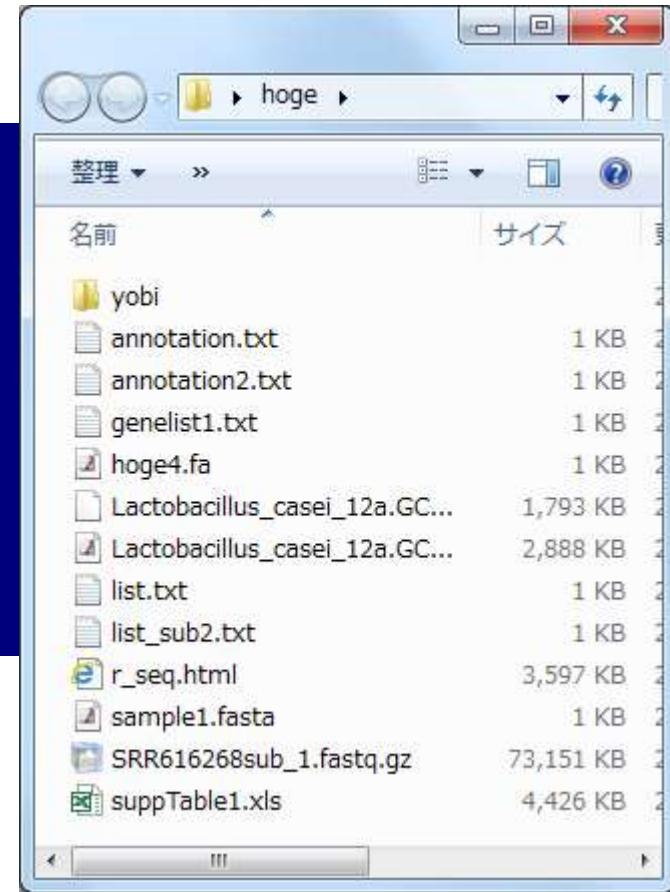


配布するUSBメモリ中のhogeフォルダを
デスクトップにコピーしておいてください。

NGSハンズオン 講習会 : R基礎

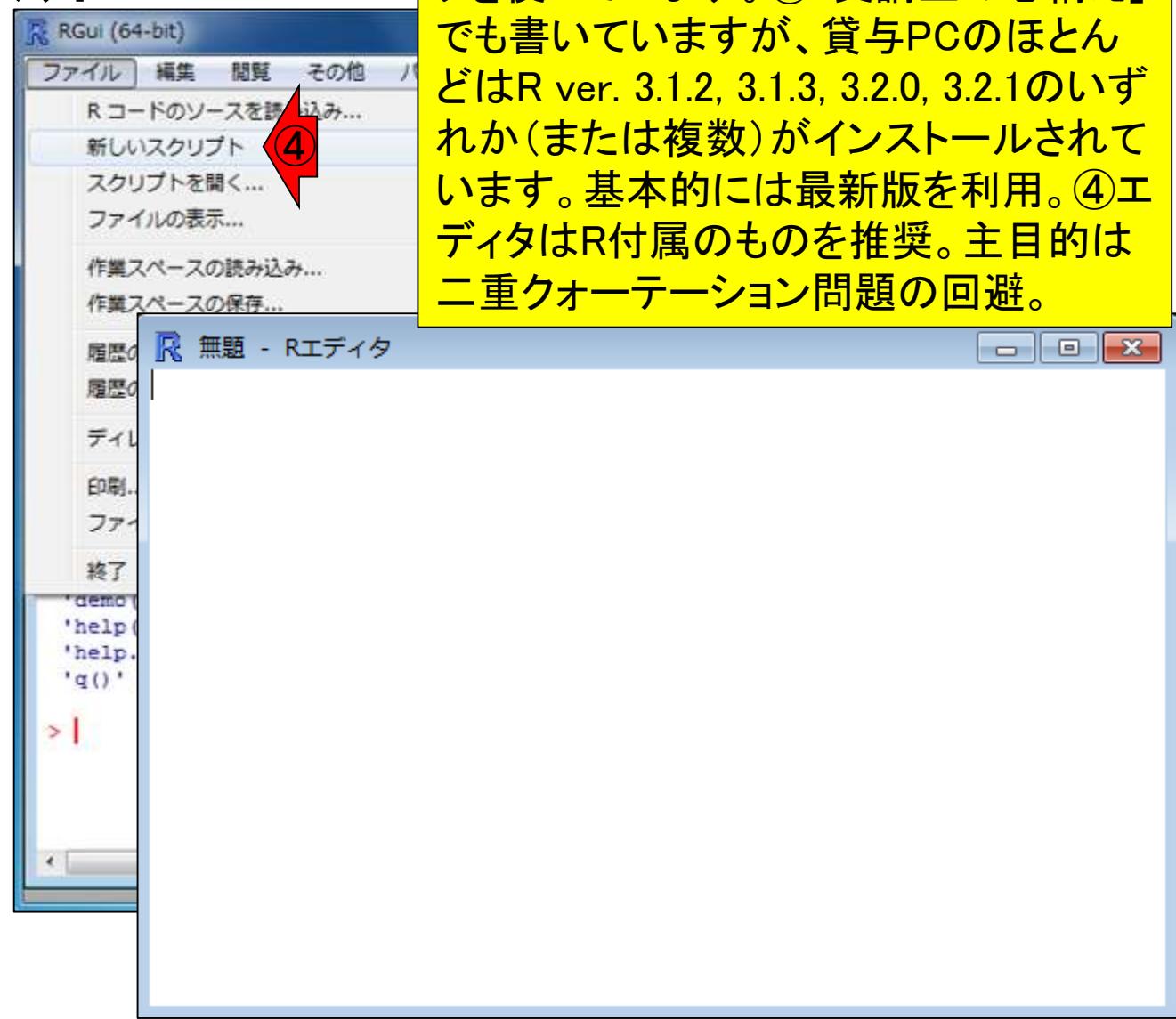
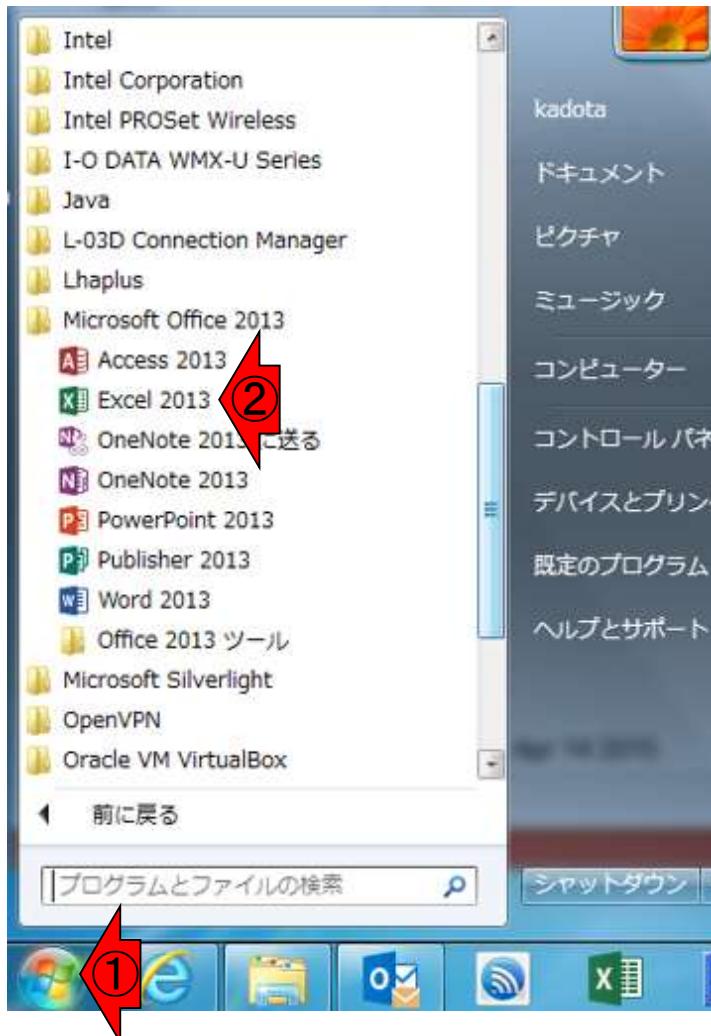
東京大学・大学院農学生命科学研究所
アグリバイオインフォマティクス教育研究プログラム
門田幸二(かどた こうじ)
kadota@iu.a.u-tokyo.ac.jp
<http://www.iu.a.u-tokyo.ac.jp/~kadota/>



Contents(全体)

- 7月22日(水):84→**83名**。Bio-Linux 8とRのインストール状況確認。基本自習(門田・寺田先生)
- 7月23日(木):92→**90名**。Linux基礎。LinuxコマンドなどUNIXの基礎の理解(門田)
- 7月24日(金):85→**83名**。スクリプト言語。シェルスクリプト(アメリカエフ株式会社 服部恵美先生)
- 7月27日(月):93→**91名**。スクリプト言語。Perl(アメリカエフ 服部先生)
- 7月28日(火):91→**90名**。スクリプト言語。Python(アメリカエフ 服部先生)
- 7月29日(水):94→**88名**。データ解析環境R(門田)
 - R基礎1(初級):R言語の基礎(インストールから利用まで)
 - R基礎2(初級):ファイルの読み込み、行列演算の基本
 - R各種パッケージ(中級):パッケージのインストール法と代表的なパッケージの利用法
- 7月30日(木):96→**91名**。データ解析環境R(門田)
 - Bioconductorの利用法1(中級):データの型やバージョンの違い
 - Bioconductorの利用法2(中級):FASTA/FASTQファイルの各種解析
- 8月3日(月):89→**84名**。NGS解析。基礎(アメリカエフ 山口昌雄先生)
- 8月4日(火):85→**80名**。NGS解析。ゲノムReseq、変異解析(アメリカエフ 山口先生)
- 8月5日(水):86 →**81名**。NGS解析。RNA-seq、統計解析(前半:山口先生、後半:門田)
- 8月6日(木):104 →**98名**。NGS解析。ChIP-seq(理研 森岡勝樹先生)

各種ソフトの場所



①②Excelは行列データファイルの確認用。門田はEmEditorというテキストエディタを使っています。③「受講生の心構え」でも書いていますが、貸与PCのほとんどはR ver. 3.1.2, 3.1.3, 3.2.0, 3.2.1のいずれか(または複数)がインストールされています。基本的には最新版を利用。④エディタはR付属のものを推奨。主目的は二重クオーテーション問題の回避。

Contents

■ R基礎(初級)

- おさらい
- コード内部の説明(ファイルの読み込み、行列演算の基礎)
- リアルRNA-seqカウントデータ(数値行列データ)

■ R各種パッケージ(中級): 代表的なパッケージの利用法

- (パッケージのインストール法)
- 基本情報取得(コンティグ数、配列長、N50、GC含量)
- 任意の領域の切り出し
- GC含量計算部分の説明

おさらい

(Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランскриプトーム、正規化、発現変動、統計、モデル、バイオインフォマティクス

(last modified 2015/07/08, since 2010)

hogeフォルダ中のr_seq.htmlをダブルクリックしてローカルで利用するのがいいかもしれません。ここで示すようなクリックして眺めるだけのネットサーフィン系の部分は、手を動かさずに前のスライドを見ているだけのほうがいいかもしれません。

What's new?

- このウェブページ 従ってフリーソフト 基本的な利用法 的にまとめた書籍
- 「解析」発現変動
- 参考資料(講義)
- R ver 3.2.0(Gen

- 基本的な利用法 (last modified 2015/04/03)
- サンプルデータ (last modified 2015/06/15) NEW
- バイオインフォマティクス人材育成カリキュラム(次世代シークエンサ) | NGSハンズオン講習会 (1)
- 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2014 (last modified 2014/04/15)
- 書籍 | トランскриプトーム解析 | について (last modified 2014/05/12)
- 書籍 | トランскриプトーム解析 | 2.3.1 RNA-seqデータ/FASTQファイル (last modified 2014/04/15)
- 書籍 | トランскриプトーム解析 | 2.3.2 リ
- 書籍 | トランскриプトーム解析 | 2.3.3 ア
- 書籍 | トランскриプトーム解析 | 2.3.4 マ
- 書籍 | トランскриプトーム解析 | 2.3.5 マ
- 書籍 | トランскриプトーム解析 | 2.3.6 カ
- 書籍 | トランскриプトーム解析 | 3.3.1 解
- 書籍 | トランскриプトーム解析 | 3.3.2 デ
- 書籍 | トランскриプトーム解析 | 3.3.3 ク
- 書籍 | レニン、ラクリー、レーナー解説 | 2.3.4 文

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) | NGSハンズオン講習会2015 NEW

NGSハンズオン講習会を2015年7月22日-8月6日の11日間で実施します。予定通り受講申込多数のため、予備日(8月26日、27日、28日)も実施することになっています。

はじめに(全員目を通しておきましょう)

- 講習会期間中アグリバイオノートPCを借りるヒトは、7/22の動作確認作業自体は行う必要はありません。しかし、7/22のところに列挙した項目の予習は必須です。「エアーハンズオン」をするなりしてチェックリスト項目をクリアしておきましょう。
- 平成26年度開催のNGS速習コース関係
 - 報告書PDF(h26_ngs_report.pdf; 約4MB)
概要、スケジュール、アンケート結果、受講生のコメントなどを見られます。
 - 報告プレゼン資料(20150126_kadota.pdf; 約1MB)
報告書PDFの短縮版のこのようなものです。Twitterやっているヒトは、ハッシュタグ #AJACS をつけて平成27年度も有効利用してください。
- 平成27年度開催のNGSハンズオン講習会関係
 - 前座プレゼン資料(20150722_kadota.pdf; 2014.07.14版; 約1MB)
概要、注意事項、受講生の心構えなどをざっくりと書いてあります。

おさらい

バイオインフォマティクス人材育成カリキュラム(次世代シークエンサ) | NGSハンズオン講習会2015 NEW

NGSハンズオン講習会を2015年7月22日-8月6日の11日間で実施します。予定通り受講申込多数のため、予備日(8月26日、27日、28日)も実施することになっています。

はじめに(全)

- ・ 講習
基本
ズオ
- ・ 平成

2015年7月22日(2015.07.22):Bio-Linux 8とRのインストール状況確認(門田幸二、寺田透)

書籍「日本乳酸菌学会誌」について示した通りのPC環境を構築しておきましょう。連載第1-3回、および第4回のウェブ資料W6-5までの予習は必須です。Rについても同様です。一週間程度はきっちり時間をかけて予習しておきましょう。7/22は、以下に示すようなことができる(わかる)ようになっていることの確認を自分でしてもらう日です。門田自身全てを完全に把握しているわけではありませんし、ウェブ資料のページ数も膨大ですので、どこにどういうことが書かれていたかの全体像の俯瞰やチェックリストという位置づけであります(頻繁に更新しているのでときどきリロードしましょう):

- Bio-Linux 8

• R

1. インストールについてをよく読み、ここに書いてある手順に従って2015年4月4日以降にインストールを行った
2. インストールについてで書いてある内容はBio-Linux8(ゲストOS)とは無関係であり、WindowsやMacintosh(つまりホストOS)上で行う作業である
3. ファイルの拡張子(.txtや.docxなど)はちゃんと表示されている
4. Rの起動と終了ができる。終了時に表示されるメッセージにこうろたえない
5. R本体だけでなくRパッケージ群のインストールもちゃんと行った
6. Rパッケージ群のインストール確認も行い、エラーが出ないことを確認した
7. library関数を用いたRパッケージのロード中に、別のパッケージがないことに起因するエラーメッセージが出ることもあるが、必要なパッケージを個別にインストールするやり方を知っている
8. 基本的な利用法をよく読み、予習を行った
9. 作業ディレクトリの変更ができる
10. 例題ファイルのダウンロード時に、拡張子が勝手に変わることがあるので注意する
11. 慣れない場合は、getwd()とlist.files()を打ち込むことで、作業ディレクトリと入力ファイルの存在確認を行う
12. エラーに遭遇した際、「ありがちなミス1-4」に当てはまっているかどうか自分で確認

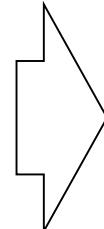


解析基礎2

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

入力1:アノテーションファイル(annotation.txt)

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



出力:hoge1.txt

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

入力2:リストファイル(genelist1.txt)

	A
1	gene1
2	gene7
3	gene9

解析基礎2

- ・ イントロ | 一般 | ランダムに行を抽出 (last modified 2014/07/17)
- ・ イントロ | 一般 | 任意の文字列を行の最初に挿入 (last modified 2014/07/17)
- ・ イントロ | 一般 | 任意のキーワードを含む行を抽出(基礎) (last modified 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 2015/02/19)
- ・ イントロ | 一般 | 任意の位置の塩基を置換 (last modified 2014/06/16)
- ・ イントロ | 一般 | 指定した範囲の配列を取得 (last modified 2014/06/16)
- ・ イントロ | 一般 | 指定したID(染色体やdesc) (last modified 2014/06/16)
- ・ イントロ | 一般 | 翻訳配列(translate)を取得 (last modified 2014/06/16)
- ・ イントロ | 一般 | 翻訳配列(translate)を取得 (last modified 2014/06/16)

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

イントロ | 一般 | 任意のキーワードを含む行を抽出(基礎)

