Chr4   Chr5   ChrC   ChrM
157712  91857 113968  90371 135017    616    723
> 157712+91857+113968+90371+135017+616+723
[1] 590264
>
```

Levelsのところはこのベクトル中に存在する文字列が表示されている。この場合全部で7種類。

table関数実行結果は文字列ごとの出現回数

multi-FASTAファイルからの各種情報抽出

(Rで)塩基配列解析(主にNGSやRNA-seq解析)

(last modified 2014/02/06, since 2010)

What's ne

- 項目名の
- 一連の解
  や発現変
- 発現変動
  新版を利
- どのブラ
  た。(201
- 2013年7
  可能です

- はじめに
- Rのイン
- サンプル
- イントロ
- イントロ
- イントロ
- イントロ
- イントロ
- イントロ
- イントロ
- イントロ
- イントロ
- イントロ
- イントロ

- 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 2014/02/05) 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 2013/08/15)
- イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome(last modified 2012/02/05)
- イントロ | 一般 | 配列取得 | プロモーター配列 | BSgenome(last modified 2013/10/10)
- イントロ | 一般 | 配列取得 | プロモーター配列 | GenomicFeatures(L
- イントロ | 一般 | 配列取得 | トランスクリプトーム配列 | biomaRt(Du
- イントロ | NGS | 様々なプラットフォーム(last modified 2013/06/12)
- イントロ | NGS | qPCRやmicroarrayなどとの比較(last modified 201
- イントロ | NGS | Viewer(last modified 2014/01/29) NEW
- イントロ | NGS | 配列取得 | FASTQ or SRALite | 公共DBから(last
- イントロ | NGS | 配列取得 | FASTQ or SRALite | SRAdb(Zhu_201
- イントロ | NGS | アノテーション情報取得 | GFF/GTF形式ファイル(
- イントロ | NGS | アノテーション情報取得 | refFlat形式ファイル(last
- イントロ | NGS | アノテーション情報取得 | biomaRt(Durinck_2009)
- イントロ | NGS | アノテーション情報取得 | TranscriptDb | TxDb.*か
- イントロ | NGS | アノテーション情報取得 | TranscriptDb | GenomicF
- イントロ | NGS | アノテーション情報取得 | TranscriptDb | GFF/GTF
- イントロ | NGS | 読み込み | FASTA形 | 基本情報を取得(last mo
- イントロ | NGS | 読み込み | FASTA形 | description行の 記述を整
- イントロ | NGS | 読み込み | FASTQ形式(last modified 2013/06/17)
- イントロ | NGS | 読み込み | FASTQ形式 | description行の 記述を整
- イントロ | NGS | 読み込み | Illuminaの*_seq.txt(last modified 2013
- イントロ | NGS | 読み込み | Illuminaの*_qseq.txt(last modified 201
- イントロ | ファイル形式の変換 | について(last modified 2013/09/30
- イントロ | ファイル形式の変換 | BAM --> BED(last modified 2013/

multi-FASTAって何?

イントロ | NGS | 読み込み | FASTA形式 | 基本情報を取得 NEW

multi-fastaファイルを読み込んで、Total lengthやaverage lengthなどの各種情報取得を行うためのやり方を示します。
「ファイル」-「ディレクトリの変更」でファイルを保存したいディレクトリに移動し以下をコピペ。

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

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

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

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

#本番(基本情報取得)
Total_len <- sum(width(fasta))        #コンティグの「トータルの長さ」を取得
Number_of_contigs <- length(fasta)    #「コンティグ数」を取得
Average_len <- mean(width(fasta))     #コンティグの「平均長」を取得
Median_len <- median(width(fasta))    #コンティグの「中央値」を取得
Max_len <- max(width(fasta))          #コンティグの長さの「最大値」を取得
Min_len <- min(width(fasta))          #コンティグの長さの「最小値」を取得

#本番(N50情報取得)
sorted <- rev(sort(width(fasta)))     #長さ情報を降順にソートした結果をsortedに格納
N50 <- sorted[cumsum(sorted) >= Total_len/2][1]#「N50」(Total_lenの50%に達したときのコンティグ長)を取

#本番(GC含量情報取得)
```

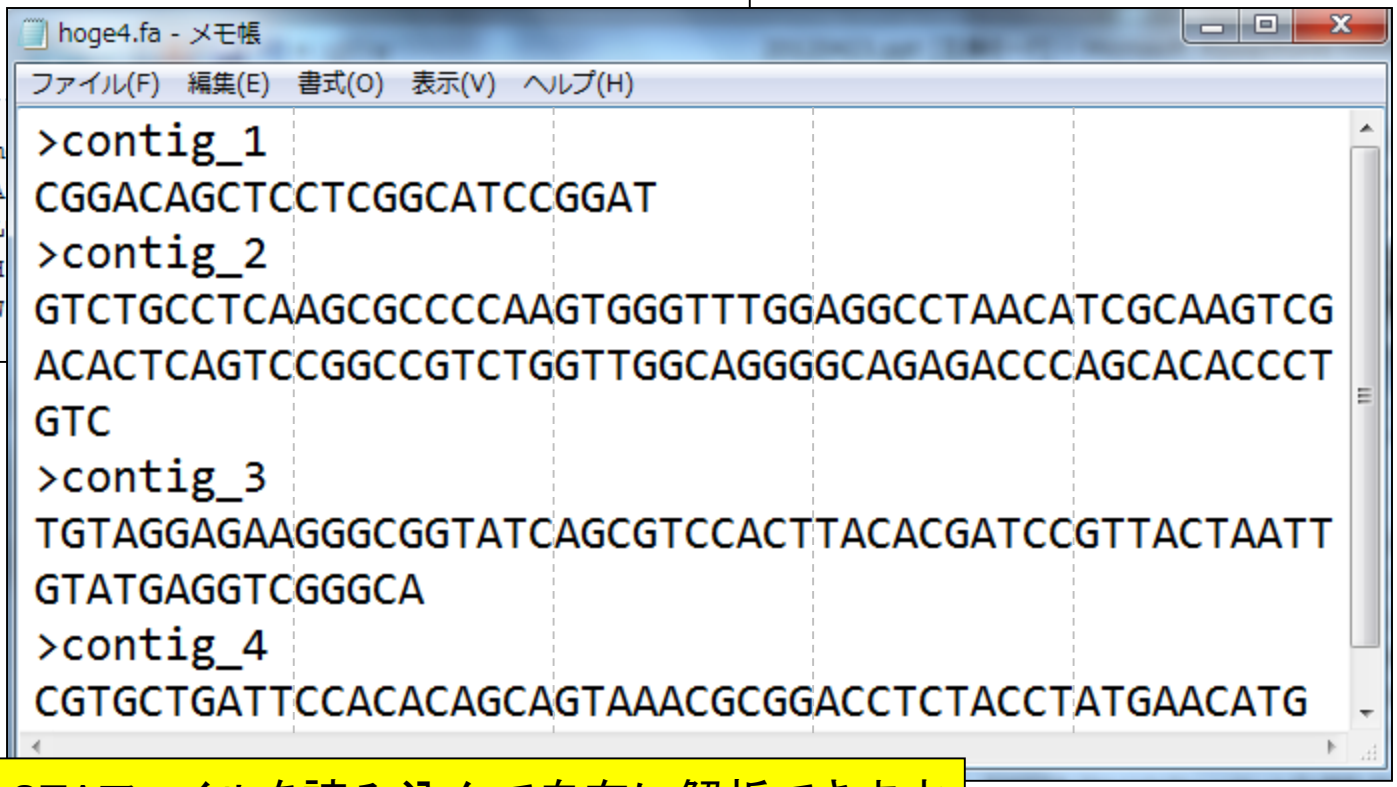
multi-FASTAファイルからの各種情報抽出

FASTAフォーマット [編集]

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

FASTA ファイルフォーマットの例を示す。

```
>gi|5524211|gb|AAD44166.1| cytochrom
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATA
EWIWGGFSVDKATLNRFFAFHFILPFTMVALAGVHL
LLILILLLLLLALLSPDMLGDPDNHMPADPLNTPLH
GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTW
IENY
```



Rでmulti-FASTAファイルを読み込んで自在に解析できます

コピー（CTRL+ALT+左クリック）&ペースト

イントロ | NGS | 読み込み | FASTA形式 | 基本情報を取得 NEW

multi-fastaファイルを読み込んで、Total lengthやaverage lengthなどの各種情報取得を行うためのやり方を示します。
「ファイル」-「ディレクトリの変更」でファイルを保存したいディレクトリに移動し以下をコピペ。