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

1. 目的のタブ区切りテキストファイル(annotation.txt)中の第1列目をキーとして、リストファイル(geneList1.txt)中のものが含まれる行全体を出力したい場合:

```
in_f1 <- "annotation.txt"          #入力ファイル名を指定してin_f1に格納(アノテーション)
in_f2 <- "geneList1.txt"            #入力ファイル名を指定してin_f2に格納(リストファイル)
out_f <- "hoge1.txt"                #出力ファイル名を指定してout_fに格納
param <- 1                          #アノテーションファイル中の検索したい列番号を指定

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

#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
out <- data[obj,]                      #objがTRUEとなる行のみ抽出した結果をoutに格納
dim(out)                            #オブジェクトoutの行数と列数を表示

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



解析基礎2

イントロ | 一般 | 任意のキーワードを含む行を抽出(基礎)

作業ディレクトリは「デスクトップ - hoge」。
hogeフォルダ中にannotation.txtと
genelist1.txtが存在するという前提。貸与
PCは黒矢印部分が「kadota」ではなく「iu」。

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。

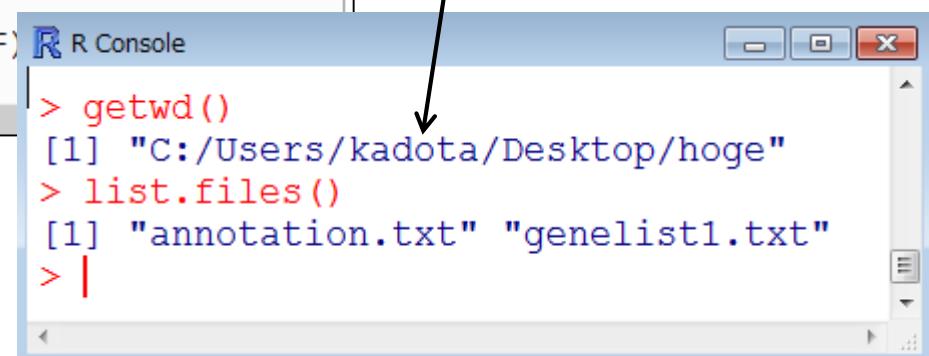
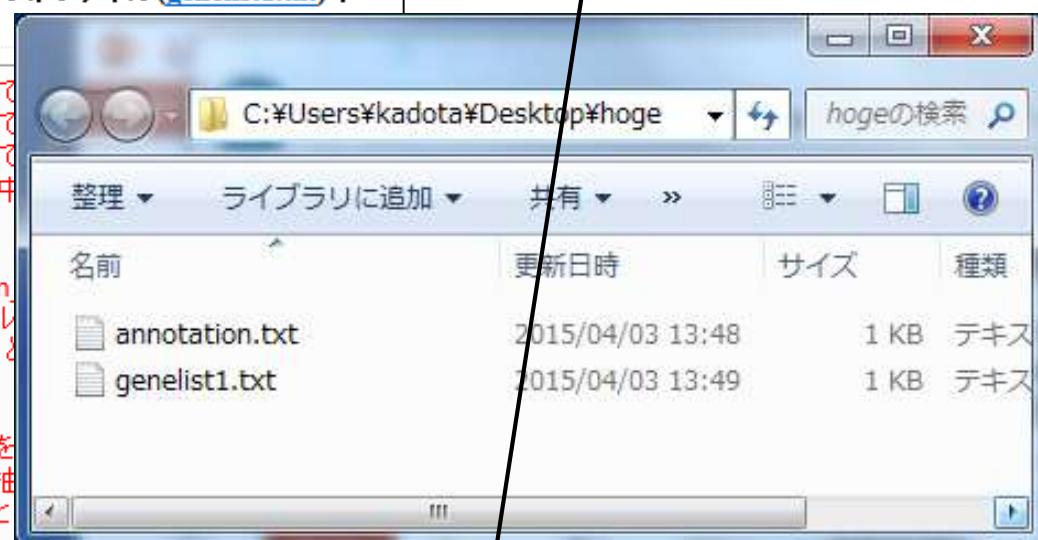
- 目的のタブ区切りテキストファイル(annotation.txt)中の第1列目をキーとして、リストファイル(genelist1.txt)中のものが含まれる行全体を出力したい場合:

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

#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
keywords <- readLines(in_f2)        #in_f2で指定したファイル
dim(data)                           #オブジェクトdataの行数を取得

#本番
obj <- is.element(as.character(data[,param]), keywords)#条件を
out <- data[obj,]                  #objがTRUEとなる行のみ抽出
dim(out)                            #オブジェクトoutの行数を取得

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



基本はコピペ

イントロ | 一般 | 任意のキーワードを含む行を抽出(基礎)

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。ブラウザがInternet Explorerの場合は、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。

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。

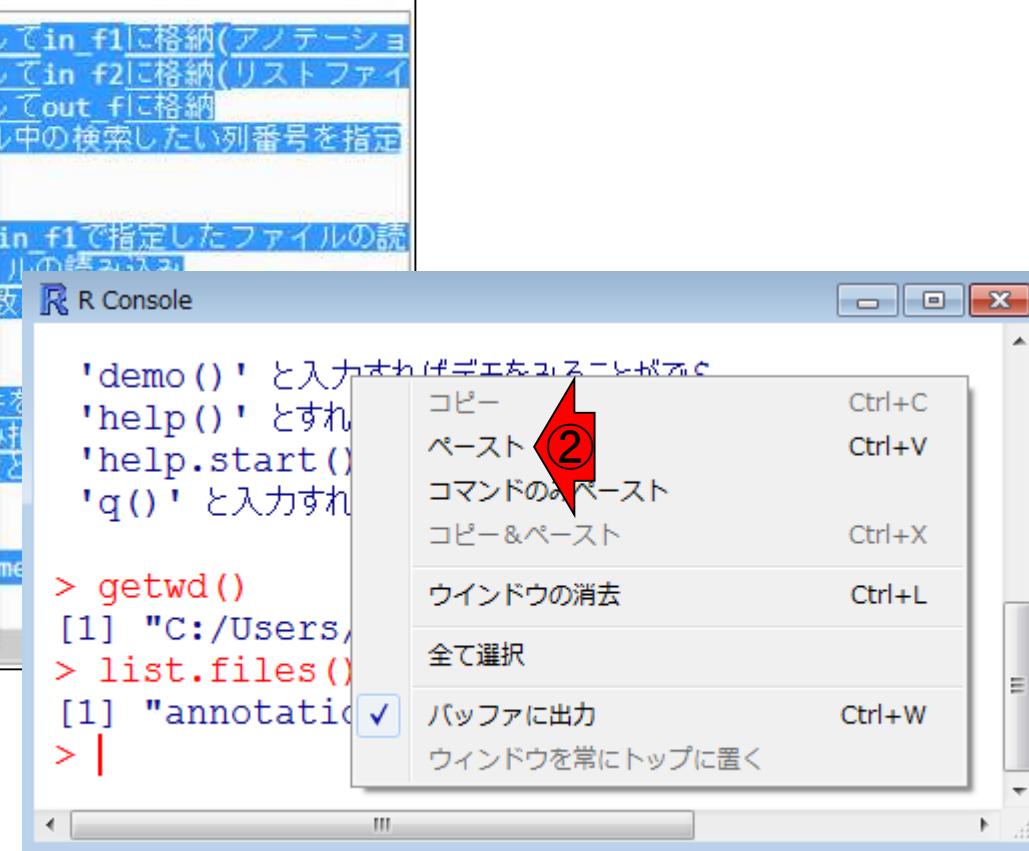
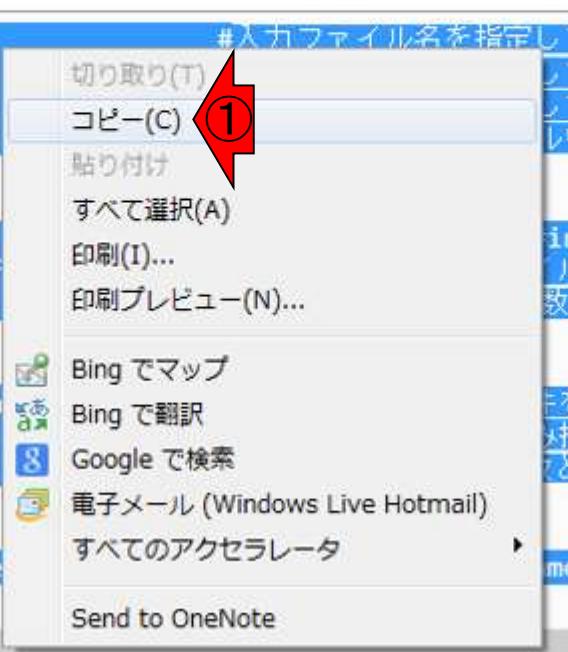
1. 目的のタブ区切りテキストファイル(annotation.txt)中の第1列目をキーとして、リストファイル(genelist1.txt)中のものが含まれる行全体を出力したい場合:

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

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

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

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



実行結果

R Console

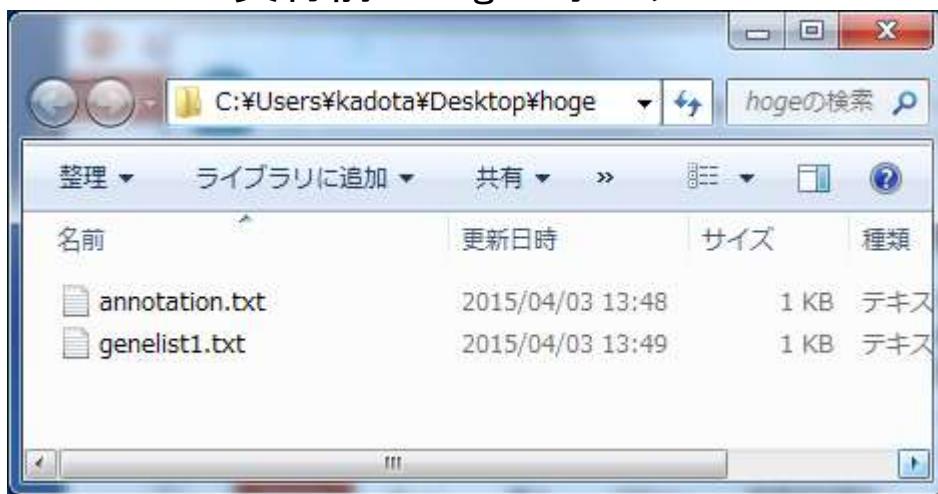
```

> #本番
> obj <- is.element(as.character(data[,pa$)
> out <- data[obj,]
> dim(out)
[1] 3 4
>
> #ファイルに保存
> write.table(out, out_f, sep="\t", appen$
> list.files()
[1] "annotation.txt" "genelist1.txt"
[3] "hoge1.txt"
> |
①

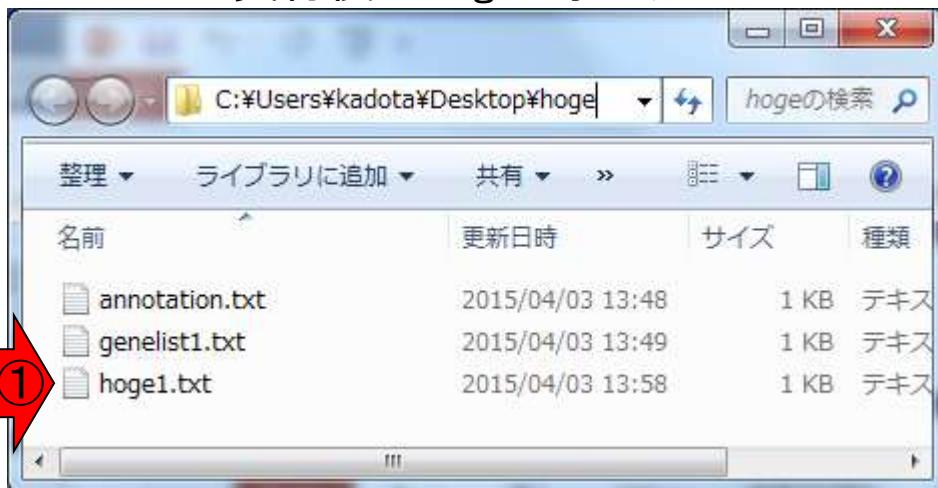
```

①出力ファイル名として指定したhoge1.txtが生成されているのがわかる。「list.files()で表示される結果」と「実行後のhogeフォルダの中身」は当然同じです

実行前のhogeフォルダ



実行後のhogeフォルダ



実行結果

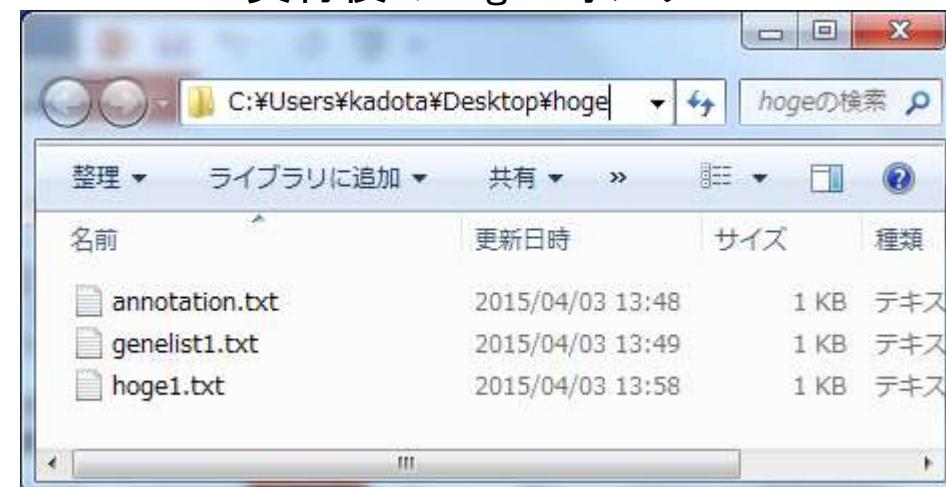
①outというオブジェクトの中身をwrite.tableという関数でファイルに出力しています。この場合、出力ファイルhoge1.txtの中身は、Rコンソール画面中でoutと打ち込むことで見られる。

R Console

```
> #ファイルに保存 ①
> write.table(out, out_f, sep="\t", append=F, quote=F$)
> list.files()
[1] "annotation.txt" "genelist1.txt" "hoge1.txt"
> out
  genename accession description subcellular_location
1   gene1     hoge01  plasma_mem          nuclear
7   gene7     hoge07  tebasaki          nuclear
9   gene9     hoge09  nihonshu          nuclear
> |
```

	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

実行後のhogeフォルダ



色の説明

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

(Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランск립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 2015/04/03, since 2010)

What's new?

- このウェブページは[インストール](#)についての推奨手順(Windows2015.04.01版とMacintosh2015.04.02版)に従ってフリーソフトRと必要なパッケージをインストール済みであるという前提で記述しています。初心者の方は[基本的な利用法](#)(Windows2015.04.03版とMacintosh2015.04.03版)で自習してください。本ウェブページを体系的にまとめた書籍もあります。(2015/04/03) 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27年度もバイオインフォ関連講義を行います。例年東大以外の企業の方、研究員、学生が2-3割程度受講しております。受講ガイダンスは4月6日17:15- 於東大農です。(2015/03/31) NEW
- R本体およびパッケージのインストール手順のところを更新しました。詳細は[インストール](#)についてをごらんください。(2015/04/02) NEW
- [MBCluster.Seq](#)パッケージを用いた遺伝子間クラスタリングのやり方を一通り示しました。(2015/03/14) NEW
- 参考資料(講義、講習会、本など)の項目を更新しました。(2015/03/09) NEW

はじめに (last modified 2015/03/31) NEW
 参考資料(講義、講習会、本など) (last modified 2015/03/09) NEW
 過去のお知らせ (last modified 2015/03/31) NEW
 インストール

インストール | R本体 | 最新版 | Win用 (last modified 2015/03/22) 推奨 NEW
 インストール | R本体 | 最新版 | Mac用 (last modified 2015/04/01) 推奨 NEW
 インストール | R本体 | 過去版 | Win用 (last modified 2015/03/22) NEW
 インストール | R本体 | 過去版 | Mac用 (last modified 2015/03/22) NEW
 インストール | Rパッケージ | ほぼ全て(20GB以上?)! (last modified 2015/03/22) NEW
 インストール | Rパッケージ | 必要最小限プラスアルファ(数GB?)! (last modified 2015/03/27) 推奨 トップページへ
 インストール | Rパッケージ | 必要最小限(数GB?)! (last modified 2015/03/23) NEW

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

応用

このサンプルコードは1列目でキーワード検索する場合。別のリストファイルを読み込んで4列目で検索したい場合のやり方を示します。

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)中の中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="")
keywords <- readLines(in_f2)        #in_f2で指定したファイル
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) #条件式: フィルタリング用の条件式
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

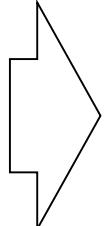
```