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

```
in_f <- "hoge4.fa"                    #入力ファイル
out_f <- "hoge1.txt"

#必要なパッケージをロー
library(Biostrings)

#入力ファイルの読み込み
fasta <- readDNAStrin

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

#本番(N50情報取得)
sorted <- rev(sort(wi
N50 <- sorted[cumsum(sorted) >= Total_len/2][1]#「N

#本番(GC含量情報取得)
```

①

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(N)...
Bing でマップ
Bing で翻訳
Google で検索
電子メール (Windows Live Hotmail)
すべてのアクセラレータ
Send to OneNote

hoge4.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

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

RGui (64-bit)

ファイル 編集 閲覧 その他

R Console

```
> #GC含量(GC content)計算のと
> count <- alphabetFrequency
> CG <- rowSums(count[,2:3])
> ACGT <- rowSums(count[,1:4])
> GC_content <- sum(CG)/sum(ACGT)
>
> #出力用に結果をまとめて(w
> tmp <- NULL
> tmp <- rbind(tmp, c("Total length (bp)", Total_length))
> tmp <- rbind(tmp, c("Number of contigs", Number_of_contigs))
> tmp <- rbind(tmp, c("Average length", Average_length))
> tmp <- rbind(tmp, c("Median length", Median_length))
> tmp <- rbind(tmp, c("Max length", Max_length))
> tmp <- rbind(tmp, c("Min length", Min_length))
> tmp <- rbind(tmp, c("N50", N50))
> tmp <- rbind(tmp, c("GC content", GC_content))
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)#tmpの$
>
>
```

②

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

hogeフォルダにhoge1.txtが作成されているはず

結果ファイルを眺めて動作確認

ID	Length
contig_1	24
contig_2	103
contig_3	65
contig_4	49

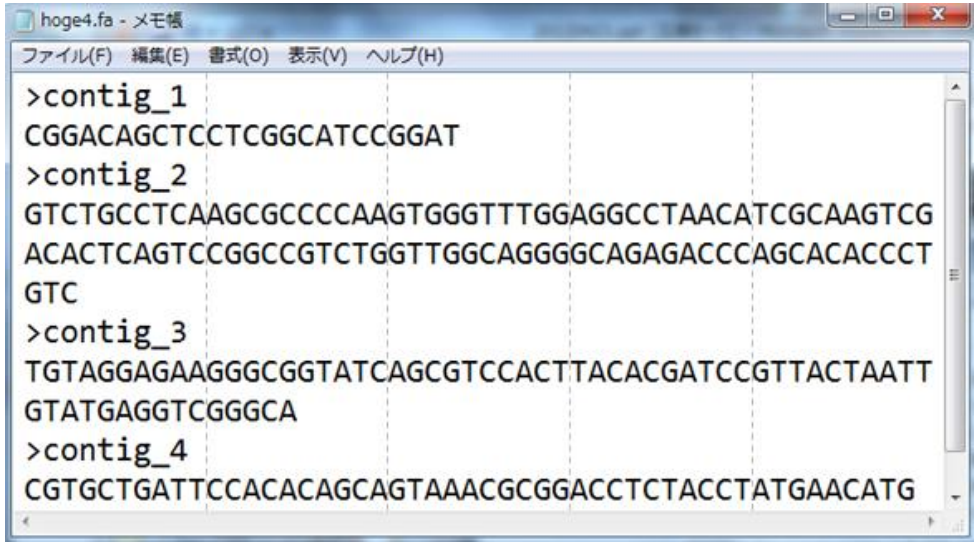
イントロ | NGS | 読み込み | FASTA形式 | 基本情報を取得 NEW

multi-fastaファイルを読み込んで、Total lengthやaverage lengthなどの各種情報取得を行うためのやり方を示します。
「ファイル」-「ディレクトリの変更」でファイルを保存したいディレクトリに移動し以下をコピペ。

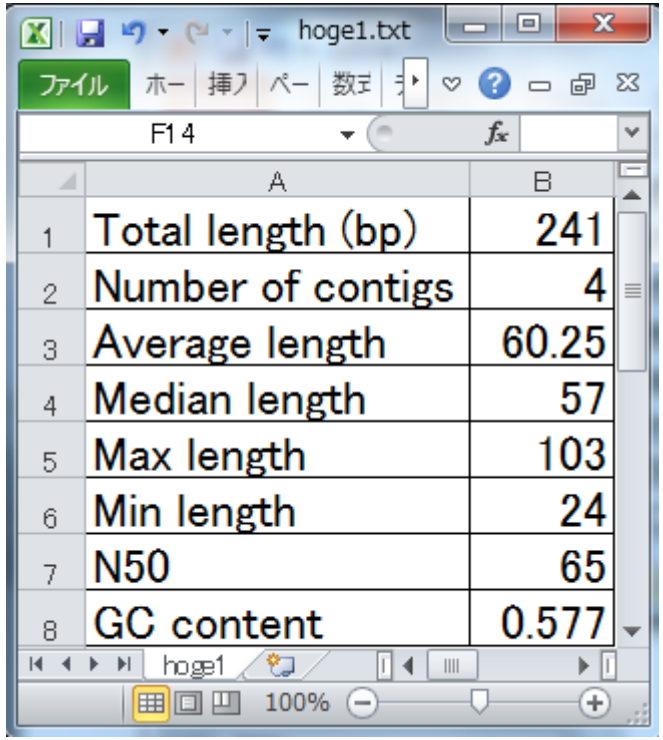
1. イントロ | 一般 | ランダムな塩基配列を作成の4.を実行して得られたmulti-fastaファイル(hoge4.fa)の場合:

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

入力: hoge4.fa

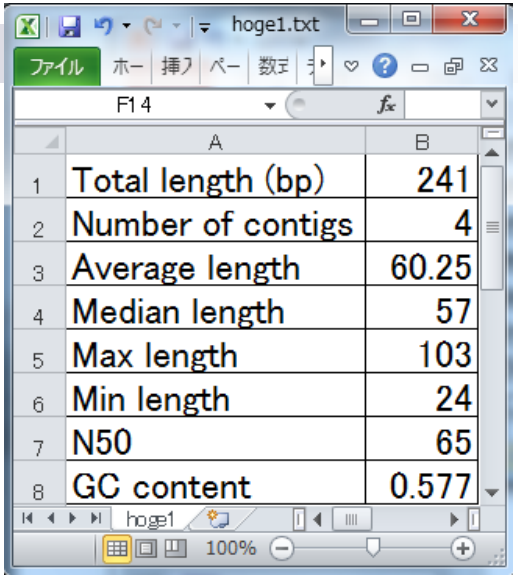


出力: hoge1.txt

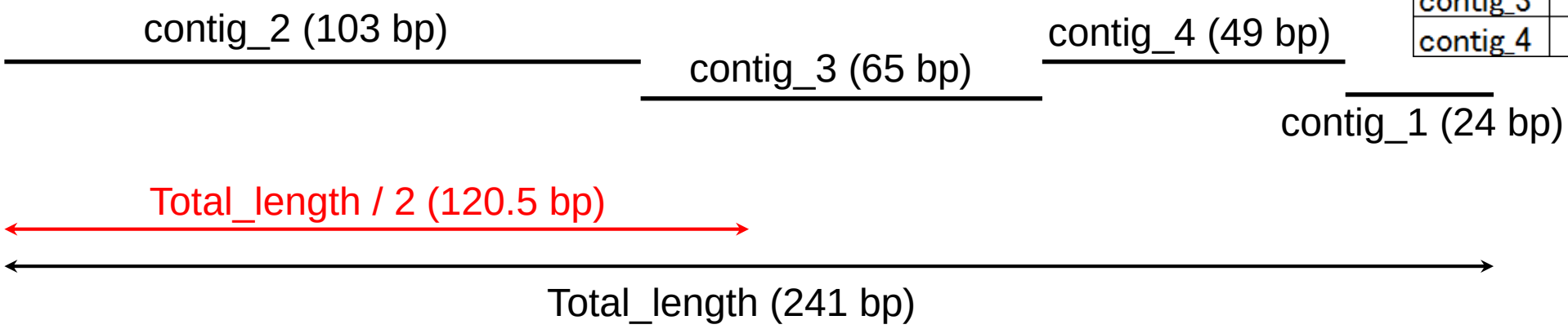


N50

- アセンブル結果の評価基準の一つ
  - 長いコンティグから足していってTotal_lengthの50%に達したときのコンティグの長さ
  - 一般に数値が大きいほどよい



ID	Length
contig_1	24
contig_2	103
contig_3	65
contig_4	49



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

情報抽出手順の一部

イントロ | NGS | 読み込み | FASTA形式 | 基本情報を取得 NEW

multi-fastaファイルを読み込んで、Total lengthやaverage lengthなどの各種情報取得を行うためのやり方を示します。
「ファイル」-「ディレクトリの変更」でファイルを保存したいディレクトリに移動し以下をコピペ。