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

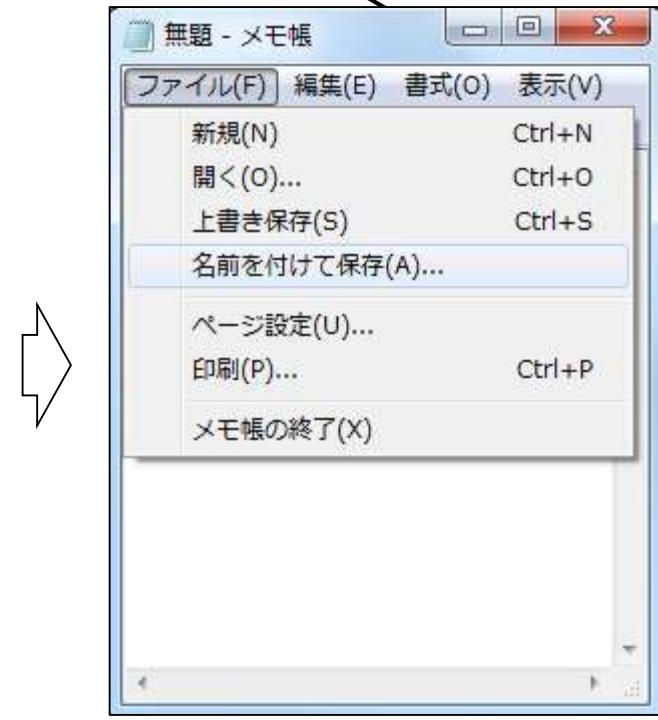
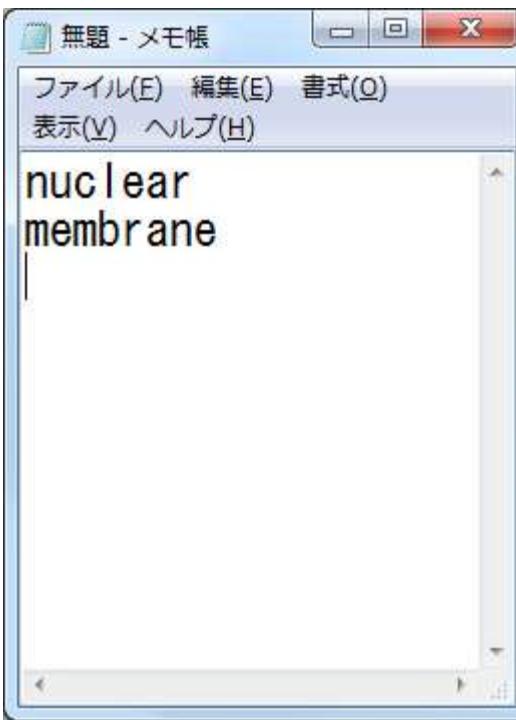
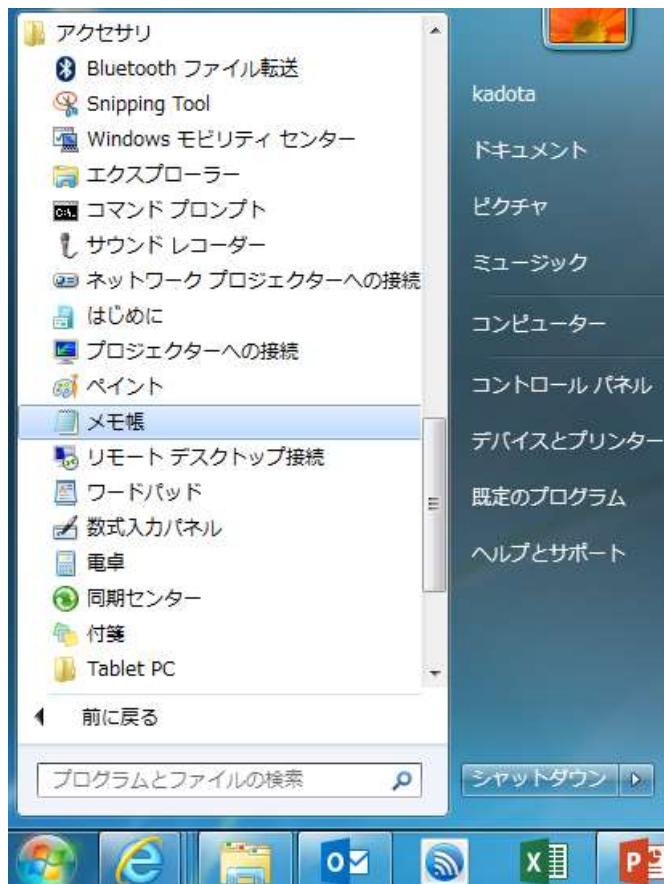
	A	B	C	D
1	genename	accession	description	subcellular_location
2	gene1	hoge01	plasma_mem	nuclear
3	gene2	hoge02	hohinu	membrane
4	gene3	hoge03	agribio	endoplasmic
5	gene4	hoge04	genesis	endoplasmic
6	gene5	hoge05	kamo	membrane
7	gene6	hoge06	netteba	humei
8	gene7	hoge07	tebasaki	nuclear
9	gene8	hoge08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



	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作成し(例: list.txt)
2. 該当箇所を変更し、Rコンソール画面上でコピペ



解答例

- 目的のキーワードリストを作成し(例: `list.txt`)
- 該当箇所を変更し、Rコンソール画面上でコピペ

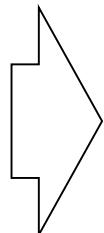
nuclear
membrane

```
run1.txt - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V)

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

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

#本番
obj <- is.element(as.character
out <- data[obj,]
dim(out)
write.table(out, out_f, sep="#"
```



```
run1.txt - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V)

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

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

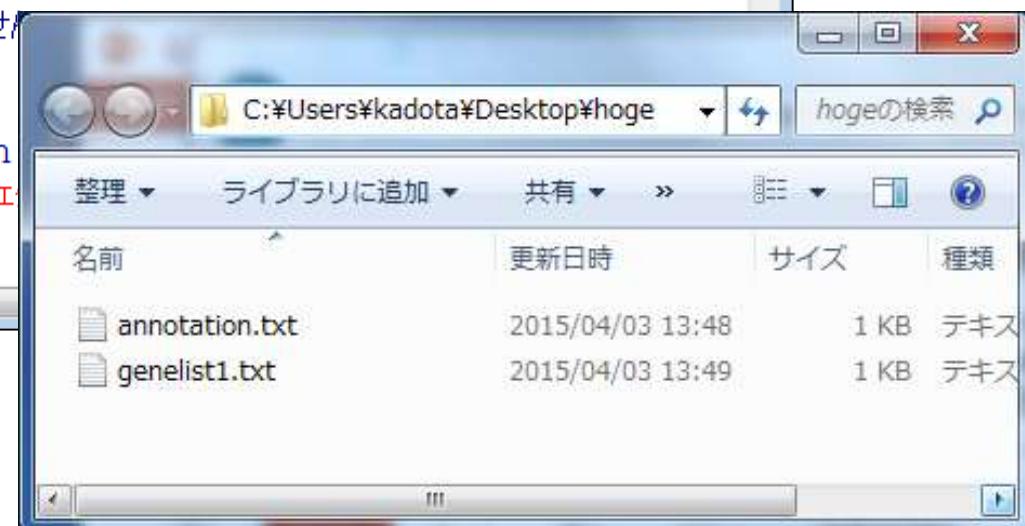
#本番
obj <- is.element(as.character
out <- data[obj,]
dim(out)
write.table(out, out_f, sep="#"
```

ありがちなミス1

R Console

```
> getwd()
[1] "C:/Users/kadota/Documents" #入力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。
> in_f1 <- "annotation.txt"
> in_f2 <- "genelist1.txt"
> out_f <- "hogel.txt"
> param <- 1
>
> #入力ファイルの読み込み
> data <- read.table(in_f1, header=TRUE, sep="\t", quote="")
  以下にエラー file(file, "rt") : コネクションを開くことができません
  追加情報: 警告メッセージ:
In file(file, "rt") :
  ファイル 'annotation.txt' を開くことができません: No such file or directory
> keywords <- readLines(in_f2) #in_f2で指定したファイルの読み込み
  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
> dim(data) #オブジェクトの次元を確認
NULL
```

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。

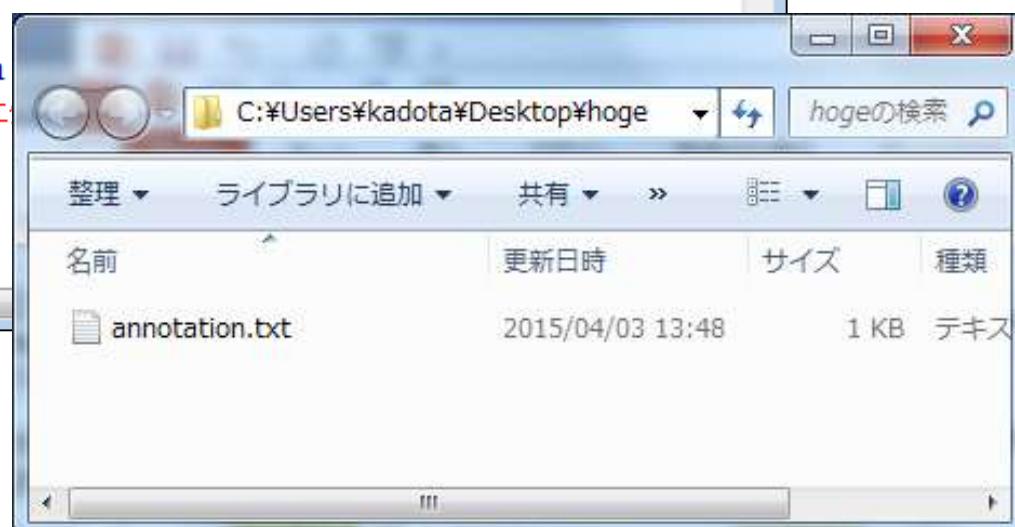


ありがちなミス2

R R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files()
[1] "annotation.txt"
> in_f1 <- "annotation.txt"                                #入力ファイル名を指定してin_f1に格納($)
> in_f2 <- "genelist1.txt"                                 #入力ファイル名を指定してin_f2に格納($)
> out_f <- "hogel.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$
>
> #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$
> keywords <- readLines(in_f2)                             #in_f2で指定したファイルの読み込み
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
#オブジェ
```

必要な入力ファイルが作業ディレクトリ中に存在しない。この場合、in_f2で指定したgenelist1.txtが存在しないため、その読み込み段階でエラーが出ている。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

R R Console

```

> #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
> keywords <- readLines(in_f2)
#in_f2で指定したファイル
> dim(data)
#オブジェクトの次元
[1] 11 4
>
> #本番
> obj <- is.element(as.character(data[,param]), keywords)
#objがTRUEなら
> out <- data[obj,]
#オブジェクトをデータフレームに変換
> dim(out)
#出力データフレームの次元
[1] 3 4
>
> #ファイルに保存
> write.table(out, out_f, sep="\t", append=F,
以下にエラー
  file(file, ifelse	append, "a", "w"))
コネクションを開くことができません
追加情報: 警告メッセージ:
In file(file, ifelse-append, "a", "w")) :
  ファイル 'hogel.txt' を開くことができません: Permission denied
> |

```

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。

A	B	C	D	E
1	gene name	accession	description	subcellular location
2	gene1	hoge01	plasma_mer	nuclear
3	gene7	hoge07	tebasaki	nuclear
4	gene9	hoge09	nihonshu	nuclear
5				

ありがちなミス4

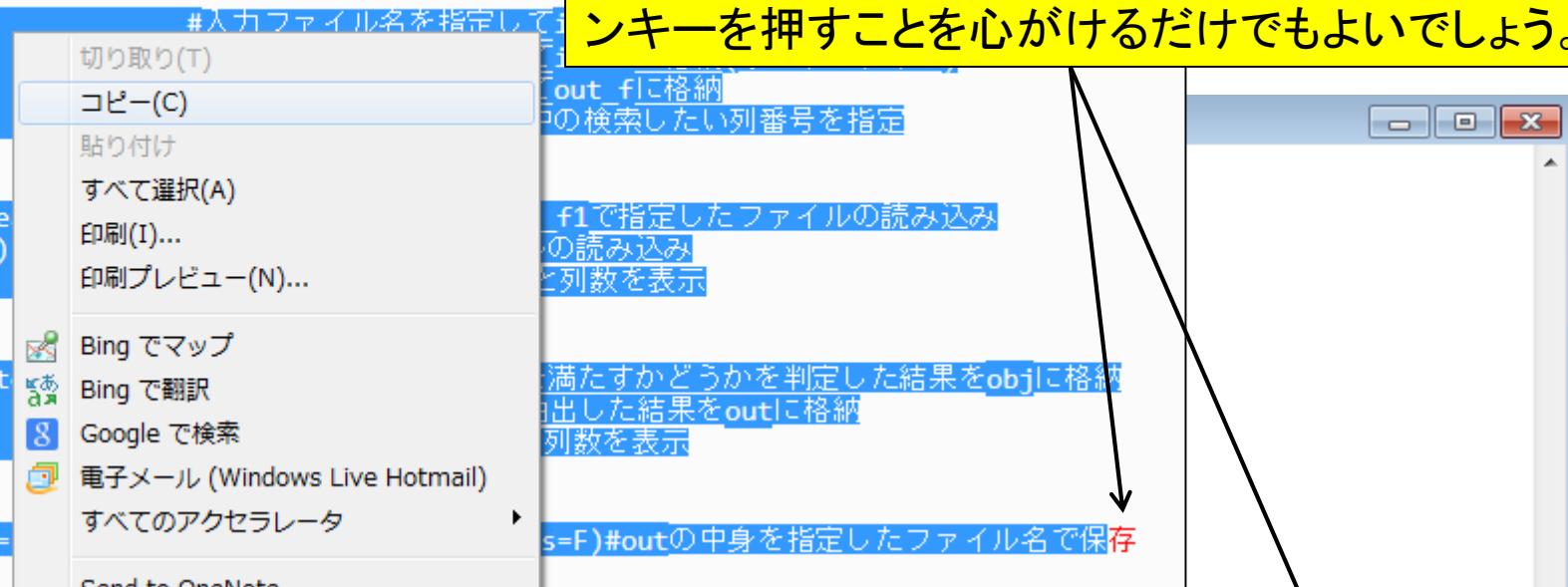
1. 目的のタブ区切りテキストファイル(annotation.txt)中の第1列目をキーとして、リストを出力したい場合:

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

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



```
$param], keywords) #条件を満たすかどうかを判定した結果をobjに格納
$ #objがTRUEとなる行のみ抽出した結果をoutに格納
$ #オブジェクトoutの行数と列数を表示
```

```
$end=F, quote=F, row.names=F) #outの中身を指定したファイル名で保
```

警告メッセージ

R R Console

```

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

```

list.txtファイル作成時に、membraneと打った後に改行を①入れた場合と②入れない場合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は警告メッセージの記述内容が比較的分かりやすいのでよく読むべし。

R R Console

```

> in_f1 <- "annotation.txt"
> in_f2 <- "list.txt"    nuclear↓
> out_f <- "hogel.txt"  membrane↓
> param <- 4            ←
>
> #入力ファイルの読み込み
> data <- read.table(in_f1, header=TRUE, sep="\t", quote
> keywords <- readLines(in_f2)          #in_f2で指定したア
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
#オブジェクトdataの行数
> dim(data)
[1] 11  4
>
> #本番
> obj <- is.element(as.character(data[,param]), keywords)
> out <- data[obj,]                  #objがTRUEとなる行数
> dim(out)
[1] 7  4
>
> #ファイルに保存

```

Contents

■ R基礎(初級)

- おさらい
- コード内部の説明(ファイルの読み込み、行列演算の基礎)
- リアルRNA-seqカウントデータ(数値行列データ)

■ R各種パッケージ(中級): 代表的なパッケージの利用法

- (パッケージのインストール法)
- 基本情報取得(コンティグ数、配列長、N50、GC含量)
- 任意の領域の切り出し
- GC含量計算部分の説明

コード内部の説明

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。

- 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"
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="")
keywords <- readLines(in_f2)
dim(data)

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

R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files()
[1] "annotation.txt" "genelist1.txt"
> in_f1 <- "annotation.txt" #入力ファイル名を$in_f1に
> in_f2 <- "genelist1.txt" #入力ファイル名を$in_f2に
> out_f <- "hoge1.txt" #出力ファイル名を$out_fに
> param <- 1 #アノテーション数を$paramに
>
> #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$in_f1をデータフレーム$dataに
> keywords <- readLines(in_f2)
#in_f2で指定した$in_f2を$keywordsに
> dim(data)
[1] 11   4
> |
```

読み込み

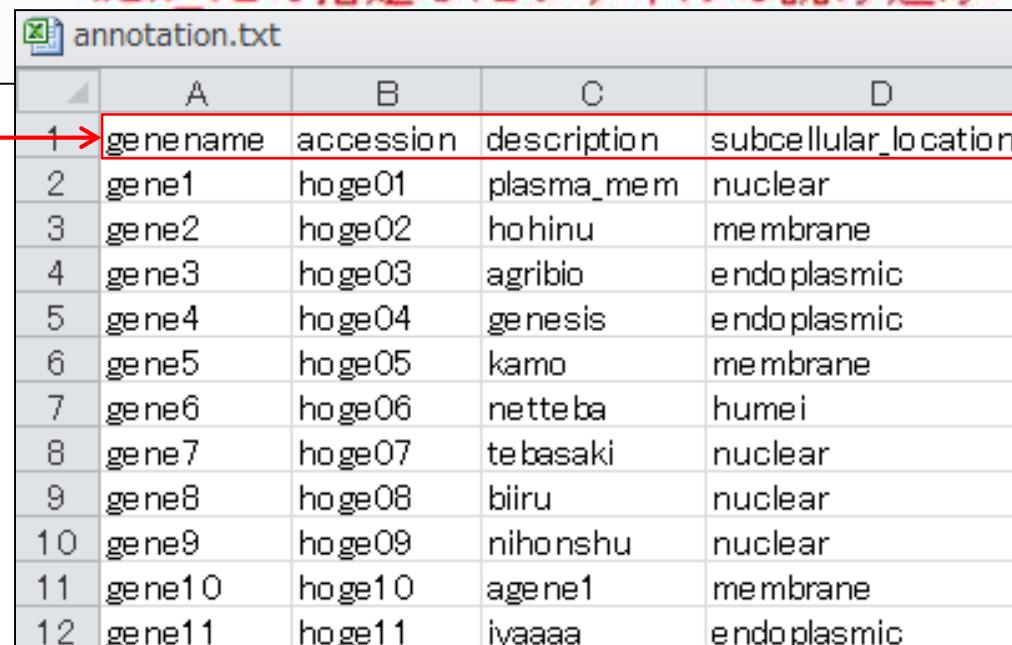
```
in_f1 <- "annotation.txt"
in_f2 <- "genelist1.txt"
out_f <- "hogel.txt"
param <- 1
```

#**④** 入力ファイルの読み込み

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

- ① in_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
- ③ ファイルの区切り文字はタブです
- ④ 読み込んだ結果をdataという名前で取り扱う

#入力ファイル名を指定してin_f1に格納
#出力ファイル名を指定してout_f1に格納
#アノテーションファイル中の検索したい



	A	B	C	D
1	genename	accession	description	subcellular_location
2	gene1	hoge01	plasma_mem	nuclear
3	gene2	hoge02	ho hinu	membrane
4	gene3	hoge03	agribio	endo plasmic
5	gene4	hoge04	genesis	endo plasmic
6	gene5	hoge05	kamo	membrane
7	gene6	hoge06	nette ba	humei
8	gene7	hoge07	te basaki	nuclear
9	gene8	hoge08	biiru	nuclear
10	gene9	hoge09	nihon shu	nuclear
11	gene10	hoge10	age ne1	membrane
12	gene11	hoge11	iyaaaa	endo plasmic

行列data

②

```
> data <- read.table(in_f1, header=TRUE, sep="\t", quote="$"
> keywords <- readLines(in_f2)
> dim(data)
[1] 11 4
```

> data

genename accession description subcellular_location

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

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

dataと打ってリターン。入力ファイルの中身を正しく読み込んでいることがわかる。
 ②header=TRUEとしているので、③このように見えて列名として認識される。

dimで行数と列数を表示

①オブジェクトdataの行数と列数は11と4。
②ウェブページ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。

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

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

#本番

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

#ファイルに保存

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

#入力ファイル名を指定してin_f1に格納
#入力ファイル名を指定してin_f2に格納
#出力ファイル名を指定してout_fに格納
#アノテーションファイル中の検索したい

R Console

```
> data <- read.table(in_f1, header=TRUE, sep="\t", quote="$")
> keywords <- readLines(in_f2)                                #in_f2で指定した$
> dim(data)                                                 #オブジェクト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	agenel	membrane
11	gene11	hoge11	iyaaaa	endoplasmic

行列の要素へのアクセス

R Console

```
[1] 11 4
```

```
> data
```

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

```
> data[6, 4]
```

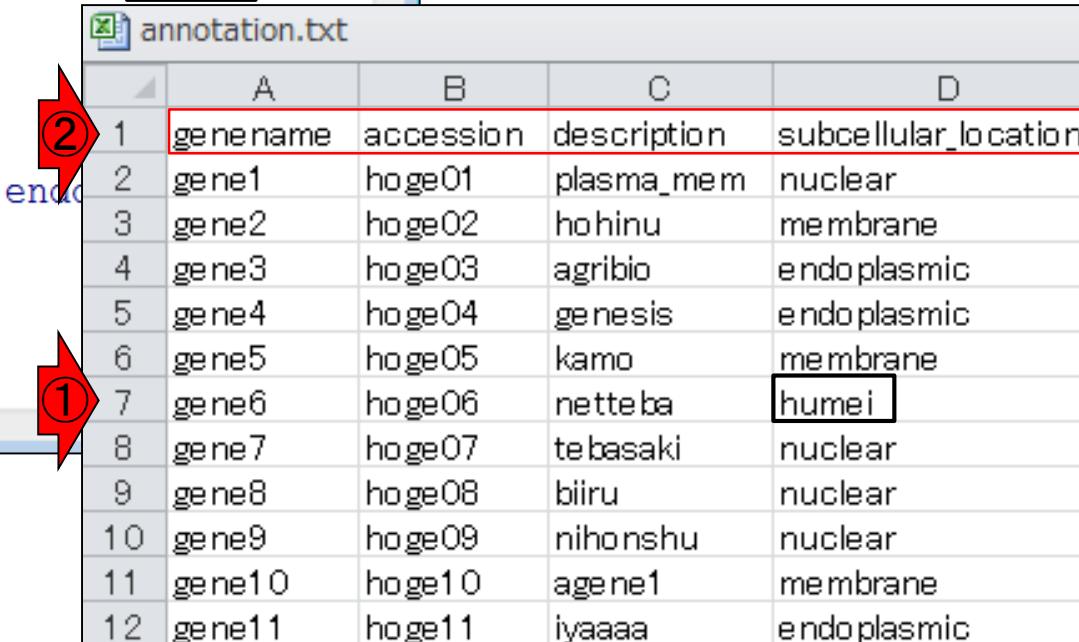
```
[1] humei
```

Levels: endoplasmic humei membrane nuclear

```
> |
```

行列dataの要素へのアクセスは[行, 列]。

①humeiは、読み込み元ファイルのannotation.txt中では7行×4列目だが、
 ②1行目をヘッダー行としているので③
 6行×4列目とする必要がある。利用例
 は、ファイル読み込み時に「x行×y列目に
 て不具合がある」のようなエラーが出た
 時のトラブルシューティングなど。



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	agenel	membrane
12	gene11	hoge11	iyaaaa	endoplasmic

Tips: 上下左右の矢印キー

R R Console

```
[1] 1
> 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    agenel
11  gene11   hoge11    iyaaaaa
> data[6, 4]
[1] humei
Levels: endoplasmic humei membrane nuclear
> data[6, 4]
```

上矢印キーを押すと、直前に打ったコマンドが表示される。最初から全部打ち直すのではなく、上下左右の矢印キーを有効に利用し最小限の労力で打つべし!



行列の要素へのアクセス

R Console

```

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

```

> data[6, 4]

[1] humei

Levels: endoplasmic humei membrane nuclear

> data[2,]

	genename	accession	description	subcellular_location
2	gene2	hoge02	hohinu	membrane

>

行列dataの要素へのアクセスは[行, 列]。

①2行目の情報のみ抽出。読み込み時にhead=TRUEとしていたので、ヘッダー行がついていることが分かる。

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

行列dataの要素へのアクセスは
[行, 列]。①2列目の情報のみ抽出。

行列の要素へのアクセス

R R Console

```

5   gene5    hoge05      kamo
6   gene6    hoge06      netteba
7   gene7    hoge07      tebasaki
8   gene8    hoge08      biiru
9   gene9    hoge09      nihonshu
10  gene10   hoge10      agene1
11  gene11   hoge11      iyaaaa
> data[6, 4]
[1] humei
Levels: endoplasmic humei membrane nuclear
> data[2, ]
  genename accession description subcellular_lo
2  gene2    1 hoge02     hohinu
> data[, 2]
[1] hoge01 hoge02 hoge03 hoge04 hoge05 hoge06
[8] hoge08 hoge09 hoge10 hoge11
11 Levels: hoge01 hoge02 hoge03 hoge04 hoge05 ... hoge11
>

```

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

行列の要素へのアクセス

行列dataの要素へのアクセスは
[行, 列]。①param列目の情報のみ抽出。②paramには1という数値が代入されていったのでこうなる。

R R Console

```

11  gene11    hoge11      iyaaaaa
> data[6, 4]
[1] humei
Levels: endoplasmic humei membrane nuclear
> data[2, ]
  genename accession description subcellular_lo
2   gene2     hoge02     hohinu
[1] hoge01 hoge02 hoge03 hoge04 hoge05 hoge06
[8] hoge08 hoge09 hoge10 hoge11
11 Levels: hoge01 hoge02 hoge03 hoge04 hoge05
> data[, param]
[1] gene1  gene2  gene3  gene4  gene5  gene6
[8] gene8  gene9  gene10 gene11
11 Levels: gene1 gene10 gene11 gene2 gene3 gene
> param
[1] 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	iyaaaaa	endoplasmic

```

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

```

②

Tips: 関数とオプション

行列dataの最初の数行を表示したい場合は、head関数を利用。①n=3というオプションを利用すると最初の3行分のみ表示。関数ごとに様々なオプションを利用可能です。このあたりは②Linuxとよく似ている。

R Console

```
> head(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
```

```
> head(data, n=3)
```

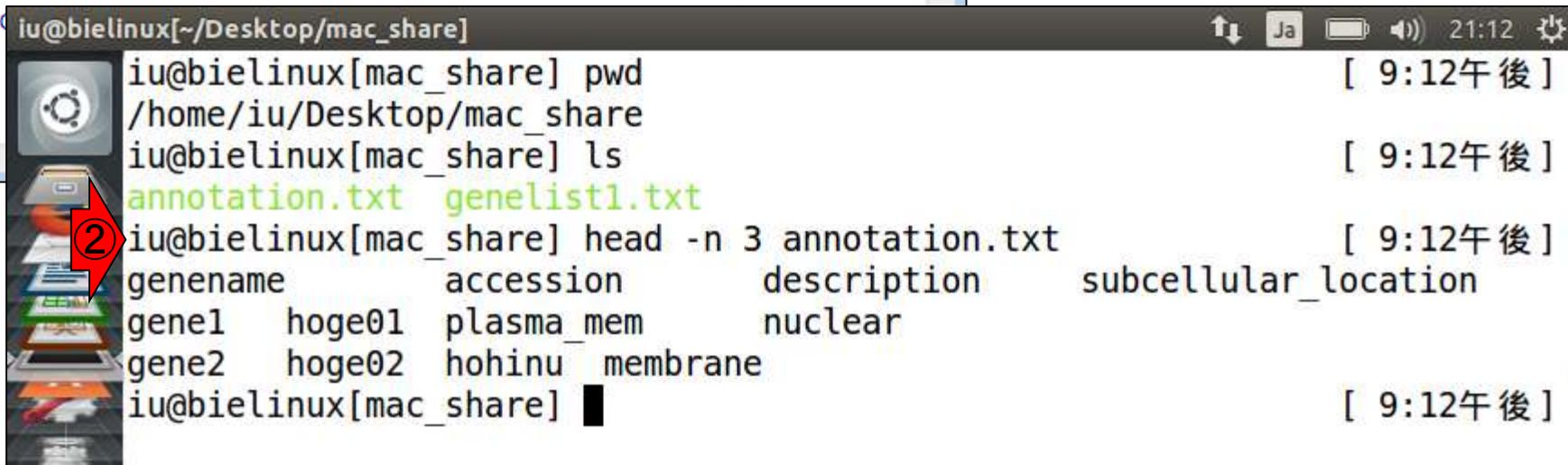
```
  genename accession description subcellular_location
1  gene1    hoge01  plasma_mem          nuclear
2  gene2    hoge02   hohinu            membrane
3  gene3    hoge03  agribio           endoplasmic
```

```
> head(data, n=1)
```

```
  genename accession
```

```
1  gene1
```

iu@bielinux[~/Desktop/mac_share]



```
iu@bielinux[mac_share] pwd [ 9:12 午後 ]
/home/iu/Desktop/mac_share
iu@bielinux[mac_share] ls [ 9:12 午後 ]
annotation.txt genelist1.txt
iu@bielinux[mac_share] head -n 3 annotation.txt [ 9:12 午後 ]
genename accession description subcellular_location
gene1 hoge01 plasma_mem nuclear
gene2 hoge02 hohinu membrane
iu@bielinux[mac_share]
```

Tips: タブ補完

R Console

```
> head(data, n=3)
  genename accession description subcellular_location
1  gene1    hoge01  plasma_mem          nuclear
2  gene2    hoge02   hohinu            membrane
3  gene3    hoge03  agribio           endoplasmic
> head(data, n=1)
  genename accession description subcellular_location
1  gene1    hoge01  plasma_mem          nuclear
> data[, 4]
[1] nuclear      membrane     endoplasmic endoplasmic
[5] membrane     humei        nuclear      nuclear
[9] nuclear      membrane     endoplasmic
Levels: endoplasmic humei membrane nuclear
> data$su
```



列番号を指定する以外にも特定の列を表示するやり方がある。head=TRUEで入力ファイルを読み込むと、列の名前を利用することができる。**①**subcellular_location列の情報を抽出したい場合は、**②**「data\$su」くらいまで打ち込んでからTabキーを押す。

annotation.txt

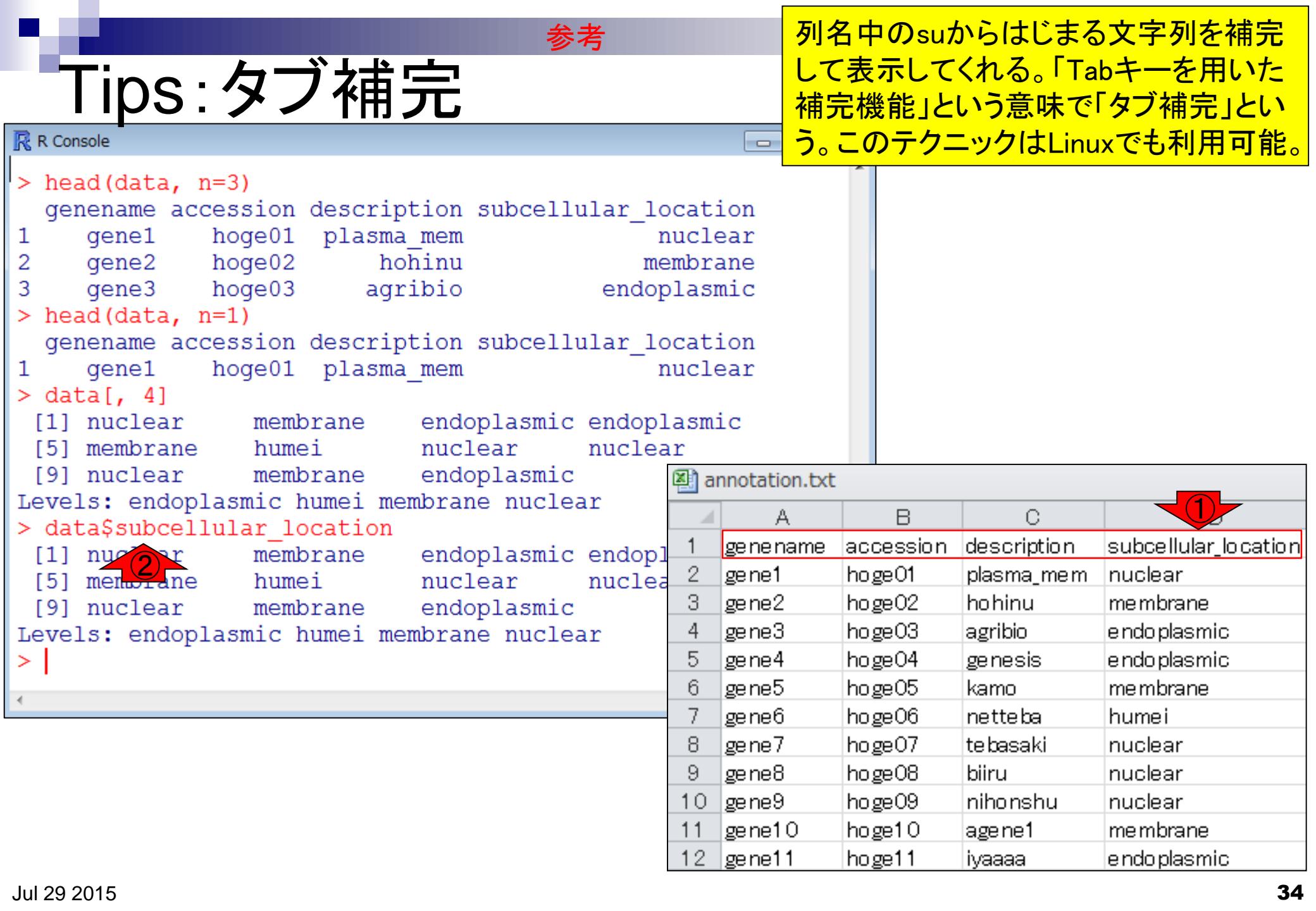
	A	B	C	D
1	genename	accession	description	subcellular_location
2	gene1	hoge01	plasma_mem	nuclear
3	gene2	hoge02	hohinu	membrane
4	gene3	hoge03	agribio	endoplasmic
5	gene4	hoge04	genesis	endoplasmic
6	gene5	hoge05	kamo	membrane
7	gene6	hoge06	netteba	humei
8	gene7	hoge07	tebasaki	nuclear
9	gene8	hoge08	biiru	nuclear
10	gene9	hoge09	nihonshu	nuclear
11	gene10	hoge10	agenie1	membrane
12	gene11	hoge11	iyaaaa	endoplasmic

Tips: タブ補完

R Console

```
> head(data, n=3)
  genename accession description subcellular_location
1  gene1    hoge01  plasma_mem          nuclear
2  gene2    hoge02   hohinu            membrane
3  gene3    hoge03  agribio           endoplasmic
> head(data, n=1)
  genename accession description subcellular_location
1  gene1    hoge01  plasma_mem          nuclear
> data[, 4]
[1] nuclear      membrane      endoplasmic endoplasmic
[5] membrane     humei        nuclear      nuclear
[9] nuclear      membrane      endoplasmic
Levels: endoplasmic humei membrane nuclear
> data$subcellular_location
[1] nuclear      membrane      endoplasmic endoplasmic
[5] membrane     humei        nuclear      nuclear
[9] nuclear      membrane      endoplasmic
Levels: endoplasmic humei membrane nuclear
> |
```

列名中のsuからはじまる文字列を補完して表示してくれる。「Tabキーを用いた補完機能」という意味で「タブ補完」という。このテクニックはLinuxでも利用可能。



The screenshot shows the RStudio interface. In the top-left, the R Console window displays R code and its output. Red annotations are present: a red arrow labeled '②' points to the 'nuclear' entry in the fourth column of the first data frame; another red arrow labeled '①' points to the 'subcellular_location' header in the fourth column of the second data frame. In the bottom-right, a file viewer window titled 'annotation.txt' shows a table with columns A, B, C, and D. The first row contains the column headers: 'genename', 'accession', 'description', and 'subcellular_location'. The entire row is highlighted with a red border.