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

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

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

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

#本番(基本情報取得)
Total_len <- sum(width(fasta))        #コンティグの「トータルの長さ」を取得
Number_of_contigs <- length(fasta)    #「コンティグ数」を
Average_len <- mean(width(fasta))     #コンティグの「平均
Median_len <- median(width(fasta))    #コンティグの「中央
Max_len <- max(width(fasta))          #コンティグの長さの
Min_len <- min(width(fasta))          #コンティグの長さの
```

R Console

```
> fasta
  A DNAStringSet instance of length 4
    width seq                                          names
[1]    24 CGGACAGCTCCTCGGCATCCGGAT                     contig_1
[2]   103 GTCTGCCTCAAGCGCC...CCCAGCACACCCTGTC         contig_2
[3]    65 TGTAGGAGAAGGGCGG...TGTATGAGGTCGGGCA         contig_3
[4]    49 CGTGCTGATTCCACAC...TCTACCTATGAACATG         contig_4
> width(fasta)
[1]  24 103  65  49
> sum(width(fasta))
[1] 241
>
```

width関数を使えば配列長情報を取り出せるようだ

情報抽出手順の一部

ID	Length
contig_1	24
contig_2	103
contig_3	65
contig_4	49

R Console

```
> fasta[1]
  A DNAStringSet instance of length 1
    width seq                                         names
[1]    24 CGGACAGCTCCTCGGCATCCGGAT                    contig_1
> fasta[2]
  A DNAStringSet instance of length 1
    width seq                                         names
[1]   103 GTCTGCCTCAAGCGCC...CCCAGCACACCCTGTC        contig_2
> width(fasta) >= 50
[1] FALSE  TRUE  TRUE FALSE
> fasta[width(fasta) >= 50]
  A DNAStringSet instance of length 2
    width seq                                         names
[1]   103 GTCTGCCTCAAGCGCC...CCCAGCACACCCTGTC        contig_2
[2]    65 TGTAGGAGAAGGGCGG...TGTATGAGGTCGGGCA        contig_3
>
```

50 bp以上のコンティグからなるサブセットの抽出ができそうだ!

上矢印キーを押すと直前に打ったコマンドが出る。有効に利用し最小限の労力で打つ。

コードの中身が分かると応用範囲が拡大

- 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 2013/06/15)
- 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 2013/06/15)
- 前処理 | フィルタリング | ACGTのみからなる配列を抽出 (last modified 2013/06/18)
- 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 2013/06/18)
- 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
- 前処理 | フィルタリング | Illuminaの pass filterin
- 前処理 | フィルタリング | GFF/GTF形式ファイル
- 前処理 | フィルタリング | 組合せ | ACGTのみ
- 前処理 | トリミング | ポリA配列除去 (last modi
- 前処理 | トリミング | アダプター配列除去(基礎
- 前処理 | トリミング | アダプター配列除去(組合
- 前処理 | トリミング | アダプター配列除去(giraf
- 前処理 | トリミング | 指定した末端塩基数だけ
- アセンブル | について (last modified 2011/07/2
- アセンブル | ゲノム用 (last modified 2013/06/1
- アセンブル | トランスクリプトーム(転写物)用 (l
- アセンブル | ゲノム既知で転写物構造推定用

指定した配列長以下のものを抽出したいときは「<=」とすればよい

前処理 | フィルタリング | 指定した長さ以上の配列を抽出 NEW

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

1. FASTA形式ファイル(hoge4.fa)の場合:

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

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

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

#本番
obj <- as.logical(width(fasta) >= param)#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
```

イントロ | 一般 | 指定した範囲の配列を取得 NEW

single-FASTA形式やmulti-FASTA形式ファイルから様々な部分配列を取得するやり方を示します。
「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. (single-)FASTA形式ファイル(sample1.fasta)の場合:

任意の範囲 (始点が3, 終点が9)の配列を抽出し、得られた部分配列をFASTA形式ファイル(hoge1.txt)に出力するやり方です。

```
in_f <- "sample1.fasta"                    #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.txt"                       #出力ファイル名を指定してout_fに格納
param <- c(3, 9)                           #抽出したい範囲の始点と終点を指定

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

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

#本番
fasta <- subseq(fasta, param[1], param[2])#paramで指定した始点と終点の範囲の配列を抽出し
fasta                                      #確認してるだけです

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

入力 : sample1.fasta

```
>kadota
AGTGACGGTCTT
```

出力 : hoge1.txt

```
>kadota
TGACGGT
```

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

任意の範囲(始点が3, 終点が9)の配列を抽出し、得られた部分配列をFASTA形式ファイル(hoge1.txt)に

```
in_f <- "sample1.fasta"                    #入力ファイル名を指定してin_fに格納
out_f <- "hoge1.txt"                       #出力ファイル名を指定してout_fに格納
param <- c(3, 9)                           #抽出したい範囲の始点と終点を指定

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

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

#本番
fasta <- subseq(fasta, param[1], param[2])#paramで指定した始点と終点の範囲の配列
fasta

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


入力:sample1.fasta

```
>kadota
AGTGACGGTCTT
```

出力:hoge1.txt

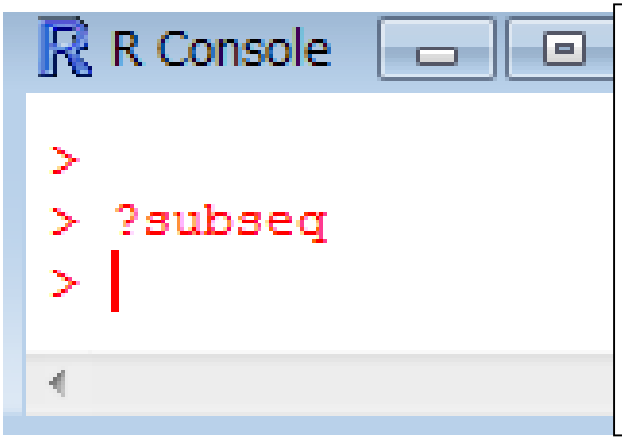
```
>kadota
TGACGGT
```

R Console

```
> fasta
  A DNAStringSet instance of length 1
    width seq                 names
[1]    12 AGTGACGGTCTT        kadota
> subseq(fasta, param[1], param[2])
  A DNAStringSet instance of length 1
    width seq                 names
[1]     7 TGACGGT             kadota
> param
[1] 3 9
> subseq(fasta, 3, 9)
  A DNAStringSet instance of length 1
    width seq                 names
[1]     7 TGACGGT             kadota
>
```

subseq関数は「塩基配列, start, end」という形式がデフォルトのようだ

関数の使用法について



XVector-class {IRanges} R Documentation

XVector objects

Description

The XVector virtual class is a general container for storing an "external vector". It inherits from the Vector, which has a very rich interface.

The following classes derive directly from the XVector class:

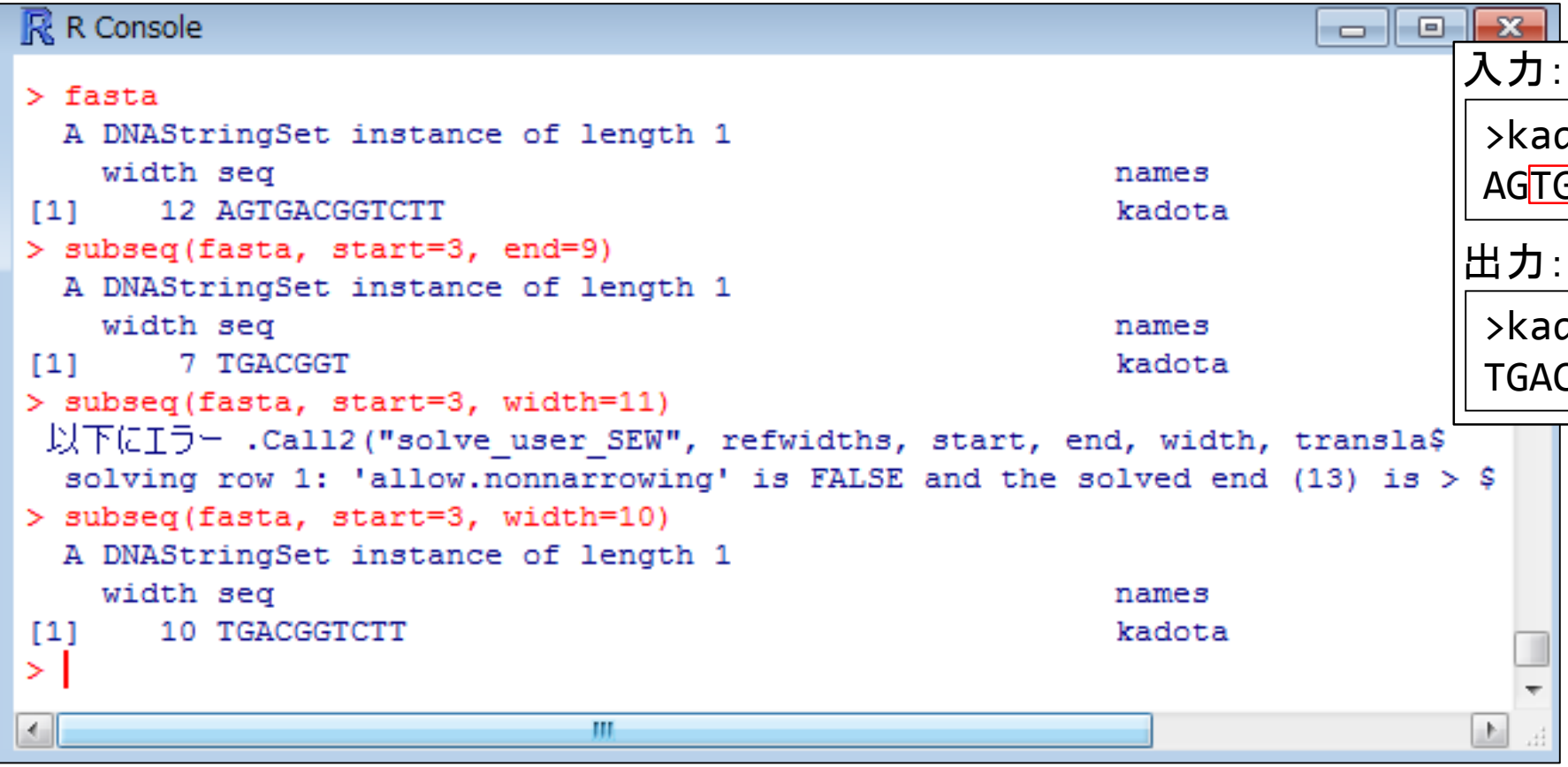
...

```
subseq(x4, start=10)
subseq(x4, start=-10)
subseq(x4, start=-20, end=-10)
subseq(x4, start=10, width=5)
subseq(x4, end=10, width=5)
subseq(x4, end=10, width=0)

x3[length(x3):1]
x3[length(x3):1, drop=FALSE]
```

[Package *IRanges* version 1.12.6 Index]

・**?関数名**で使用法を記したウェブページが開く
・ページの下のほうに、大抵の場合使用例が掲載されている
・使用法既知の関数のマニュアルをいくつか読んで慣れておく

原因既知状態で意図的にエラーを出す



入力：sample1.fasta

```
>kadota
AGTGACGGTCTT
```

出力：hoge1.txt

```
>kadota
TGACGGT
```

マニュアルの使用例をいくつか試して、ステップアップ

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

目的のaccession番号が複数ある場合に対応したものです。予め用意しておいた「1列目:accession, 2列目:start位置, 3列目:end位置」からなるリストファイル (list_sub2.txt) を読み込ませて、目的の配列のmulti-fastaファイルをゲットするやり方です。

```
in_f1 <- "hoge4.fa"                    #入力ファイル名を指定してin_f1に格納(multi-fastaファイル)
in_f2 <- "list_sub2.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 <- "hoge4.txt"                   #出力ファイル名を指定してout_fに格納

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

#入力ファイルの読み込み
fasta <- readDNAStringSet(in_f1, format="fasta")#in_f1で指定したファイルの読み込み
posi <- read.table(in_f2)              #in_f2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 <- NULL                           #最終的に得る結果を格納するためのプレースホルダhogeを作成して
for(i in 1:nrow(posi)){                #length(posi)回だけループを回す
    obj <- names(fasta) == posi[i,1]   #条件を満たすかどうかを判定した結果をobjに格納
    hoge <- append(hoge, subseq(fasta[obj], start=posi[i,2], end=posi[i,3]))#subseq関数を用いて
}
```

入力1: hoge4.fa

hoge4.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

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

入力2: list_sub2.txt

contig_4	1	10
contig_2	3	8

出力:hog4.txt

```
>contig_4
CGTGCTGATT
>contig_2
CTGCCT
```

- 正規化 | サンプル間 | 3群間 | 複製あり | iDEGES/edgeR(Sun 2013)(last modified 2013/09/15)推奨
- 正規化 | サンプル間 | 3群間 | 複製あり | TMM(Robinson 2010)(last modified 2013/09/16)
- 解析 | 一般 | アラインメント(ペアワイズ;基本編1) (last modified 2010/6/8)
- 解析 | 一般 | アラインメント(ペアワイズ;基本編2) (last modified 2010/6/8)
- 解析 | 一般 | アラインメント(ペアワイズ;応用編) (last modified 2010/6/8)
- 解析 | 一般 | パターンマッチング (last modified 2013/06/19)
- 解析 | 一般 | GC含量 (GC contents) (last modified 2013/06/24)
- 解析 | 一般 | Sequence logos(Schneider 1990)(last modified 2012/06/27)
- 解析 | 一般 | 上流配列解析 | LDSS(Yamamoto 2007)(last modified 2012/07/17)
- 解析 | 一般 | 上流配列解析 | Relative
- 解析 | 基礎 | 平均-分散プロット(Techn
- 解析 | 基礎 | 平均-分散プロット(Biolog
- 解析 | クラスタリング | について(last m
- 解析 | クラスタリング | サンプル間 | hc
- 解析 | クラスタリング | 遺伝子間 | MB
- 解析 | 発現変動 | ポアソン分布 | シミュ
- 解析 | 発現変動 | 負の二項分布 | シミ
- 解析 | 発現変動 | について(last modif
- 解析 | 発現変動 | 2群間 | 対応なし | に
- 解析 | 発現変動 | 2群間 | 対応なし | 複
- 解析 | 発現変動 | 2群間 | 対応なし | 複

• 解析 | 一般 | GC含量 (GC contents)

配列ごとのGC含量を計算したいとき

解析 | 一般 | GC含量 (GC contents)

multi-fasta形式ファイルを読み込んで配列ごとのGC含量 (GC contents)を出力するやり方を示します。出力ファイルは、「description」「CGの総数」「ACGTの総数」「配列長」「%GC含量」としています。尚、%GC含量は「CGの総数/ACGTの総数」で計算しています。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

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

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

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

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

#本番
hoge <- alphabetFrequency(fasta)            #A,C,G,T,..の数を各配列ごとにカウントした結果を
CG <- rowSums(hoge[,2:3])                   #C,Gの総数を計算してCGに格納
ACGT <- rowSums(hoge[,1:4])                 #A,C,G,Tの総数を計算してACGTに格納
GC_content <- CG/ACGT*100                   #%GC含量を計算してGC_contentに格納

#ファイルに保存
tmp <- cbind(names(fasta), CG, ACGT, width(fasta), GC_content)#ファイルに保存したい
colnames(tmp) <- c("description", "CG", "ACGT", "Length", "%GC_contents")#列名情報
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.names=T)#tmp
```

解析 | 一般 | GC含量 (GC contents)

multi-fasta形式ファイルを読み込んで配列ごとのGC含量 (GC contents)を出力するやり方を示します。出力ファイルは、「description」「CGの総数」「ACGTの総数」「配列長」「%GC含量」としています。尚、%GC含量は「CGの総数/ACGTの総数」で計算しています。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

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

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

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

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

入力: hoge4.fa

hoge4.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

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

出力: hoge1.txt

	A	B	C	D	E
1	description	CG	ACGT	Length	%GC_contents
2	contig_1	16	24	24	66.67
3	contig_2	65	103	103	63.11
4	contig_3	33	65	65	50.77
5	contig_4	25	49	49	51.02

このコードをテンプレートとすればゲノム配列を読み込んで、染色体ごとのGC含量を算出可能

課題2

1. シロイヌナズナのゲノム配列ファイル（TAIR10_chr_all.fas）を入力として染色体ごとの%GC含量計算結果を示せ（小数点2ケタ程度でよい）。
2. 右表はシロイヌナズナゲノム配列決定に関する原著論文中のGC含量の数値である。発表当時（2000年ごろ）と2010年ごろのバージョン（TAIR10）の違いを含めて簡単に考察せよ。

原著論文の%GC含量

	Length	GC contents
chr1	28.76MB	35.80%
chr2	19.60MB	35.80%
chr3	23.17MB	35.40%
chr4	17.40MB	36.02%
chr5	25.95MB	34.50%

入力ファイルはhoge
フォルダ内にあります

GC含量

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

#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

#本番
hoge <- alphabetFrequency(fasta)            #A,C,G,T,..の数を各配
CG <- rowSums(hoge[,2:3])                   #C,Gの総数
ACGT <- rowSums(hoge[,1:4])                 #A,C,G,T
GC_content <- CG/ACGT*100                   #%GC含量

#ファイルに保存
tmp <- cbind(names(fasta), CG, ACGT, width(fast
colnames(tmp) <- c("description", "CG", "ACGT",
write.table(tmp, out_f, sep="\t", append=F, quo
```

詳細を解説します

alphabetFrequency関数を適用すると塩基ごとの出現頻度が得られます…が「M, R, W, …」は？

hoge4.fa - メモ帳

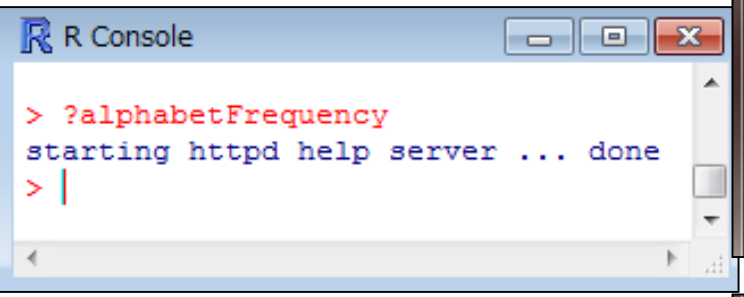
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

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

R Console

```
> fasta
  A DNAStringSet instance of length 4
    width seq                                 names
[1]    24 CGGACAGCTCCTCGGCATCCGGAT            contig_1
[2]   103 GTCTGCCTCAAGC...GCACACCCTGTC        contig_2
[3]    65 TGTAGGAGAAGGG...TGAGGTCGGGCA        contig_3
[4]    49 CGTGCTGATTCCA...CCTATGAACATG        contig_4
> hoge
     A  C  G  T M R W S Y K V H D B N - +
[1,] 4  9  7  4 0 0 0 0 0 0 0 0 0 0 0 0 0
[2,] 20 34 31 18 0 0 0 0 0 0 0 0 0 0 0 0 0
[3,] 16 13 20 16 0 0 0 0 0 0 0 0 0 0 0 0 0
[4,] 14 15 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0
> ?alphabetFrequency
starting httpd help server ... done
>
```

GC含量



alphabetFrequency関数がどういう出力結果を返すのか詳細を知りたい場合はValueのところを読むとよい

letterFrequency {Biostrings} R Documentation

Calculate the frequency of letters in a biological sequence, or the consensus matrix of a set of sequences

Description

Given a biological sequence (or a set of biological sequences), the `alphabetFrequency` function computes the frequency of each letter of the relevant alphabet.

Usage

```
alphabetFrequency(x, as.prob=FALSE, ...)
hasOnlyBaseLetters(x)
```

Arguments

x	An XString, XStringSet, XStringViews or MaskedXString object for

Details

`alphabetFrequency`, `letterFrequency`, and `letterFrequencyInSlidingView` are generic functions defined in the Biostrings package

Value

`alphabetFrequency` returns an integer vector when `x` is an XString or MaskedXString object. When `x` is an XStringSet or XStringViews object, then it returns an integer matrix with `length(x)` rows where the `i`-th row contains the frequencies for `x[[i]]`. If `x` is a DNA or RNA input, then the returned vector is named with the letters in the alphabet. If the `baseOnly` argument is `TRUE`, then the returned vector has only 5 elements: 4 elements corresponding to the 4 nucleotides + the 'other' element.

GC含量

Value

`alphabetFrequency` returns an integer vector when `x` is an XString or MaskedXString object. When `x` is an XStringSet or XStringViews object, then it returns an integer matrix with `length(x)` rows where the `i`-th row contains the frequencies for `x[[i]]`. If `x` is a DNA or RNA input, then the returned vector is named with the letters in the alphabet. If the `baseOnly` argument is `TRUE`, then the returned vector has only 5 elements: 4 elements corresponding to the 4 nucleotides + the 'other' element.

M(A/C), R(A/G), W(A/T), S(C/G), …, N(A/C/G/T)という事実は常識だから書かれていないのか?!

R Console

```
> alphabetFrequency(fasta)
      A  C  G  T M R W S Y K V H D B N - +
[1,]  4  9  7  4 0 0 0 0 0 0 0 0 0 0 0 0 0
[2,] 20 34 31 18 0 0 0 0 0 0 0 0 0 0 0 0 0
[3,] 16 13 20 16 0 0 0 0 0 0 0 0 0 0 0 0 0
[4,] 14 15 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0
> alphabetFrequency(fasta, baseOnly=T)
      A  C  G  T other
[1,]  4  9  7  4     0
[2,] 20 34 31 18     0
[3,] 16 13 20 16     0
[4,] 14 15 10 10     0
>
```

argument（引数）とは、入力や出力の形式などを指定するオプションであり、関数ごとに異なる。

GC含量

Usage

```
alphabetFrequency(x, as.prob=FALSE, ...)
hasOnlyBaseLetters(x)
uniqueLetters(x)
```

Arguments

x	An XString, XStringSet, XStringViews or MaskedXStrin object for alphabetFrequency, letterFrequency, or uniqueLetters. DNA or RNA input for hasOnlyBaseLetters. An XString object for letterFrequencyInSlidingView. A character vector, or an XStringSet or XStringViews object for consensusMatrix. A consensus matrix (as returned by consensusMatrix), or an XStringSet or XStringViews object for consensusString.
as.prob	If TRUE then probabilities are reported, otherwise counts (the default).

R Console

```
> alphabetFrequency(fasta, baseOnly=T)
      A  C  G  T other
[1,]  4  9  7  4     0
[2,] 20 34 31 18     0
[3,] 16 13 20 16     0
[4,] 14 15 10 10     0
> alphabetFrequency(fasta, baseOnly=T, as.prob=T)
          A         C         G         T other
[1,] 0.1666667 0.3750000 0.2916667 0.1666667     0
[2,] 0.1941748 0.3300971 0.3009709 0.1747573     0
[3,] 0.2461538 0.2000000 0.3076923 0.2461538     0
[4,] 0.2857143 0.3061224 0.2040816 0.2040816     0
>
```

hoge4.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

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

相対出現頻度（probability）にすることも可能です。

GC含量

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

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

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

#本番
hoge <- alphabetFrequency(fasta)            #A,C,G,T,..
CG <- rowSums(hoge[,2:3])                   #C,Gの総数を
ACGT <- rowSums(hoge[,1:4])                 #A,C,G,Tの総
GC_content <- CG/ACGT*100                   #%GC含量を計

#ファイルに保存
tmp <- cbind(names(fasta), CG, ACGT, width(fasta),
colnames(tmp) <- c("description", "CG", "ACGT", "L
write.table(tmp, out_f, sep="\t", append=F, quote=
```

「コンティグごとのCとGの出現頻度の和」は、「行の和」として得ることができる

R Console

```
> hoge
     A  C  G  T M R W S Y K V H D B N - +
[1,] 4  9  7  4 0 0 0 0 0 0 0 0 0 0 0 0 0
[2,] 20 34 31 18 0 0 0 0 0 0 0 0 0 0 0 0 0
[3,] 16 13 20 16 0 0 0 0 0 0 0 0 0 0 0 0 0
[4,] 14 15 10 10 0 0 0 0 0 0 0 0 0 0 0 0 0
> dim(hoge)
[1]  4 17
> hoge[,2:3]
     C  G
[1,] 9  7
[2,] 34 31
[3,] 13 20
[4,] 15 10
> rowSums(hoge[,2:3])
[1] 16 65 33 25
>
```

GC含量

```
in_f <- "hoge4.fa"                          #入力ファイル
out_f <- "hoge1.txt"                        #出力ファイル

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

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

#本番
hoge <- alphabetFrequency(fasta)            #A,C,G,T,..の
CG <- rowSums(hoge[,2:3])                   #C,Gの総数を
ACGT <- rowSums(hoge[,1:4])                 #A,C,G,Tの総
GC_content <- CG/ACGT*100                   #%GC含量を計

#ファイルに保存
tmp <- cbind(names(fasta), CG, ACGT, width(fasta), GC_content)#ファイルに保存したい
colnames(tmp) <- c("description", "CG", "ACGT", "Length", "%GC_contents")#列名情報
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.
```

R Console

```
> CG
[1] 16 65 33 25
> ACGT
[1]  24 103  65  49
> CG/ACGT
[1] 0.6666667 0.6310680 0.5076923 0.5102041
> CG/ACGT*100
[1] 66.66667 63.10680 50.76923 51.02041
> ATGC
 エラー:  オブジェクト 'ATGC' がありません
>
```

出力ファイルの形式やヘッダー行はどのようにして決めているのか?

hoge1.txt

	A	B	C	D	E
1	description	CG	ACGT	Length	%GC_contents
2	contig_1	16	24	24	66.67
3	contig_2	65	103	103	63.11
4	contig_3	33	65	65	50.77
5	contig_4	25	49	49	51.02

hoge1

GC含量