	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	agenie1	membrane
12	gene11	hoge11	iyaaaa	endoplasmic

Tips:table関数

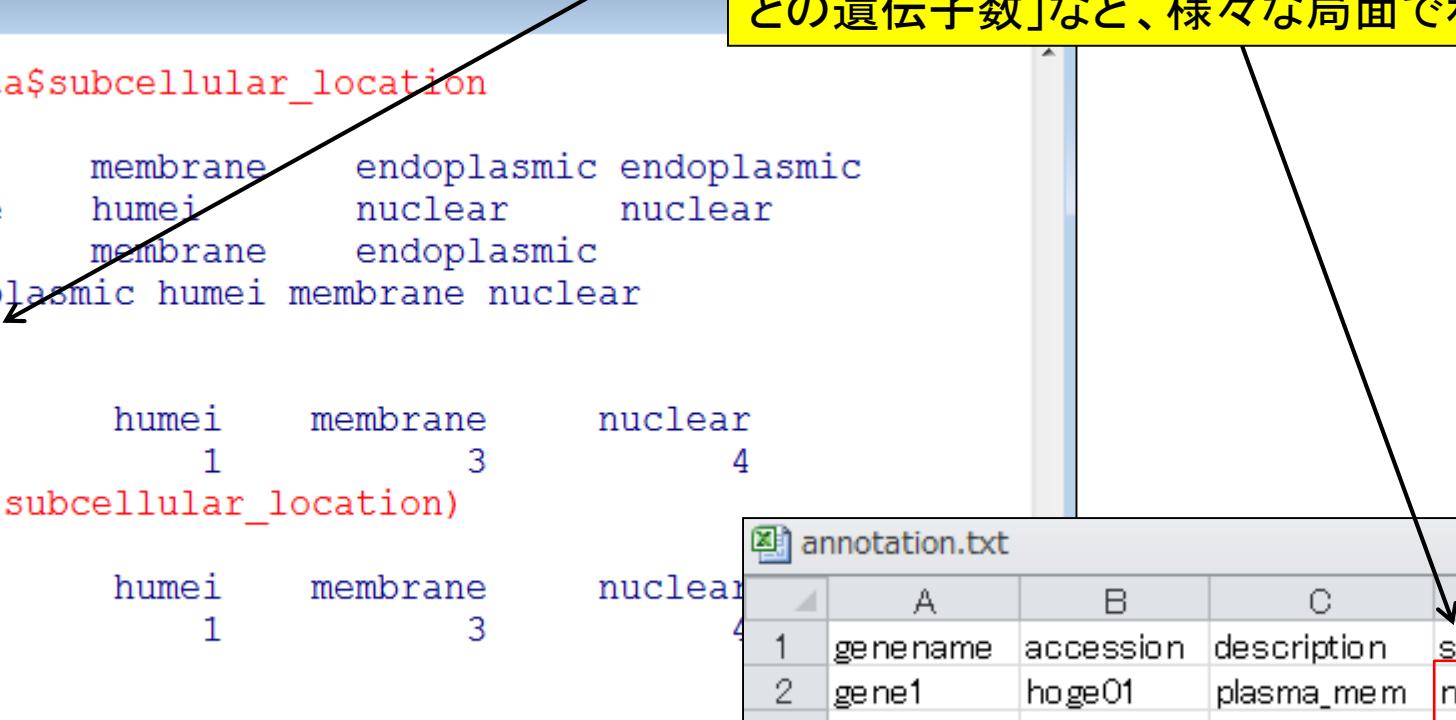
R Console

```

> hoge <- data$subcellular_location
> hoge
[1] nuclear      membrane      endoplasmic endoplasmic
[5] membrane     humei        nuclear       nuclear
[9] nuclear      membrane      endoplasmic
Levels: endoplasmic humei membrane nuclear
> table(hoge)
hoge
endoplasmic      humei      membrane      nuclear
            3          1          3          4
> table(data$subcellular_location)
subcellular_location
endoplasmic      humei      membrane      nuclear
            3          1          3          4
> |

```

table関数は、ベクトル中の要素ごとの出現回数を返す。「NGSデータ中の特定のリードの出現回数(後述)」や、「アノテーションファイル中の染色体ごとの遺伝子数」など、様々な局面で利用可能。



The screenshot shows the R console output and a file explorer window. The R console displays the creation of a vector 'hoge' from the 'subcellular_location' column of a data frame, followed by two calls to the 'table()' function. The first call to 'table(hoge)' creates a named vector 'hoge' with counts for 'endoplasmic', 'humei', 'membrane', and 'nuclear'. The second call to 'table(data\$subcellular_location)' shows the same counts for the entire column.

Annotation file 'annotation.txt' is shown in the file explorer. It is a CSV file with columns A, B, C, and D. Column D, 'subcellular_location', contains values like 'nuclear', 'membrane', 'humei', and 'endo plasmic'. The last row, 'endo plasmic', is highlighted with a red border.

	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	endo plasmic
5	gene4	hoge04	genesis	endo plasmic
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	agenel1	membrane
12	gene11	hoge11	iyaaaa	endo plasmic

sort関数と併用することで全体像を俯瞰可能。例えば①nuclearに局在する遺伝子数が最も多く4個であった、などが簡単にわかる。

Tips: ソート

```
R R Console
> sort(table(hoge))
hoge
  humei endoplasmic    membrane      nuclear
      1           3           3           4
> sort(table(hoge), decreasing=T)
hoge
  nuclear endoplasmic    membrane      humei
      4           3           3           1
> hoge2 <- table(hoge)
> sort(hoge2, decreasing=T)
hoge
  nuclear endoplasmic    membrane      humei
      4           3           3           1
> | 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	agenie1	membrane
12	gene11	hoge11	iyaaaa	endoplasmic

Tips : is.element関数

hogeベクトルに対して、"nuclear"の文字が存在する場所をTRUE、存在しない場所をFALSEとして返す。as.character関数は、文字列ベクトルとして取り扱いたい場合に利用。

R Console

```
> hoge
[1] nuclear      membrane     endoplasmic endoplasmic
[5] membrane     humei        nuclear      nuclear
[9] nuclear      membrane     endoplasmic
Levels: endoplasmic humei membrane nuclear
> is.element(hoge, "nuclear")
[1] TRUE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
[10] FALSE FALSE
> as.character(hoge)
[1] "nuclear"    "membrane"   "endoplasmic"
[4] "endoplasmic" "membrane"   "humei"
[7] "nuclear"    "nuclear"    "nuclear"
[10] "membrane"   "endoplasmic"
> is.element(as.character(hoge), "nuclear")
[1] TRUE FALSE FALSE FALSE FALSE FALSE  TRUE
[10] FALSE FALSE
> |
```

	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	agenel	membrane
12	gene11	hoge11	iyaaaa	endoplasmic

Tips: “二重クオーテーション”

R R Console

```
> is.element(hoge, "nuclear")
[1] TRUE FALSE FALSE FALSE FALSE FALSE
[7] TRUE TRUE TRUE FALSE FALSE
> is.element(hoge, "nuclear")
> is.element(hoge, "nuclear")
> is.element(hoge, "nuclear")
[1] TRUE FALSE FALSE FALSE FALSE FALSE
[7] TRUE TRUE TRUE FALSE FALSE
> |
```

二重クオーテーションが自動で
変更されるエディタは非推奨で
す。日本語の二重クオーテー
ションもだめです。Microsoft
WordやPDFファイル中のコード
のコピペ時によくハマります。

目的をおさらい

目的是、数万～数百万行からなるファイルを読み込んで特定のキーワードを含む行のみ取り出すテクニックを習得。

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

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

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

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

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



目的をおさらい

#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  
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
```

入力2:リストファイル
(genelist1.txt)

	A
1	gene1
2	gene7
3	gene9

```
R Console  
> data[, param]  
[1] gene1 gene2 gene3 gene4 gene5 gene6 gene7  
[8] gene8 gene9 gene10 gene11  
11 Levels: gene1 gene10 gene11 gene2 gene3 gene4 ... gene9  
> keywords  
[1] "gene1" "gene7" "gene9"  
> is.element(as.character(data[,param]), keywords)  
[1] TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE TRUE  
[10] FALSE FALSE  
> obj <- is.element(as.character(data[,param]), keywords)  
> obj  
[1] TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE TRUE  
[10] 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  
> |
```

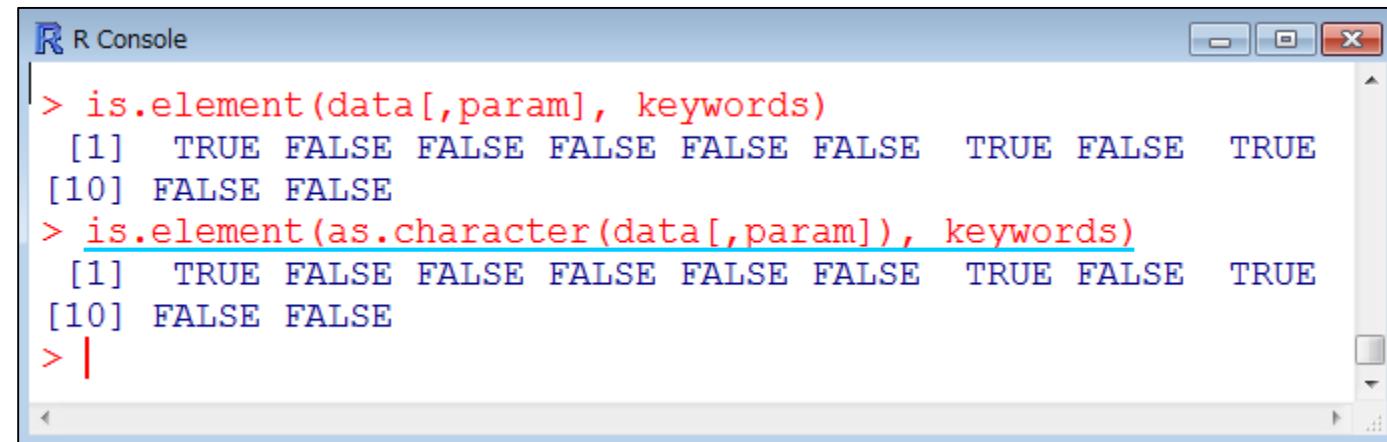
目的をおさらい

#本番

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

#オブジェクトoutの行数と列数を表示

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。少なくとも現在(R ver. 3.1.3以降)は、この関数がなくても大丈夫なようだ。同じ関数でもバージョンによって挙動が異なるということ(バージョンの違いの一例)。



The screenshot shows the R Console window with the title "R Console". It displays two identical command-line outputs. The first command is `> is.element(data[,param], keywords)`, which returns a vector of length 10 with values [1] TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE TRUE and [10] FALSE FALSE. The second command is `> is.element(as.character(data[,param]), keywords)`, which also returns the same vector. A cursor is visible at the end of the second command line.

```
> is.element(data[,param], keywords)
[1] TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE TRUE
[10] FALSE FALSE
> is.element(as.character(data[,param]), keywords)
[1] TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE TRUE
[10] FALSE FALSE
> |
```

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

```

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

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

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

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

```

genelist1.txt

	A
1	gene1
2	gene7
3	gene9

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

```

```

in_f <- "annotation.txt"          #入力ファイル名を指定してin_fに格納(アノテーション)
out_f <- "hoge12.txt"            #出力ファイル名を指定してout_fに格納
param1 <- 1                      #アノテーションファイル中の検索したい列番号を指定
param2 <- c("gene1", "gene7", "gene9") #検索したい文字列を指定

```

このコードはヘッダー行
がある場合のものです

```

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

#本番
obj <- is.element(as.character(data[,param1]), param2)#条件を満たすかどうかを判定した結果をobjに格納
out <- data[obj,]                                       #objがTRUEとなる行のみ抽出した結果をoutに格納
dim(out)                                                 #オブジェクトoutの行数と列数を表示

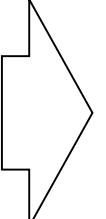
#ファイルに保存
write.table(out, out_f, sep="\t", append=F, quote=F, row.names=F)#outの中身を指定したファイル名で保存

```

入力: annotation.txt

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

出力: hoge12.txt



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

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:

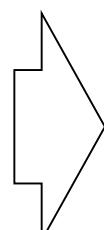
入力ファイル中にヘッダー行がない場合の読み込み例です。

```
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  
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

```
in_f <- "annotation.txt"          #入力ファイル名を指定してin_fに格納(アノテーションファイル)
out_f <- "hoge12.txt"            #出力ファイル名を指定してout_fに格納
param1 <- 1                      #アノテーションファイル中の検索したい列番号を指定
param2 <- c("gene1", "gene7", "gene9") #検索したい文字列を指定
```

ヘッダー行がある場合

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

#本番
obj <- is.element(as.character(data[,param1]), param2)#条件を満たすかどうかを判定した結果をobjに格納
out <- data[obj,]                                #objがTRUEとなる行のみ抽出した結果をoutに格納
dim(out)                                         #オブジェクトoutの行数と列数を表示

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

入力ファイル中にヘッダー行がない場合の読み込み例です。

```
in_f <- "annotation2.txt"          #入力ファイル名を指定してin_fに格納(アノテーションファイル)
out_f <- "hoge13.txt"            #出力ファイル名を指定してout_fに格納
param1 <- 1                      #アノテーションファイル中の検索したい列番号を指定
param2 <- c("gene1", "gene7", "gene9") #検索したい文字列を指定
```

ヘッダー行がない場合

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

#本番
obj <- is.element(as.character(data[,param1]), param2)#条件を満たすかどうかを判定した結果をobjに格納
out <- data[obj,]                                #objがTRUEとなる行のみ抽出した結果をoutに格納
dim(out)                                         #オブジェクトoutの行数と列数を表示

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

Contents

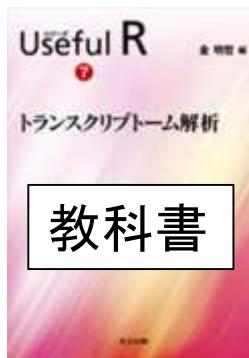
■ R基礎(初級)

- おさらい
- コード内部の説明(ファイルの読み込み、行列演算の基礎)
- リアルRNA-seqカウントデータ(数値行列データ)

■ R各種パッケージ(中級): 代表的なパッケージの利用法

- (パッケージのインストール法)
- 基本情報取得(コンティグ数、配列長、N50、GC含量)
- 任意の領域の切り出し
- GC含量計算部分の説明

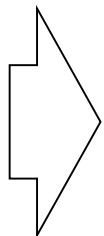
カウントデータ



カウントデータとは、「マップされたリード数」をカウントしたデータのこと。以下の例では1サンプルなので1列分のデータしかないが、一般には複数サンプルのデータを取得し、サンプル間比較が行われるので複数の列からなる。それゆえ、数値ベクトルではなく数値行列。詳細は8/5のRNA-seq前半で。

目的サンプルの
RNA-Seqデータ

mapping



リファレンス配列:ゲノム



count

	T1
遺伝子1	14
遺伝子2	5
遺伝子3	12
遺伝子4	1
遺伝子5	...
...	...

・サンプルデータ

数値行列

・(削除予定)個別パッケージのインストール (last modified 2015/02/20)

・基本的な利用法 (last modified 2015/04/03)

・サンプルデータ (modified 2015/06/15) NEW

・バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ) | NGSハンドブック

・バイオハンドブック

・書籍

・書籍

・書籍

実験の詳細には立ち入らないが、3生物種間比較を行った公共RNA-seqカウントデータ(Blekhman et al., Genome Res., 2010)を用いて、Rの王道的な使い方である数値行列解析のテクニックを伝授。8/5の統計解析のところでこのデータを利用予定です。

サンプルデータ NEW

1. 41. Blekhman et al., Genome Res., 2010のリアルカウントデータです。Supplementary Table1で提供されているエクセルファイル (<http://genome.cshlp.org/content/suppl/2009/12/16/gr.099226.109.DC1/suppTable1.xls>; 約4.3MB) からカウントデータのみ抽出し、きれいに整形しなおしたものがここでの出力ファイルになります。20,689 genes×36 samplesのカウントデータ([sample_blekhman_36.txt](#))です。実験デザインの詳細はFigure S1中に描かれていますが、ヒト(Homo Sapiens; HS), チンパンジー(Pan troglodytes; PT), アカゲザル(Rhesus macaque; RM)の3種類の生物種の肝臓サンプル(liver sample)の比較を行っています。生物種ごとにオス3個体メス3個体の計6個体使われており(six individuals; six biological replicates)、技術的なばらつき(technical variation)を見積もるべく各個体は2つに分割されてデータが取得されています(duplicates; two technical replicates)。それゆえ、ヒト12サンプル、チンパンジー12サンプル、アカゲザル12サンプルの計36サンプル分のデータということになります。以下で行っていることはカウントデータの列のみ「ヒトのメス(HSF1, HSF2, HSF3)」「ヒトのオス(HSM1, HSM2, HSM3)」「チンパンジーのメス(PTF1, PTF2, PTF3)」「チンパンジーのオス(PTM1, PTM2, PTM3)」「アカゲザルのメス(RMF1, RMF2, RMF3)」「アカゲザルのオス(RMM1, RMM2, RMM3)」の順番で並び替えたものをファイルに保存しています。もう少し美しくやることも原理的には可能ですが、そこは本質的な部分ではありませんので、ここではアドホック(その場しのぎ、の意味)な手順で行っています。当然ながら、エクセルなどでファイルの中身を眺めて完全に列名を把握しているという前提です。尚、「R1L4.HSF1」と「R4L2.HSF1」が「HSF1」というヒトのメス一個体のtechnical replicatesであることは列名や文脈から読み解けます。

```
#in_f <- "http://genome.cshlp.org/content/suppl/2009/12/16/gr.099226.109.DC1/suppTable1.xls">#入力ファイル名を指定してin_fに格納
in_f <- "suppTable1.xls"
out_f <- "sample_blekhman_36.txt" #出力ファイル名を指定してout_fに格納
```

#入力ファイルの読み込み

```
hoge <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="")#in_fで指定したファイルの読み込み
dim(hoge) #行数と列数を表示
```

#サブセットの取得

```
data <- cbind( #必要な列名の情報を取得したい列の順番で結合した結果をdataに格納
  hoge$R1L4.HSF1, hoge$R4L2.HSF1, hoge$R2L7.HSF2, hoge$R3L2.HSF2, hoge$R8L1.HSF3, hoge$R8L2.HSF3,
  hoge$R1L1.HSM1, hoge$R5L2.HSM1, hoge$R2L3.HSM2, hoge$R4L8.HSM2, hoge$R3L6.HSM3, hoge$R4L1.HSM3,
  hoge$R1L2.PTF1, hoge$R4L4.PTF1, hoge$R2L4.PTF2, hoge$R6L6.PTF2, hoge$R3L7.PTF3, hoge$R5L3.PTF3,
  hoge$R1L6.PTM1, hoge$R3L3.PTM1, hoge$R2L8.PTM2, hoge$R4L6.PTM2, hoge$R6L2.PTM3, hoge$R6L4.PTM3,
  hoge$R1L7.RMF1, hoge$R5L1.RMF1, hoge$R2L2.RMF2, hoge$R5L8.RMF2, hoge$R3L4.RMF3, hoge$R4L7.RMF3,
  hoge$R1L3.RMM1, hoge$R3L8.RMM1, hoge$R2L6.RMM2, hoge$R5L4.RMM2, hoge$R3L1.RMM3, hoge$R4L3.RMM3)
```

xls形式ファイルもOK

41. Blekhman et al., Genome Res., 2010のリアルカウントデータです。Supplementary (<http://genome.cshlp.org/content/suppl/2009/12/16/gr.099226.109.DC1/suppTable1.xls>)に整形しなおしたものがここでの出力ファイルになります。20,689 genes×36 samplesの詳細はFigure S1中に描かれていますが、ヒト (Homo Sapiens; HS), macaque; RM) の3種類の生物種の肝臓サンプル(liver sample)の比較を行っており(six individuals; six biological replicates)、技術的なばらつき(technical variance)が取得されています(duplicates; two technical replicates)。それゆえ、ヒト12サンプルの計36サンプル分のデータということになります。以下で行っていることはカウントのオス(HSM1, HSM2, HSM3)」「チンパンジーのメス(PTF1, PTF2, PTF3)」「チンパンジーのメス(RMF1, RMF2, RMF3)」「アカゲザルのオス(RMM1, RMM2, RMM3)」のよう少し美しくやることも原理的には可能ですが、そこは本質的な部分ではありません。手順で行っています。当然ながら、エクセルなどでファイルの中身を眺めて完全に列名を把握しているという前提です。