```
in_f <- "hoge4.fa"                          #入力ファイル
out_f <- "hoge1.txt"                        #出力ファイル

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

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

#本番
hoge <- alphabetFrequency(fasta)            #A,C,G,T,..の
CG <- rowSums(hoge[,2:3])                   #C,Gの総数を
ACGT <- rowSums(hoge[,1:4])                 #A,C,G,Tの総
GC_content <- CG/ACGT*100                   #%GC含量を計

#ファイルに保存
tmp <- cbind(names(fasta), CG, ACGT, width(fasta),
colnames(tmp) <- c("description", "CG", "ACGT", "Le
write.table(tmp, out_f, sep="\t", append=F, quote=F
```

R Console

```
> fasta
  A DNAStringSet instance of length 4
    width seq                     names
[1]    24 CGGACAGCT...ATCCGGAT contig_1
[2]   103 GTCTGCCTC...ACCCTGTC contig_2
[3]    65 TGTAGGAGA...GTCGGGCA contig_3
[4]    49 CGTGCTGAT...TGAACATG contig_4
> names(fasta)
[1] "contig_1" "contig_2" "contig_3" "contig_4"
> CG
[1] 16 65 33 25
> cbind(names(fasta), CG)
                CG
[1,] "contig_1" "16"
[2,] "contig_2" "65"
[3,] "contig_3" "33"
[4,] "contig_4" "25"
>
```

hoge1.txt

	A	B	C	D	E
1	description	CG	ACGT	Length	%GC_contents
2	contig_1	16	24	24	66.67
3	contig_2	65	103	103	63.11
4	contig_3	33	65	65	50.77
5	contig_4	25	49	49	51.02

hoge1

同じ要素数からなるベクトル同士を列（column）方向で結合（**bind**）

GC含量

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

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

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

#本番
hoge <- alphabetFrequency(fasta)            #A,C,G,T,..の数を各配列ごとにカウントした結果をhogeに格納
CG <- rowSums(hoge[,2:3])                   #C,Gの総数を計算してCGに格納
ACGT <- rowSums(hoge[,1:4])                 #A,C,G,Tの総数を計算してACGTに格納
GC_content <- CG/ACGT*100                   #%GC含量を計算してGC_contentに格納

#ファイルに保存
tmp <- cbind(names(fasta), CG, ACGT, width(fasta), GC_content)#保存したい情報をtmpに格納
colnames(tmp) <- c("description", "CG", "ACGT", "Length", "%GC_contents")#列名を付与
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.names=T)#tmpの中身を指定したファイル名で保存
```

cbind関数適用時に、出力したい順番に並べているだけです

hoge1.txt

	A	B	C	D	E
1	description	CG	ACGT	Length	%GC_contents
2	contig_1	16	24	24	66.67
3	contig_2	65	103	103	63.11
4	contig_3	33	65	65	50.77
5	contig_4	25	49	49	51.02

hoge1

GC含量

```
#ファイルに保存
tmp <- cbind(names(fasta), CG, ACGT, width(fasta), GC_content)#保存したい情報をtmpに格納
colnames(tmp) <- c("description", "CG", "ACGT", "Length", "%GC_contents")#列名を付与
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F, col.names=T)#tmpの中身を指定したファイル名で保存
```

R Console

```
> tmp
                CG   ACGT         GC_content
[1,] "contig_1" "16" "24"   "24"   "66.6666666666667"
[2,] "contig_2" "65" "103"  "103"  "63.1067961165049"
[3,] "contig_3" "33" "65"   "65"   "50.7692307692308"
[4,] "contig_4" "25" "49"   "49"   "51.0204081632653"
> colnames(tmp) <- c("description", "CG", "ACGT", "Length", "%GC_contents")#$
> tmp
     description CG   ACGT  Length %GC_contents
[1,] "contig_1"  "16" "24"  "24"   "66.6666666666667"
[2,] "contig_2"  "65" "103" "103"  "63.1067961165049"
[3,] "contig_3"  "33" "65"  "65"   "50.7692307692308"
[4,] "contig_4"  "25" "49"  "49"   "51.0204081632653"
>
```

列（column）の名前（names）を
適当に変更して出力しています

hoge1.txt

	A	B	C	D	E
1	description	CG	ACGT	Length	%GC_contents
2	contig_1	16	24	24	66.67
3	contig_2	65	103	103	63.11
4	contig_3	33	65	65	50.77
5	contig_4	25	49	49	51.02

hoge1

ヒトゲノム中のCpG出現確率は低い

- 全部で16通りの2連続塩基の出現頻度分布を調べると、CGとなる確率の実測値(0.986%)は期待値(4.2%)よりもかなり低い
- 期待値
  - ゲノム中のGC含量を考慮した場合：約41%(A:0.295, C:0.205, G: 0.205, T:0.295)なので、0.205×0.205= 4.2%
  - ゲノム中のGC含量を考慮しない場合：50%(A:0.25, C:0.25, G: 0.25, T:0.25)なので、0.25×0.25= 6.25%

2連続塩基の出現頻度：基本形

イントロ | 一般 | 2連続塩基の出現頻度情報を取得 NEW

multi-fasta形式ファイルを読み込んで、"AA", "AC", "AG", "AT", "CA", "CC", "CG", "CT", "GA", "GC", "GG", "GT", "TA", "TC", "TG", "TT"の計$4^2 = 16$通りの2連続塩基の出現頻度を調べるやり方を示します。
ヒトゲノムで"CG"の割合が期待値よりも低い(Lander et al., 2001; Saxonov et al., 2006)ですが、それを簡単に検証できます。
「ファイル」-「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

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

タイトル通りの出現頻度です。

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

#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

#本番
out <- dinucleotideFrequency(fasta)         #2連続塩基の出現頻度情報をoutに格

#ファイルに保存
tmp <- cbind(names(fasta), out)             #最初の列にID情報、そのあとに出現
write.table(tmp
```

hoge4.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

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

出力：hoge1.txt

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

2連続塩基の出現確率：基本形

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

出現頻度ではなく、出現確率を得るやり方です。

```
in_f <- "hoge4.fa"                          #入力ファイル名を指定してin_fに格納
out_f <- "hoge2.txt"                        #出力ファイル名を指定してout_fに格納

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

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

#本番
out <- dinucleotideFrequency(fasta, as.prob=T)#2連続塩基の出現確率情報を

#ファイルに保存
tmp <- cbind(names(fasta), out)             #最初の列にID情報、そのあとに出現
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)#tmp
```

hoge4.fa - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

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

出力：hoge2.txt

	AA	AC	AG	AT	CA	CC	CG	CT	GA	GC	GG	GT	TA	TC	TG	TT
contig_1	0.0%	4.3%	4.3%	8.7%	8.7%	8.7%	13.0%	8.7%	8.7%	8.7%	13.0%	0.0%	0.0%	13.0%	0.0%	0.0%
contig_2	3.9%	5.9%	8.8%	1.0%	10.8%	10.8%	4.9%	5.9%	3.9%	8.8%	9.8%	7.8%	1.0%	7.8%	5.9%	2.9%
contig_3	3.1%	6.3%	7.8%	6.3%	6.3%	3.1%	7.8%	3.1%	6.3%	4.7%	10.9%	9.4%	9.4%	6.3%	4.7%	4.7%
contig_4	6.3%	12.5%	4.2%	6.3%	10.4%	6.3%	6.3%	8.3%	6.3%	6.3%	2.1%	4.2%	6.3%	4.2%	8.3%	2.1%

2連続塩基の出現確率:ヒトゲノム

5. BSgenomeパッケージ中のヒトゲノム配列("BSgenome.Hsapiens.UCSC.hg19")の場合:

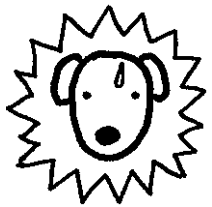
出現頻度ではなく、出現確率

```
out_f <- "hoge5.txt"
param <- "BSgenome.H

#必要なパッケージをロー
library(Biostrings)
library(param, chara

#前処理(paramで指定し
tmp <- ls(paste("pac
genome <- eval(parse
fasta <- getSeq(geno
names(fasta) <- seqn
fasta

#本番
out <- dinucleotideF

#ファイルに保存
tmp <- cbind(names(f
write.table(tmp, out
```

出力:hoge5.txt

	AA	AC	AG	AT	CA	CC	CG	CT	GA	GC	GG	GT	TA	TC	TG	TT
chr1	9.5%	5.0%	7.1%	7.5%	7.3%	5.4%	1.0%	7.1%	6.0%	4.4%	5.4%	5.0%	6.4%	6.0%	7.3%	9.5%
chr2	10.0%	5.0%	7.0%	7.9%	7.2%	5.0%	0.9%	7.0%	5.9%	4.1%	5.0%	5.0%	6.7%	5.9%	7.2%	10.0%
chr3	10.1%	5.0%	6.9%	8.0%	7.2%	4.9%	0.8%	6.9%	5.9%	4.0%	4.9%	5.0%	6.9%	5.9%	7.2%	10.2%
chr4	10.6%	5.0%	6.7%	8.5%	7.1%	4.5%	0.8%	6.7%	5.8%	3.8%	4.5%	5.0%	7.4%	5.8%	7.1%	10.6%
chr5	10.2%	5.0%	6.9%	8.1%	7.2%	4.8%	0.8%	6.9%	5.9%	4.0%	4.8%	5.0%	7.0%	5.9%	7.2%	10.2%
chr6	10.2%	5.0%	6.9%	8.1%	7.2%	4.9%	0.9%	6.9%	5.9%	4.0%	4.9%	5.0%	6.9%	5.9%	7.2%	10.2%
chr7	9.8%	5.0%	7.0%	7.8%	7.2%	5.2%	1.0%	7.0%	5.9%	4.3%	5.2%	5.0%	6.6%	5.9%	7.2%	9.9%
chr8	10.0%	5.1%	6.9%	7.9%	7.2%	5.0%	0.9%	6.9%	5.9%	4.1%	5.0%	5.0%	6.7%	5.9%	7.2%	10.0%
chr9	9.7%	5.0%	7.0%	7.6%	7.3%	5.3%	1.0%	7.0%	5.9%	4.3%	5.3%	5.0%	6.4%	5.9%	7.3%	9.7%
chr10	9.6%	5.0%	7.0%	7.5%	7.3%	5.4%	1.0%	7.1%	6.0%	4.4%	5.4%	5.1%	6.3%	6.0%	7.3%	9.6%
chr11	9.5%	5.0%	7.1%	7.6%	7.3%	5.4%	1.0%	7.1%	6.0%	4.4%	5.4					9.5%
chr12	9.8%	5.0%	7.0%	7.7%	7.2%	5.2%	1.0%	7.0%	5.9%	4.2%	5.2					9.8%
chr13	10.5%	5.0%	6.7%	8.5%	7.1%	4.6%	0.8%	6.7%	5.8%	3.8%	4.6%	5.0%	7.3%	5.8%	7.1%	10.5%
chr14	9.7%	5.0%	7.0%	7.7%	7.2%	5.2%	1.0%	7.0%	5.9%	4.3%	5.2%	5.1%	6.6%	5.9%	7.3%	9.9%
chr15	9.4%	5.1%	7.1%	7.3%	7.3%	5.5%	1.1%	7.2%	6.0%	4.5%	5.5%	5.1%	6.2%	6.0%	7.3%	9.4%
chr16	8.6%	5.1%	7.3%	6.7%	7.5%	6.2%	1.4%	7.3%	6.1%	5.0%	6.2%	5.1%	5.4%	6.1%	7.6%	8.6%
chr17	8.4%	5.0%	7.4%	6.4%	7.4%	6.5%	1.5%	7.4%	6.0%	5.2%	6.5%	5.0%	5.3%	6.1%	7.4%	8.5%
chr18	10.1%	5.0%	6.9%	8.1%	7.2%	4.9%	0.9%	6.9%	5.9%	4.1%	4.9%	5.1%	6.9%	5.9%	7.2%	10.1%
chr19	7.6%	5.1%	7.4%	5.7%	7.6%	7.2%	1.9%	7.4%	6.1%	5.7%	7.3%	5.1%	4.5%	6.1%	7.6%	7.6%
chr20	8.7%	5.0%	7.3%	6.8%	7.5%	6.0%	1.2%	7.3%	6.0%	4.9%	6.1%	5.1%	5.6%	6.1%	7.6%	8.9%
chr21	9.9%	5.1%	6.9%	7.8%	7.3%	5.2%	1.1%	6.8%	5.9%	4.3%	5.2%	5.1%	6.6%	5.9%	7.3%	9.8%
chr22	7.6%	5.1%	7.5%	5.8%	7.7%	7.1%	1.7%	7.5%	6.1%	5.7%	7.1%	5.1%	4.6%	6.1%	7.7%	7.6%
chrX	10.1%	5.0%	6.8%	8.2%	7.2%	4.9%	0.8%	6.8%	5.9%	3.9%	4.9%	5.1%	7.0%	5.9%	7.2%	10.2%
chrY	9.9%	5.1%	6.8%	8.1%	7.3%	4.9%	0.8%	6.8%	6.0%	3.9%	4.9%	5.2%	6.7%	6.0%	7.5%	10.0%

2分強かかります



- イントロ | 一般 | 2連続塩基の 出現頻度情報を取得(last modified 2014/02/07)
- 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 2014/04/10) NEW
- イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome(last modified 2014/04/01) 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) NEW
- イントロ | 一般 | 配列取得 | プロモーター配列 | BSgenome(last modified 2013/10/10)
- イントロ | 一般 | 配列取得 | プロモーター配
- イントロ | 一般 | 配列取得 | トランスクリプ
- イントロ | 一般 | 配列取得 | トランスクリプ
- イントロ | NGS | 様々なプラットフォーム(l
- イントロ | NGS | qPCRやmicroarrayなどと
- イントロ | NGS | Viewer(last modified 201
- イントロ | NGS | 配列取得 | FASTQ or SR
- イントロ | NGS | 配列取得 | FASTQ or SR
- イントロ | NGS | アノテーション情報取得
- イントロ | NGS | アノテーション情報取得
- イントロ | NGS | アノテーション情報取得
- イントロ | NGS | アノテーション情報取得
- イントロ | NGS | アノテーション情報取得
- イントロ | NGS | アノテーション情報取得
- イントロ | NGS | アノテーション情報取得
- イントロ | NGS | アノテーション情報取得
- イントロ | NGS | 読み込み | FASTA形式
- イントロ | NGS | 読み込み | FASTA形式
- イントロ | NGS | 読み込み | FASTQ形式(

- イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome

イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome

BSgenomeパッケージを用いて様々な生物種のゲノム配列を取得するやり方を示します。ミヤマハタザオ (A. lyrata)、セイヨウミツバチ (A. mellifera)、シロイヌナズナ(A.thaliana)、ウシ(B.taurus)、線虫(C.elegans)、犬(C.familiaris)、キイロショウジョウバエ(D.melanogaster)、ゼブラフィッシュ(D.rerio)、大腸菌(E.coli)、イトヨ(G.aculeatus)、セキショクヤケイ(G.gallus)、ヒト(H.sapiens)、アカゲザル(M.mulatta)、マウス(M.musculus)、チンパンジー(P.troglodytes)、ラット(R.norvegicus)、出芽酵母(S.cerevisiae)、トキソプラズマ(T.gondii)と実に様々な生物種が利用可能であることがわかります。

getSeq関数はBSgenomeオブジェクト中の「single sequences」というあたりにリストアップされているchr...というものを全て抽出しています。したがって、例えばマウスゲノムは「chr1」以外に「chr1_random」や「chrUn_random」なども等価に取扱っている点に注意してください。

「ファイル」-「ディレクトリの変更」でファイルを保存したいディレクトリに移動し以下をコピペ。

1. 利用可能な生物種とRにインストール済みの生物種をリストアップしたい場合:

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

#本番（利用可能なリストアップ；インストール済みとは限らない）
available.genomes()                        #このパッケージ中で利用可能なゲノムをリストアップ

#本番（インストール済みの生物種をリストアップ）
installed.genomes()                        #インストール済みの生物種をリストアップ

#後処理（パッケージ名でだいたいわかるがproviderやversionを分割して表示したい場合）
installed.genomes(splitNameParts=TRUE) #インストール済みの生物種をリストアップ
```

様々な生物種のゲノム配列がRのパッケージとして提供されています。installed.genomes関数のところで2分程度かかります

イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome

BSgenomeパッケージを用いて様々な生物種のゲノム配列を取得するやり方を示します。ミヤマハタザオ（A. lyrata）、セイヨウ
ミツバチ（A. mellifera）、シロイヌナズナ（A.thaliana）、
バエ（D.melanogaster）、ゼブラフィッシュ（D.rerio）、
（H.sapiens）、アカゲザル（M.mulatta）、マウス（M.mu
（S.cerevisiae）、トキソプラズマ（T.gondii）と実に様々
getSeq関数はBSgenomeオブジェクト中の「single sequ
ています。したがって、例えばマウスゲノムは「chr1」
注意してください。
「ファイル」−「ディレクトリの変更」でファイルを保存し