尚、"R1L4.HSF1"と"R4L2.HSF1"が「HSF1というヒトのメス一個体のtechnical replicates」であることは列名や文脈から読み解けます。

```
#in_f <- "http://genome.cshlp.org/content/suppl/2009/12/16/gr.099226.109.DC1/suppTable1.xls" #①ファイル名を指定してin_fに格納
in_f <- "suppTable1.xls" #②入力ファイル名を指定してin_fに格納
out_f <- "sample_blekhman.xls" #出力ファイル名を指定してout_fに格納

#入力ファイルの読み込み
hoge <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="")#in_fで指定したファイルの読み込み
dim(hoge) #行数と列数を表示

#サブセットの取得
data <- cbind(
  hoge$R1L4.HSF1, hoge$R4L2.HSF1, hoge$R2L7.HSF2, hoge$R3L2.HSF2, hoge$R8L1.HSF3, hoge$R8L2.HSF3,
  hoge$R1L1.HSM1, hoge$R5L2.HSM1, hoge$R2L3.HSM2, hoge$R4L8.HSM2, hoge$R3L6.HSM3, hoge$R4L1.HSM3,
  hoge$R1L2.PTF1, hoge$R4L4.PTF1, hoge$R2L4.PTF2, hoge$R6L6.PTF2, hoge$R3L7.PTF3, hoge$R5L3.PTF3,
  hoge$R1L6.PTM1, hoge$R3L3.PTM1, hoge$R2L8.PTM2, hoge$R4L6.PTM2, hoge$R6L2.PTM3, hoge$R6L4.PTM3,
  hoge$R1L7.RMF1, hoge$R5L1.RMF1, hoge$R2L2.RMF2, hoge$R5L8.RMF2, hoge$R3L4.RMF3, hoge$R4L7.RMF3,
  hoge$R1L3.RMM1, hoge$R3L8.RMM1, hoge$R2L6.RMM2, hoge$R5L4.RMM2, hoge$R3L1.RMM3, hoge$R4L3.RMM3)
colnames(data) <- c( #列名を付加
  "R1L4.HSF1", "R4L2.HSF1", "R2L7.HSF2", "R3L2.HSF2", "R8L1.HSF3", "R8L2.HSF3",
  "R1L1.HSM1", "R5L2.HSM1", "R2L3.HSM2", "R4L8.HSM2", "R3L6.HSM3", "R4L1.HSM3",
  "R1L2.PTF1", "R4L4.PTF1", "R2L4.PTF2", "R6L6.PTF2", "R3L7.PTF3", "R5L3.PTF3",
  "R1L6.PTM1", "R3L3.PTM1", "R2L8.PTM2", "R4L6.PTM2", "R6L2.PTM3", "R6L4.PTM3",
  "R1L7.RMF1", "R5L1.RMF1", "R2L2.RMF2", "R5L8.RMF2", "R3L4.RMF3", "R4L7.RMF3")
```

①xls形式のエクセルファイルを読み込むことができる。(但し、このファイルは壊れているなどというメッセージが出ており、実はタブ区切りテキストファイルなのに、.xlsという拡張子が無理やりつけられているというオチかもしれない…)

②それほど大きなサイズでなければ、ネットワーク経由で直接読み込むこともできる。他に、read.csvやreadLines関数などを駆使してファイルを読み込むことができる。

#以降は無視される

41. Blekhman et al., Genome Res., 2010のリアルカウントデータです。Suppl (<http://genome.cshlp.org/content/suppl/2009/12/16/gr.099226.109.DC1/suppTable1.xls>)に整形しなおしたものがここでの出力ファイルになります。20,689 genes×3種類の生物種の肝臓サンプル(liver sample)の比較を行っており(six individuals; six biological replicates)、技術的なばらつき(technical variation)を元にするべく各個体は2つに分かれていますが取得されています(duplicates; two technical replicates)。それゆえ、ヒト12サンプル、チンパンジー12サンプル、アカゲザル12サンプルの計36サンプル分のデータということになります。以下で行っていることはカウントデータの列のみ「ヒトのメス(HSF1, HSF2, HSF3)」「ヒトのオス(HSM1, HSM2, HSM3)」「チンパンジーのメス(PTF1, PTF2, PTF3)」「チンパンジーのオス(PTM1, PTM2, PTM3)」「アカゲザルのメス(RMF1, RMF2, RMF3)」「アカゲザルのオス(RMM1, RMM2, RMM3)」の順番で並び替えたものをファイルに保存しています。もう少し美しくやることも原理的には可能ですが、そこは本質的な部分ではありませんので、ここではアドホック(その場しのぎ、の意味)な手順で行っています。当然ながら、エクセルなどでファイルの中身を眺めて完全に列名を把握しているという前提です。

③先頭に#がついているものは無視される(実行されない)。つまり②のコマンドは無効で、①のコマンドのみが実行される。①だけではこのファイルをどこから取得したのかわからないが、このようにコメントアウト(#をつけること)して完全なURL情報がわかるようにしている。このあたりはLinuxのシェルスクリプトと同じ。

③ "R4L4.HSF1"と"R4L2.HSF1"が「HSF1というヒトのメス一個体のtechnical replicates」であることは列名や文脈から読み解けます。

```
#in_f <- "http://genome.cshlp.org/content/suppl/2009/12/16/gr.099226.109.DC1/suppTable1.xls" #②のファイル  
in_f <- "suppTable1.xls" #入力ファイル名を指定してin_fに格納  
out_f <- "sample_blekhman.txt" #出力ファイル名を指定してout_fに格納
```

#入力ファイルの読み込み

```
hoge <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #in_fで指定したファイルの読み込み  
dim(hoge) #行数と列数を表示
```

#サブセットの取得

```
data <- cbind( #必要な列名の情報を取得したい列の順番で結合した結果をdataに格納  
  hoge$R1L4.HSF1, hoge$R4L2.HSF1, hoge$R2L7.HSF2, hoge$R3L2.HSF2, hoge$R8L1.HSF3, hoge$R8L2.HSF3,  
  hoge$R1L1.HSM1, hoge$R5L2.HSM1, hoge$R2L3.HSM2, hoge$R4L8.HSM2, hoge$R3L6.HSM3, hoge$R4L1.HSM3,  
  hoge$R1L2.PTF1, hoge$R4L4.PTF1, hoge$R2L4.PTF2, hoge$R6L6.PTF2, hoge$R3L7.PTF3, hoge$R5L3.PTF3,  
  hoge$R1L6.PTM1, hoge$R3L3.PTM1, hoge$R2L8.PTM2, hoge$R4L6.PTM2, hoge$R6L2.PTM3, hoge$R6L4.PTM3,  
  hoge$R1L7.RMF1, hoge$R5L1.RMF1, hoge$R2L2.RMF2, hoge$R5L8.RMF2, hoge$R3L4.RMF3, hoge$R4L7.RMF3,  
  hoge$R1L3.RMM1, hoge$R3L8.RMM1, hoge$R2L6.RMM2, hoge$R5L4.RMM2, hoge$R3L1.RMM3, hoge$R4L3.RMM3)  
colnames(data) <- c( #列名を付加  
  "R1L4.HSF1", "R4L2.HSF1", "R2L7.HSF2", "R3L2.HSF2", "R8L1.HSF3", "R8L2.HSF3",  
  "R1L1.HSM1", "R5L2.HSM1", "R2L3.HSM2", "R4L8.HSM2", "R3L6.HSM3", "R4L1.HSM3",  
  "R1L2.PTF1", "R4L4.PTF1", "R2L4.PTF2", "R6L6.PTF2", "R3L7.PTF3", "R5L3.PTF3",  
  "R1L6.PTM1", "R3L3.PTM1", "R2L8.PTM2", "R4L6.PTM2", "R6L2.PTM3", "R6L4.PTM3",  
  "R1L7.RMF1", "R5L1.RMF1", "R2L2.RMF2", "R5L8.RMF2", "R3L4.RMF3", "R4L7.RMF3",
```

list.files, file.info

41. Blekhman et al., Genome Res., 2010のリアルカウントデータです。Suppl. (<http://genome.cshlp.org/content/suppl/2009/12/16/gr.099226.109.DC1/sup>)

①getwd()で作業フォルダの確認。②list.files()で解析したいファイルの存在確認。”supp”を含むファイル名のもののみ出力させるテクニック。③file.info()で④ファイルサイズ(約4.5MB)などの詳細情報がわかる。

に整形しなおしたものがここでの出力ファイルになります。20,689 genes×36 samplesのカウントデータ([sample_blekhman_36.txt](#))です。実験デザインの詳細はFigure S1中に描かれていますが、ヒト(Homo Sapiens; HS), チンパンジー(Pan troglodytes; PT), アカゲザル(Rhesus macaque; RM)の3種類の生物種の肝臓サンプル(liver sample)の比較を行っています。生物種ごとにオス3個体メス3個体の計6個体使われており(six individuals; six biological replicates)、技術的なばらつき(technical variation)を見積もるべく各個体は2つに分割されてデータが取得されています(duplicates; two technical replicates)。それゆえ、ヒト12サンプル、チンパンジー12サンプル、アカゲザル12サンプルの計36サンプル分のデータということになります。以下で行っていることはカウントデータの列のみ「ヒトのメス(HSF1, HSF2, HSF3)」「ヒトのオス(HSM1, HSM2, HSM3)」「チンパンジーのメス(PTF1, PTF2, PTF3)」「チンパンジーのオス(PTM1, PTM2, PTM3)」「アカゲザルのメス(RMF1, RMF2, RMF3)」「アカゲザルのオス(RMM1, RMM2, RMM3)」の順番で並び替えたものをファイルに保存しています。もう少し美しくやることも原理的には可能ですが、そこは本質的な部分ではありませんので、ここではアドホック(その場しのぎ、の意味)な手順で行っています。当然ながら、エクセルなどでファイルの中身を眺めて完全に列名を把握しているという前提です。
尚、"R1L4.HSF1"と"R4L2.HSF1"が「HSF1というヒトのメス一個体のtechnical replicates」であることは列名や文脈から読み解けます。

```
#in_f <- "http://genome.cshlp.org/content/suppl/2009/12/16/gr.099226.109.DC1/suppTable1.xls">#入力ファイル名を指定してin_fに格納
in_f <- "suppTable1.xls"
out_f <- "sample_blekhman_36.txt" #出力ファイル名を指定してout_fに格納
```

#入力ファイルの読み込み

```
hoge <- read.table(in_f, header=TRUE, row.names=1) #1
```

#サブセットの取得

```
data <- cbind(#2
```

```
hoge$R1L4.HSF1, hoge$R4L2.HSF1, hoge$R2L8.HSF1,
```

```
hoge$R1L1.HSM1, hoge$R5L2.HSM1, hoge$R2L8.HSM1,
```

```
hoge$R1L2.PTF1, hoge$R4L4.PTF1, hoge$R2L8.PTF1,
```

```
hoge$R1L6.PTM1, hoge$R3L3.PTM1, hoge$R2L8.PTM1,
```

```
hoge$R1L7.RMF1, hoge$R5L1.RMF1, hoge$R2L8.RMF1,
```

```
hoge$R1L3.RMM1, hoge$R3L8.RMM1, hoge$R2L8.RMM1)
```

```
colnames(data) <- c(#列名
```

```
"R1L4.HSF1", "R4L2.HSF1", "R2L7.HSF2", "R2L8.HSF3",
```

```
"R1L1.HSM1", "R5L2.HSM1", "R2L3.HSM2", "R2L8.HSM3",
```

```
"R1L2.PTF1", "R4L4.PTF1", "R2L4.PTF2", "R2L8.PTF3",
```

```
"R1L6.PTM1", "R3L3.PTM1", "R2L8.PTM2", "R2L8.PTM3",
```

```
"R1L7.RMF1", "R5L1.RMF1", "R2L2.RMF2", "R2L8.RMF3",
```

R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files(pattern="supp")
[1] "suppTable1.xls"
> file.info("suppTable1.xls")
  size isdir mode mtime
suppTable1.xls 4531819 FALSE 666 2012-08-28 10:38:12
                                         ctime atime
suppTable1.xls 2015-07-04 15:54:57 2015-07-04 15:54:57
               exe
suppTable1.xls no
> |
```

Linuxの場合

```
iu@bielinux[~/Desktop/mac_share] ①
iu@bielinux[mac_share] pwd
/home/iu/Desktop/mac_share [ 9:21午後 ]
iu@bielinux[mac_share] ② ls *supp*
suppTable1.xls [12:05午前]
iu@bielinux[mac_share] ③ ls -la suppTable1.xls
-rwxrwxrwx 1 root root 4531819 8月 28 2012 suppTable1.xls [12:05午前]
iu@bielinux[mac_share] ④
```

```
R Console ①
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
② > list.files(pattern="supp")
[1] "suppTable1.xls"
③ > file.info("suppTable1.xls")
  size isdir mode          mtime
suppTable1.xls 4531819 FALSE 666 2012-08-28 10:38:12
               ctime           atime
suppTable1.xls 2015-07-04 15:54:57 2015-07-04 15:54:57
               exe
suppTable1.xls no
> ④
```

読み込み確認

①黒枠部分をコピーして、読み込んでいることを確認。②コピー時に、③灰色部分は「反転」しないのでコピーできているか不安かもしれないが、ちゃんとコピーできているので気にしない。

```
#in_f <- "http://genome.cshlp.org/content/suppl/2009/12/16/gr.09926.xls"
in_f <- "suppTable1.xls" #入力ファイル名を指定してin_fに格納
out_f <- "sample_blekhman_36.txt" #出力ファイル名を指定してout_fに格納
```

#入力ファイルの読み込み

```
hoge <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="")#in_fで指定したファイルの読み込み
dim(hoge) #行数と列数を表示
```

#サブセットの取得

```
data <- cbind(
  hoge$R1L4.HSF1, hoge$R4L2.HSF1, hoge$R2L7.HSF2, hoge$R3L2.HSF2, hoge$R8L1.HSF3, hoge$R8L2.HSF3,
  hoge$R1L1.HSM1, hoge$R5L2.HSM1, hoge$R2L3.HSM2, hoge$R4L8.HSM2, hoge$R3L6.HSM3, hoge$R4L1.HSM3,
  hoge$R1L2.PTF1, hoge$R4L4.PTF1, hoge$R2L4.PTF2, hoge$R6L6.PTF2, hoge$R3L7.PTF3, hoge$R5L3.PTF3,
  hoge$R1L6.PTM1, hoge$R3L1.RMF1, hoge$R1L7.RMF1, hoge$R5L1.RMF1, hoge$R1L3.RMM1, hoge$R3L8.RMM1,
  colnames(data) <- c(
    "R1L4.HSF1", "R4L2.HSF1",
    "R1L1.HSM1", "R5L2.HSM1",
    "R1L2.PTF1", "R4L4.PTF1",
    "R1L6.PTM1", "R3L3.PTM1",
    "R1L7.RMF1", "R5L1.RMF1"
  )
```

③サブの取得

```
data <- cbind(
  hoge$R1L4.HSF1, hoge$R4L2.HSF1, hoge$R2L7.HSF2, hoge$R3L2.HSF2, hoge$R8L1.HSF3, hoge$R8L2.HSF3,
  hoge$R1L1.HSM1, hoge$R5L2.HSM1, hoge$R2L3.HSM2, hoge$R4L8.HSM2, hoge$R3L6.HSM3, hoge$R4L1.HSM3,
  hoge$R1L2.PTF1, hoge$R4L4.PTF1, hoge$R2L4.PTF2, hoge$R6L6.PTF2, hoge$R3L7.PTF3, hoge$R5L3.PTF3,
  hoge$R1L6.PTM1, hoge$R3L1.RMF1, hoge$R1L7.RMF1, hoge$R5L1.RMF1, hoge$R1L3.RMM1, hoge$R3L8.RMM1,
  colnames(data) <- c(
    "R1L4.HSF1", "R4L2.HSF1", "R2L7.HSF2",
    "R1L1.HSM1", "R5L2.HSM1", "R2L3.HSM2",
    "R1L2.PTF1", "R4L4.PTF1", "R2L4.PTF2",
    "R1L6.PTM1", "R3L3.PTM1", "R2L8.PTM2", "R4L6.PTM2", "R6L2.PTM3", "R6L4.PTM3",
    "R1L7.RMF1", "R5L1.RMF1", "R2L2.RMF2", "R5L8.RMF2", "R3L4.RMF3", "R4L7.RMF3",
  )
```

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

順番で結合した結果をdataに格納

hoge\$R8L1.HSF3, hoge\$R8L2.HSF3,
hoge\$R3L6.HSM3, hoge\$R4L1.HSM3,
hoge\$R3L7.PTF3, hoge\$R5L3.PTF3,
hoge\$R6L2.PTM3, hoge\$R6L4.PTM3,
hoge\$R3L4.RMF3, hoge\$R4L7.RMF3,
hoge\$R3L1.RMM3, hoge\$R4L3.RMM3)

R2L2.HSF2",
R1L1.HSM3",
R2L3.PTF3",
R2L8.PTM2", R4L6.PTM2", R6L2.PTM3",
R6L4.PTM3", R1L7.RMF3", R2L2.RMF2", R5L8.RMF2", R3L4.RMF3", R4L7.RMF3",

読み込み確認

```
#in_f <- "http://genome.cshlp.org/content/suppl/2009/12/16/gr.099226.109.DC1/suppTable1.xls">#入力ファイル名を指定してin_fに格納
in_f <- "suppTable1.xls" #入力ファイル名を指定してin_fに格納
out_f <- "sample_blekhman_36.txt" #出力ファイル名を指定してout_fに格納

#入力ファイルの読み込み
hoge <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="")#in_fで指定したファイルの読み込み
dim(hoge) #行数と列数を表示
```

#サブセットの取得

```
data <- cbind(
  hoge$R1L4.HSF1, hoge$R4L2.HSF1, hoge$R1L1.HSM1, hoge$R5L2.HSM1, hoge$R1L2.PTF1, hoge$R4L4.PTF1, hoge$R1L6.PTM1, hoge$R3L3.PTM1, hoge$R1L7.RMF1, hoge$R5L1.RMF1, hoge$R1L3.RMM1, hoge$R3L8.RMM1, hoge$R2L4.RMF1, hoge$R2L3.RMM1, hoge$R2L8.PTM1, hoge$R2L2.RMF2)
colnames(data) <- c(
  "R1L4.HSF1", "R4L2.HSF1", "R2L7.HSF2", "R1L1.HSM1", "R5L2.HSM1", "R2L3.HSM2", "R1L2.PTF1", "R4L4.PTF1", "R2L4.PTF2", "R1L6.PTM1", "R3L3.PTM1", "R2L8.PTM2", "R1L7.RMF1", "R5L1.RMF1", "R2L2.RMF2")
```

#必要な列名の情報を取得したい列の順番で結合した結果をdataに格納

R R Console

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\$

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
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ます
'help()'とすればオンラインヘルプが出ます
'help.start()'で HTML ブラウザに
'q()'と入力すれば R を終了します。

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



読み込み確認

```
#in_f <- "http://genome.cshlp.org/content/suppl/2009/12/16/gr.0992"
in_f <- "suppTable1.xls"          #入力ファイル名を指定してin_f
out_f <- "sample_blekhman_36.txt" #出力ファイル名を指定してout_f
```

```
#入力ファイルの読み込み
hoge <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="")#in_fで指定したファイルの読み込み
dim(hoge)                  #行数と列数を表示
```

サブセットの取得

```
data <- cbind(
  hoge$R1L4.HSF1, hoge$R4L2.HSF1, hoge$R1L1.HSM1, hoge$R5L2.HSM1,
  hoge$R1L2.PTF1, hoge$R4L4.PTF1, hoge$R1L6.PTM1, hoge$R3L3.PTM1,
  hoge$R1L7.RMF1, hoge$R5L1.RMF1, hoge$R1L3.RMM1, hoge$R3L8.RMM1,
  colnames(data) <- c(
    "R1L4.HSF1", "R4L2.HSF1", "R2L7.F",
    "R1L1.HSM1", "R5L2.HSM1", "R2L3.F",
    "R1L2.PTF1", "R4L4.PTF1", "R2L4.F",
    "R1L6.PTM1", "R3L3.PTM1", "R2L8.F",
    "R1L7.RMF1", "R5L1.RMF1", "R2L2.F")
```

read.table関数を用いてsuppTable1.xlsを読み込む際、①ヘッダー行あり(header=T)として、また②(行名として用いるため)1列目を行名(row.names=1)としている。このため、残りのデータは①20,689行×55列となる。



R Console

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

> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> #in_f <- "http://genome.cshlp.org/content/suppl/2009/12/16/gr$"
> in_f <- "suppTable1.xls"          #入力ファイル名を指定$
> out_f <- "sample_blekhman_36.txt" #出力ファイル名を指定$
>
> #入力ファイルの読み込み
> hoge <- read.table(in_f, header=TRUE, row.names=1, sep="\t", $           #行数と列数を表示
[1] 20689      55
> |
```

suppTable1.xls

確かに入力ファイル(suppTable1.xls)は、①の幅的にも55列くらいありそぐだと納得できる。また、②2列目以降からすぐにカウントデータになっているわけではないこともわかる。

The screenshot shows an Excel spreadsheet titled "suppTable1.xls" with a table of gene information. The table has columns labeled A through J. Column A contains Ensembl Gene IDs, and columns B through J contain various gene-related metrics. A red arrow labeled ② points to the header row. Another red arrow labeled ① points to the bottom right corner of the data range, where the formula bar shows the formula for the last cell.

A	B	C	D	E	F	G	H	I	J
1 EnsemblGeneID	GeneSymbol	numExons	deHC	deHR	deCR	deHC.m	deHC.f	deHR.m	deHR.f
2 ENSG000000000003	TSPAN6		7	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
3 ENSG000000000005	TNMD		7	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
4 ENSG000000000419	DPM1		6	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
5 ENSG000000000457	SCYL3		13	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
6 ENSG000000000460	C1orf112		22	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE
7 ENSG000000000938	FGR		12	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

Below the table, the Excel status bar shows "準備完了". In the bottom-left corner of the image, there is a red arrow pointing to the R code window.

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> in_f <- "http://genome.cshlp.org/content/suppl/2009/12/16/gr$"
> in_f <- "suppTable1.xls"                      #入力ファイル名を指定$
> out_f <- "sample_blekhman_36.txt"          #出力ファイル名を指定$
>
> #入力ファイルの読み込み
> hoge <- read.table(in_f, header=TRUE, row.names=1, sep="\t", $           #行数と列数を表示
> dim(hoge)
[1] 20689      55
> |
```

suppTable1.xls

行列の一部を抽出して表示。行列hogeの①1-7行目、および②1-6列目を抽出して表示。こんな感じでうまく読み込んでいることを確認する。

suppTable1.xls - Excel

	A	B	C	D	E	F	G	H	I	J
1	EnsemblGeneID	GeneSymbol	numExons	deHC	deHR	deCR	deHC.m	deHC.f	deHR.m	deHR
2	ENSG000000000003	TSPAN6		7	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
3	ENSG000000000005	TNMD		7	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
4	ENSG000000000419	DPM1		6	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
5	ENSG000000000457	SCYL3		13	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
6	ENSG000000000460	Clorf112		22	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE
7	ENSG000000000938	FGR		12	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

suppTable1

R Console

```

① > hoge[1:7, 1:6]
②
  GeneSymbol numExons deHC deHR deCR deHC.m
  ENSG000000000003 TSPAN6      7 TRUE FALSE FALSE FALSE
  ENSG000000000005 TNMD       7 FALSE FALSE FALSE FALSE
  ENSG000000000419 DPM1       6 FALSE FALSE FALSE FALSE
  ENSG000000000457 SCYL3      13 FALSE FALSE TRUE FALSE
  ENSG000000000460 Clorf112    22 FALSE TRUE TRUE FALSE
  ENSG000000000938 FGR        12 FALSE FALSE FALSE FALSE
  ENSG000000000971 CFH        16 FALSE TRUE TRUE FALSE
>

```

head関数

①head関数を用いて、最初の1行分のみ表示。55列分もあるので、1行だけ表示させることでも結構な画面サイズを要する。

EnsemblGeneID	GeneSymbol	numExons	deHC	deHR	deCR	deHC.m	deHC.f	deHR.m	deHR.f	deCR.m
ENSG000000000003	TSPAN6	7	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
ENSG000000000005	TNMD	7	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
ENSG000000000419	DRM1	6	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

R Console

①

```
> head(hoge, n=1)
      GeneSymbol numExons deHC deHR deCR deHC.m deHC.f deHR.m deHR.f deCR.m
ENSG000000000003    TSPAN6      7 TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
                     deCR.f Stabilizing DirectionalHuman DirectionalChimp SexDimorphic
ENSG000000000003    FALSE      FALSE      TRUE      FALSE      FALSE
                     SexDimorphic.HighMales SexDimorphic.HighFemales ExonDimorphic
ENSG000000000003
                     FALSE      FALSE      FALSE
                     HumanSpecificExonUsage R1L1.HSM1 R1L2.PTF1 R1L3.RMM1 R1L4.HSF1 R1L6.PTM1
ENSG000000000003
                     FALSE      60       285      207      172      176
                     R1L7.RMF1 R2L2.RMF2 R2L3.HSM2 R2L4.PTF2 R2L6.RMM2 R2L7.HSF2 R2L8.PTM2
ENSG000000000003    259      299      219      213      676      147      316
                     R3L1.RMM3 R3L2.HSF2 R3L3.PTM1 R3L4.RMF3 R3L6.HSM3 R3L7.PTF3 R3L8.RMM1
ENSG000000000003    338      153      233      242      199      180      217
                     R4L1.HSM3 R4L2.HSF1 R4L3.RMM3 R4L4.PTF1 R4L6.PTM2 R4L7.RMF3 R4L8.HSM2
ENSG000000000003    160      157      367      289      369      238      202
                     R5L1.RMF1 R5L2.HSM1 R5L3.PTF3 R5L4.RMM2 R5L8.RMF2 R6L2.PTM3 R6L4.PTM3
ENSG000000000003    252      61       206      672      165      216      212
                     R6L6.PTF2 R8L1.HSF3 R8L2.HSF3
ENSG000000000003    216      78       90
>
```

suppTable1.xls

①別の表現方法。黒枠の列以降が目的のカウント情報であることが読み取れる。これはIlluminaのRNA-seqカウントデータ。Illuminaは実験単位をラン(Run; R)で表現する。また1つのRun中に複数のレーン(Lane; L)があるので複数サンプルを流せる。それゆえR1L1.HSM1は、Run1のLane1に流したHSM1というサンプルのカウントデータと読み取る。

EnsemblGeneID	GeneSymbol	numExons	deHC	deHR	deCR	deHC.m	deHC.f	deHR.m	deHR.f	deCR.m
ENSG000000000003	TSPAN6	7	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
ENSG000000000005	TNMD	7	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
ENSG000000000049	DPM1	6	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
R Console										
①										
> hoge[1,]										
ENSG000000000003	TSPAN6	7	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
			deCR.f	Stabilizing	Directional	Human	Directional	Chimp	SexDimorphic	
ENSG000000000003			FALSE	FALSE		TRUE		FALSE		FALSE
			SexDimorphic	.HighMales	SexDimorphic	.HighFemales	ExonDimorphic			
ENSG000000000003					FALSE		FALSE		FALSE	
			HumanSpecificExonUsage	R1L1.HSM1	R1L2.PTF1	R1L3.RMM1	R1L4.HSF1	R1L6.PTM1		
ENSG000000000003				FALSE	60		285	207	172	176
			R1L7.RMF1	R2L2.RMF2	R2L3.HSM2	R2L4.PTF2	R2L6.RMM2	R2L7.HSF2	R2L8.PTM2	
ENSG000000000003		259	299	219	213	676	147	316		
	R3L1.RMM3	R3L2.HSF2	R3L3.PTM1	R3L4.RMF3	R3L6.HSM3	R3L7.PTF3	R3L8.RMM1			
ENSG000000000003	338	153	233	242	199	180	217			
	R4L1.HSM3	R4L2.HSF1	R4L3.RMM3	R4L4.PTF1	R4L6.PTM2	R4L7.RMF3	R4L8.HSM2			
ENSG000000000003	160	157	367	289	369	238	202			
	R5L1.RMF1	R5L2.HSM1	R5L3.PTF3	R5L4.RMM2	R5L8.RMF2	R6L2.PTM3	R6L4.PTM3			
ENSG000000000003	252	61	206	672	165	216	212			
	R6L6.PTF2	R8L1.HSF3	R8L2.HSF3							
ENSG000000000003	216	78	90							
>										

suppTable1.xls

ヒト(HS)

- オス3匹(M1, M2, **M3**)
- メス3匹(F1, F2, F3)

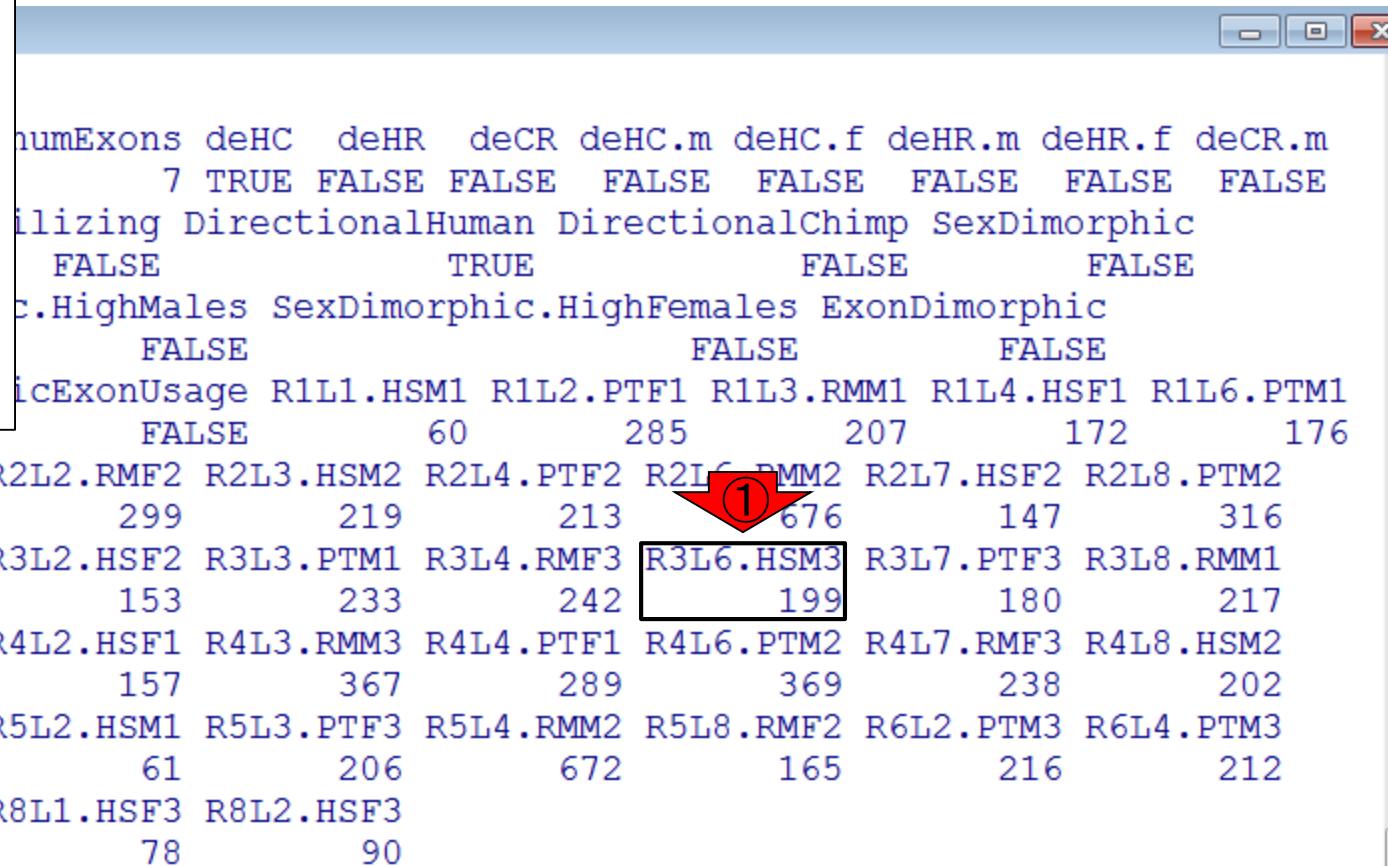
チンパンジー(PT)

- オス3匹(M1, M2, M3)
- メス3匹(F1, F2, F3)

アカゲザル(RM)

- オス3匹(M1, M2, M3)
- メス3匹(F1, F2, F3)

このデータは、3種類の生物種間比較。ヒト(*Homo sapiens*; HS)、チンパンジー(*Pan troglodytes*; PT)、アカゲザル(*Rhesus macaque*; RM)。生物種ごとにオス3匹、メス3匹。雄雌を考慮しなければbiological replicates (生物学的な反復)は6。①黒枠はヒトのオスで、個体識別番号が3のデータ(HSM3)と解釈する。



	numExons	deHC	deHR	deCR	deHC.m	deHR.m	deHR.f	deCR.m
	7	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
	ilizing	Directional	Human	Directional	Chimp	SexDimorphic		
	FALSE		TRUE		FALSE		FALSE	
	c.HighMales	SexDimorphic.	HighFemales	ExonDimorphic				
	FALSE			FALSE			FALSE	
	icExonUsage	R1L1.HSM1	R1L2.PTF1	R1L3.RMM1	R1L4.HSF1	R1L6.PTM1		
	FALSE	60	285	207	172	176		
ENSG000000000003	R1L7.RMF1	R2L2.RMF2	R2L3.HSM2	R2L4.PTF2	R2L5.RMM2	R2L7.HSF2	R2L8.PTM2	
ENSG000000000003	259	299	219	213	676	147	316	
	R3L1.RMM3	R3L2.HSF2	R3L3.PTM1	R3L4.RMF3	R3L6.HSM3	R3L7.PTF3	R3L8.RMM1	
ENSG000000000003	338	153	233	242	199	180	217	
	R4L1.HSM3	R4L2.HSF1	R4L3.RMM3	R4L4.PTF1	R4L6.PTM2	R4L7.RMF3	R4L8.HSM2	
ENSG000000000003	160	157	367	289	369	238	202	
	R5L1.RMF1	R5L2.HSM1	R5L3.PTF3	R5L4.RMM2	R5L8.RMF2	R6L2.PTM3	R6L4.PTM3	
ENSG000000000003	252	61	206	672	165	216	212	
	R6L6.PTF2	R8L1.HSF3	R8L2.HSF3					
ENSG000000000003	216	78	90					

> |

suppTable1.xls

ヒト(HS)

- オス3匹(M1, M2, **M3**)
- メス3匹(F1, F2, F3)

チンパンジー(PT)

- オス3匹(M1, M2, M3)
- メス3匹(F1, F2, F3)

アカゲザル(RM)

- オス3匹(M1, M2, M3)
- メス3匹(F1, F2, F3)

ENSG000000000003

R1L1.RMF1 R1L2.RMF1 R1L3.RMF1 R1L4.RMF1 R1L5.RMF1 R1L6.RMF1

ENSG000000000003

259 299 219 213 676 147 316

R2L1.RMF1 R2L2.RMF1 R2L3.RMF1 R2L4.RMF1 R2L5.RMF1 R2L6.RMF1 R2L7.RMF1 R2L8.RMF1

ENSG000000000003

338 153 233 242 199 180 217

R3L1.RMF1 R3L2.RMF1 R3L3.RMF1 R3L4.RMF1 R3L5.RMF1 R3L6.RMF1 R3L7.RMF1 R3L8.RMF1

ENSG000000000003

160 157 367 289 369 238 202

R4L1.RMF1 R4L2.RMF1 R4L3.RMF1 R4L4.RMF1 R4L5.RMF1 R4L6.RMF1 R4L7.RMF1 R4L8.RMF1

ENSG000000000003

252 61 206 672 165 216 212

R5L1.RMF1 R5L2.RMF1 R5L3.RMF1 R5L4.RMF1 R5L5.RMF1 R5L6.RMF1 R5L7.RMF1 R5L8.RMF1

ENSG000000000003

216 78 90

よく見ると、①Run3のLane6で流したHSM3 (i.e., R3L6.HSM3)以外にも、②Run4のLane1で流した同じHSM3のデータ(i.e., R4L1.HSM3)が存在する。これらは、同一個体由来データである。つまり、technical replicates (技術的な反復)は2である。

	numExons	deHC	deHR	deCR	deHC.m	deHR.m	deHR.f	deCR.m
	7	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
	ilizing	DirectionalHuman	DirectionalChimp	SexDimorphic				
	FALSE	TRUE	FALSE	FALSE				
	c.HighMales	SexDimorphic.HighFemales	ExonDimorphic					
	FALSE	FALSE	FALSE					
	icExonUsage	R1L1.HSM1	R1L2.PTF1	R1L3.RMM1	R1L4.HSF1	R1L6.PTM1		
	FALSE	60	285	207	172	176		
ENSG000000000003	R1L7.RMF1	R2L2.RMF1	R2L3.HSM2	R2L4.PTF2	R2L5.RMM2	R2L7.HSF2	R2L8.PTM2	
ENSG000000000003	259	299	219	213	676	147	316	
	R3L1.RMM3	R3L2.HSF2	R3L3.PTM1	R3L4.RMF3	R3L6.HSM3	R3L7.PTF3	R3L8.RMM1	
ENSG000000000003	338	153	233	242	199	180	217	
	R4L1.HSM3	R4L2.HSF1	R4L3.RMM3	R4L4.PTF1	R4L6.PTM2	R4L7.RMF3	R4L8.HSM2	
ENSG000000000003	160	157	367	289	369	238	202	
	R5L1.RMF1	R5L2.HSM1	R5L3.PTF3	R5L4.RMM2	R5L8.RMF2	R6L2.PTM3	R6L4.PTM3	
ENSG000000000003	252	61	206	672	165	216	212	
	R6L6.PTF2	R8L1.HSF3	R8L2.HSF3					
ENSG000000000003	216	78	90					

>

suppTable1.xls

個体ごとにサンプルを分割して得たデータが全個体について存在する。technical replicates (技術的な反復)は2。

ヒト(HS)

- オス3匹(M1, M2, **M3**)
- メス3匹(F1, F2, F3)

チンパンジー(PT)

- オス3匹(M1, M2, M3)
- メス3匹(F1, **F2**, F3)

アカゲザル(RM)

- オス3匹(M1, M2, M3)
- メス3匹(**F1**, F2, F3)

ENSG000000000003

	R1L7.RMF1	R2L2.RMF2	R2L3.HSM2	R2L4.PTF2	R2L6.RMM2	R2L7.HSF2	R2L8.PTM2
ENSG000000000003	259	299	219	213	676	147	316
	R3L1.RMM3	R3L2.HSF2	R3L3.PTM1	R3L4.RMF3	R3L6.HSM3	R3L7.PTF3	R3L8.RMM1
ENSG000000000003	338	153	233	242	199	180	217
	R4L1.HSM3	R4L2.HSF1	R4L3.RMM3	R4L4.PTF1	R4L6.PTM2	R4L7.RMF3	R4L8.HSM2
ENSG000000000003	160	157	367	289	369	238	202
	R5L1.RMF1	R5L2.HSM1	R5L3.PTF3	R5L4.RMM2	R5L8.RMF2	R6L2.PTM3	R6L4.PTM3
ENSG000000000003	252	61	206	672	165	216	212
	R6L6.PTF2	R8L1.HSF3	R8L2.HSF3				
ENSG000000000003	216	78	90				

> |

colnames関数

行列hogeの①列名を表示。目的のカウントデータが20列目以降にあることが分かる。

ヒト(HS)

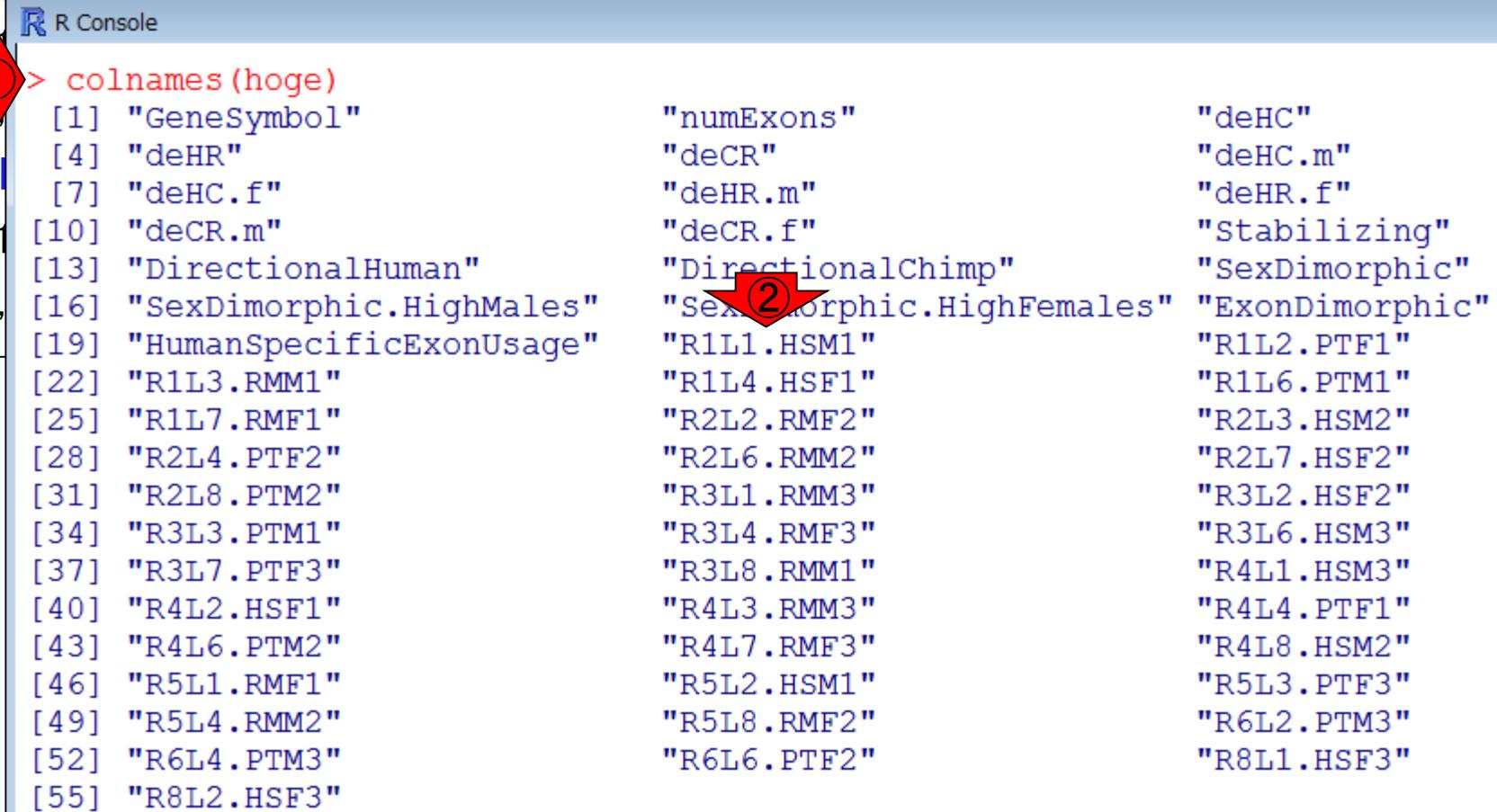
- オス3匹(M1, M2, **M3**)
- メス3匹(F1, F2, F3)

チンパンジー(PT)

- オス3匹(M1, M2, **M3**)
- メス3匹(F1, F2, **F3**)

アカゲザル(RI)

- オス3匹(M1, M2, **M3**)
- メス3匹(F1, F2, **F3**)