1. 利用可能な生物種とRにインストール済みの生物

```
#必要なパッケージをロード
library(BSgenome)

#本番（利用可能なリストアップ；インストー
available.genomes()

#本番（インストール済みの生物種をリストア
installed.genomes()

#後処理（パッケージ名でだいたいわかるがpr
installed.genomes(splitNameParts=TRUE)
```

R Console

```
[24] "BSgenome.Hsapiens.UCSC.hg18"
[25] "BSgenome.Hsapiens.UCSC.hg19"
[26] "BSgenome.Mmulatta.UCSC.rheMac2"
[27] "BSgenome.Mmulatta.UCSC.rheMac3"
[28] "BSgenome.Mmusculus.UCSC.mm10"
[29] "BSgenome.Mmusculus.UCSC.mm8"
[30] "BSgenome.Mmusculus.UCSC.mm9"
[31] "BSgenome.Osativa.MSU.MSU7"
[32] "BSgenome.Ptroglodytes.UCSC.panTro2"
[33] "BSgenome.Ptroglodytes.UCSC.panTro3"
[34] "BSgenome.Rnorvegicus.UCSC.rn4"
[35] "BSgenome.Rnorvegicus.UCSC.rn5"
[36] "BSgenome.Scer
[37] "BSgenome.Scer
[38] "BSgenome.Scer
[39] "BSgenome.Tgon
>
> #本番(インストール済みの生物種をリストアッ$
> installed.genomes()                    #イ$
[1] "BSgenome.Celegans.UCSC.ce2"
[2] "BSgenome.Drerio.UCSC.danRer7"
[3] "BSgenome.Ecoli.NCBI.20080805"
[4] "BSgenome.Hsapiens.UCSC.hg19"
[5] "BSgenome.Hsapiens.UCSC.hg19.Rbowtie"
[6] "BSgenome.Mmusculus.UCSC.mm9"
[7] "BSgenome.Scerevisiae.UCSC.sacCer2"
>
```

ヒトゲノム(BSgenome.Hsapiens.UCSC.hg19)の2連続塩基出現頻度計算ができたのは、このパッケージをインストール済みだからです

イントロ | 一般 | 配列取得 | ゲノム配列 | BSgenome

BSgenomeパッケージを用いて様々な生物種のゲノム配列を取得するやり方を示します。ミヤマハタザオ (A. lyrata)、セイヨウミツバチ (A. mellifera)、シロイヌナズナ(A.thaliana)、ウシ(B.taurus)、線虫(C.elegans)、犬(C.familiaris)、キイロショウジョウバエ(D.melanogaster)、ゼブラフィッシュ(D.rerio)、大腸菌(E.coli)、イトヨ(G.aculeatus)、セキショクヤケイ(G.gallus)、ヒト(H.sapiens)、アカゲザル(M.mulatta)、マウス(M.musculus)、チンパンジー(P.troglodytes)、ラット(R.norvegicus)、出芽酵母(S.cerevisiae)、トキソプラズマ(T.gondii)と実に様々な生物種が利用可能であることがわかります。

getSeq関数はBSgenomeオブジェクト中の「single sequen

ています。したがって、例えばマウスゲノムは「chr1」以

注意してください。

「ファイル」-「ディレクトリの変更」でファイルを保存した

1. 利用可能な生物種とRにインストール済みの生物種

```
#必要なパッケージをロード
library(BSgenome)

#本番（利用可能なリストアップ；インストール
available.genomes()

#本番（インストール済みの生物種をリストアッ
installed.genomes()

#後処理（パッケージ名でだいたいわかるがprov
installed.genomes(splitNameParts=TRUE)
```

R Console

```
[24] "BSgenome.Hsapiens.UCSC.hg18"
[25] "BSgenome.Hsapiens.UCSC.hg19"
[26] "BSgenome.Mmulatta.UCSC.rheMac2"
[27] "BSgenome.Mmulatta.UCSC.rheMac3"
[28] "BSgenome.Mmusculus.UCSC.mm10"
[29] "BSgenome.Mmusculus.UCSC.mm8"
[30] "BSgenome.Mmusculus.UCSC.mm9"
[31] "BSgenome.Osativa.MSU.MSU7"
[32] "BSgenome.Ptroglodytes.UCSC.panTro2"
[33] "BSgenome.Ptroglodytes.UCSC.panTro3"
[34] "BSgenome.Rnorvegicus.UCSC.rn4"
[35] "BSgenome.Rnorvegicus.UCSC.rn5"
[36] "BSgenome.Scerevisiae.UCS
[37] "BSgenome.Scerevisiae.UCS
[38] "BSgenome.Scerevisiae.UCS
[39] "BSgenome.Tgondii.ToxoDB.
>
>#本番(インストール済みの生物種をリストアッ$
> installed.genomes()                    #イ$
[1] "BSgenome.Celegans.UCSC.ce2"
[2] "BSgenome.Drerio.UCSC.danRer7"
[3] "BSgenome.Ecoli.NCBI.20080805"
[4] "BSgenome.Hsapiens.UCSC.hg19"
[5] "BSgenome.Hsapiens.UCSC.hg19.Rbowtie"
[6] "BSgenome.Mmusculus.UCSC.mm9"
[7] "BSgenome.Scerevisiae.UCSC.sacCer2"
>
```

もしゼブラフィッシュゲノム
(BSgenome.Drerio.UCSC.danRer7)
がインストールされていなければ...

2. ゼブラフィッシュ("BSgenome.Drerio.UCSC.danRer7")のゲノム情報をRにインストールしたい場合:

400MB程度あります...

```
param <- "BSgenome.Drerio.UCSC.danRer7"#パッケージ名を指定

#本番
source("http://bioconductor.org/biocLite.R")#おまじない
biocLite(param)                          #おまじない

#後処理（インストール済みの生物種をリストアップ）
installed.genomes()                      #インストール済みの生物種をリストアップ
```

ここを参考にして **available.genomes()** でリストアップされていない任意のパッケージ名を指定

3. インストール済みのゼブラフィッシュのゲノム配列をmulti-fastaファイルで保存したい場合:

1.4GB程度のファイルが生成されます...

```
out_f <- "hoge3.txt"                     #出力ファイル名を指定してout_fに格納
param <- "BSgenome.Drerio.UCSC.danRer7"#パッケージ名を指定

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

#前処理(paramで指定したパッケージ中のオブジェクト名をgenomeに統一)
#tmp <- unlist(strsplit(param, ".", fixed=TRUE))[2]#paramで指定した文字列からオブジェクト名を取得
tmp <- ls(paste("package", param, sep=":"))#paramで指定したパッケージで利用可能なオブジェクト名を
genome <- eval(parse(text=tmp))          #文字列tmpをRオブジェクトとしてgenomeに格納(パッケージ中に
genome                                   #確認してるだけです

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

#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
```

available.genomes() でリストアップされているパッケージ名を指定可能です

multi-FASTAファイルとして保存したい場合

5. インストール済みのヒト("BSgenome.Hsapiens.UCSC.hg19")のゲノム配列をmulti-fastaファイルで保存したい場合:

3.0GB程度のファイルが生成されます…。ヒトゲノムは、まだ完全に22本の常染色体とX, Y染色体の計24本になっているわけではないことがわかります。

```
out_f <- "hoge5.txt"                          #出力ファイル名を指定してout_fに格納
param <- "BSgenome.Hsapiens.UCSC.hg19" #パッケージ名を指定

#必要なパッケ
library(para                                  paramで指定したパッケージの読み込み

#前処理(paran                                 ェクト名をgenomeに統一)
#tmp <- unlist(strsplit(par
tmp <- ls(paste("package",
genome <- eval(parse(text=t
genome

#本番
fasta <- getSeq(genome)
names(fasta) <- seqnames(ge

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

8分程度かかります。
実行はやめましょう。

フリーズするので、得られたhoge5.txtを
テキストエディタで開くことはやめましょう

R Console

```
|   chrUn_gl000246
|   chrUn_gl000247
|   chrUn_gl000248
|   chrUn_gl000249
|
| multiple sequences (see '?mseqnames'):
|   upstream1000  upstream2000  upstream5000
|
| (use the '$' or '[[' operator to access a
| given sequence)
>
> #本番
> fasta <- getSeq(genome)                    #ゲノム塩基配列情報を抽出し$
> names(fasta) <- seqnames(genome)           #description情報を追加して$
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=50)#fasta$
>
```

①　②　③　④　⑤

内部的には、
①writeXStringSet関数を用いて、
②**fasta**オブジェクトを、
③out_fで指定したファイル名で、
④FASTA形式で、
⑤1行あたりの文字数を50文字
にして保存しています。