```
R Console
> colnames(hoge)
[1] "GeneSymbol"          "numExons"           "deHC"
[4] "deHR"                "deCR"               "deHC.m"
[7] "deHC.f"              "deHR.m"             "deHR.f"
[10] "deCR.m"              "deCR.f"              "Stabilizing"
[13] "DirectionalHuman"    "DirectionalChimp"   "SexDimorphic"
[16] "SexDimorphic.HighMales" "SexDimorphic.HighFemales" "ExonDimorphic"
[19] "HumanSpecificExonUsage" "R1L1.HSM1"           "R1L2.PTF1"
[22] "R1L3.RMM1"            "R1L4.HSF1"           "R1L6.PTM1"
[25] "R1L7.RMF1"            "R2L2.RMF2"           "R2L3.HSM2"
[28] "R2L4.PTF2"             "R2L6.RMM2"           "R2L7.HSF2"
[31] "R2L8.PTM2"             "R3L1.RMM3"           "R3L2.HSF2"
[34] "R3L3.PTM1"             "R3L4.RMF3"           "R3L6.HSM3"
[37] "R3L7.PTF3"             "R3L8.RMM1"           "R4L1.HSM3"
[40] "R4L2.HSF1"             "R4L3.RMM3"           "R4L4.PTF1"
[43] "R4L6.PTM2"             "R4L7.RMF3"           "R4L8.HSM2"
[46] "R5L1.RMF1"             "R5L2.HSM1"           "R5L3.PTF3"
[49] "R5L4.RMM2"             "R5L8.RMF2"           "R6L2.PTM3"
[52] "R6L4.PTM3"             "R6L6.PTF2"           "R8L1.HSF3"
[55] "R8L2.HSF3"
```

lengthは要素数

ヒト(HS)

- オス3匹(M1, M2, **M3**)
- メス3匹(F1, F2, **F3**)

チンパンジー(PT)

- オス3匹(M1, M2)
- メス3匹(F1, F2,

アカゲザル(RM)

- オス3匹(M1, M2)
- メス3匹(F1, F2)

①



```
R Console
```

```
[31] "R2L8.PTM2" "R3L1.RMM3" "R3L2.HSF2"
[34] "R3L3.PTM1" "R3L4.RMF3" "R3L6.HSM3"
[37] "R3L7.PTF3" "R3L8.RMM1" "R4L1.HSM3"
[40] "R4L2.HSF1" "R4L3.RMM3" "R4L4.PTF1"
[43] "R4L6.PTM2" "R4L7.RMF3" "R4L8.HSM2"
[46] "R5L1.RMF1" "R5L2.HSM1" "R5L3.PTF3"
[49] "R5L4.RMM2" "R5L8.RMF2" "R6L2.PTM3"
[52] "R6L4.PTM3" "R6L6.PTF2" "R8L1.HSF3"
[55] "R8L2.HSF3"

> colnames(hoge) [20:55]
[1] "R1L1.HSM1" "R1L2.PTF1" "R1L3.RMM1" "R1L4.HSF1" "R1L6.PTM1"
[6] "R1L7.RMF1" "R2L2.RMF2" "R2L3.HSM2" "R2L4.PTF2" "R2L6.RMM2"
[11] "R2L7.HSF2" "R2L8.PTM2" "R3L1.RMM3" "R3L2.HSF2" "R3L3.PTM1"
[16] "R3L4.RMF3" "R3L6.HSM3" "R3L7.PTF3" "R3L8.RMM1" "R4L1.HSM3"
[21] "R4L2.HSF1" "R4L3.RMM3" "R4L4.PTF1" "R4L6.PTM2" "R4L7.RMF3"
[26] "R4L8.HSM2" "R5L1.RMF1" "R5L2.HSM1" "R5L3.PTF3" "R5L4.RMM2"
[31] "R5L8.RMF2" "R6L2.PTM3" "R6L4.PTM3" "R6L6.PTF2" "R8L1.HSF3"
[36] "R8L2.HSF3"

> length(colnames(hoge))
[1] 55

> length(colnames(hoge) [20:55])
[1] 36
```

②



③



①行列hogeの列名の20-55番目の要素のみを表示。(55 - 20 +1)=36個の要素数と手計算できるが、length関数を用いて、②オリジナルが55個の要素、③サブセットの要素数が36個という結果を得ることもできる。lengthは、要素数分だけループを回したりする際にも用いられる。

列の並びがマイチ

ヒト(HS)

- オス3匹(M1, M2, **M3**)
- メス3匹(F1, F2, F3)

チンパンジー(PT)

- オス3匹(M1, M2, M3)
- メス3匹(F1, F2, F3)

アカゲザル(RM)

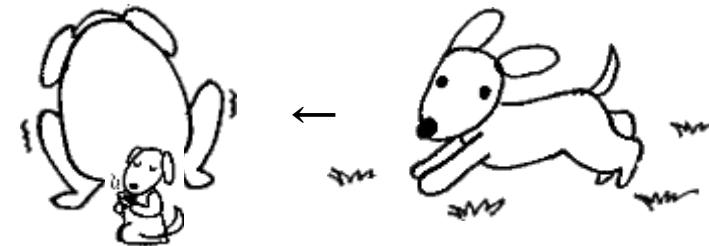
- オス3匹(M1, M2, M3)
- メス3匹(F1, F2, F3)

①行列hoge中の20–55番目の列を抽出した結果をdataに格納。これがsubsettingの基本形。②行列dataの最初の1行目のみ表示。うまく抽出できていることがわかる。③しかしよく見ると、生物種ごとのようなきれいな並びになっていないので、マイチ。

R Console

```
> dim(hoge)
[1] 20689      55
> data <- hoge[, 20:55]
> dim(data)
[1] 20689      36
> data[1, ]
R1L1.HSM1  R1L2.PTF1  R1L3.RMM1  R1L4.HSF1  R1L6.PTM1
ENSG000000000003          60        285        207        172        176
R1L7.RMF1  R2L2.RMF2  R2L3.HSM2  R2L4.PTF2  R2L6.RMM2
ENSG000000000003          259        299        219        213        676
R2L7.HSF2  R2L8.PTM2  R3L1.RMM3  R3L2.HSF2  R3L3.PTM1
ENSG000000000003          147        316        338        153        233
R3L4.RMF3  R3L6.HSM3  R3L7.PTF3  R3L8.RMM1  R4L1.HSM3
ENSG000000000003          242        199        180        217        160
R4L2.HSF1  R4L3.RMM3  R4L4.PTF1  R4L6.PTM2  R4L7.RMF3
ENSG000000000003          157        367        289        369        238
R4L8.HSM2  R5L1.RMF1  R5L2.HSM1  R5L3.PTF3  R5L4.RMM2
ENSG000000000003          202        252         61        206        672
R5L8.RMF2  R6L2.PTM3  R6L4.PTM3  R6L6.PTF2  R8L1.HSF3
ENSG000000000003          165        216        212        216         78
R8L2.HSF3
ENSG000000000003              90
```

嘘のようなホントの話



- ヒト(HS)
 - オス3匹(M1, M2, **M3**)
 - メス3匹(F1, F2, F3)
- チンパンジー(PT)
 - オス3匹(M1, M2, M3)
 - メス3匹(F1, F2, F3)
- アカゲザル(RM)
 - オス3匹(M1, M2, M3)
 - メス3匹(F1, F2, F3)

R Console

```

> dim(hoge)
[1] 20689    55
> data <- hoge[, 20:55]
> dim(data)
[1] 20689    36
> data[1, ]

```

	R1L1.HSM1	R1L2.PTF1	R1L3.RMM1	R1L4.HSF1	R1L6.PTM1
ENSG00000000003	60	285	207	172	176
	R1L7.RMF1	R2L2.RMF2	R2L3.HSM2	R2L4.PTF2	R2L6.RMM2
ENSG00000000003	259	299	219	213	676
	R2L7.HSF2	R2L8.PTM2	R3L1.RMM3	R3L2.HSF2	R3L3.PTM1
ENSG00000000003	147	316	338	153	233
1 矢印	3L6.HSM3	R3L7.PTF3	R3L8.RMM1	R4L1.HSM3	
ENSG	2 →		199	180	217
	3 ↑				160
ENSG	4 ↗	4L3.RMM3	R4L4.PTF1	R4L6.PTM2	R4L7.RMF3
		367	289	369	238
ENSG	5 ←	5L1.RMF1	R5L2.HSM1	R5L3.PTF3	R5L4.RMM2
	6 ↓	252	61	206	672
ENSG	7 ⇒	6L2.PTM3	R6L4.PTM3	R6L6.PTF2	R8L1.HSF3
	8 ⇔	216	212	216	78
ENSG	9 やじるし				

環境依存文字(unicode)

```

> data

```

①

header=TRUEとしたおかげで、data\$列名、hoge\$列名などとして特定の列のみ取り扱える。

列名で並び替え

- ヒト(HS)
 - オス3匹(M1, M2, M3)
 - メス3匹(F1, F2, F3)

- チンパンジー(PT)
 - オス3匹(M1, M2, M3)
 - メス3匹(F1, F2, F3)

- アカゲザル(RM)
 - オス3匹(M1, M2, M3)
 - メス3匹(F1, F2, F3)

```
R Console
> data[1, ]
          R1L1.HSM1  R1L2.PTF1  R1L3.RMM1  R1L4.HSF1  R1L6.PTM1
ENSG000000000003       60        285        207        172        176
                           R1L7.RMF1  R2L2.RMF2  R2L3.HSM2  R2L4.PTF2  R2L6.RMM2
ENSG000000000003     259        299        219        213        676
                           R2L7.HSF2  R2L8.PTM2  R3L1.RMM3  R3L2.HSF2  R3L3.PTM1
ENSG000000000003     147        316        338        153        233
                           R3L4.RMF3  R3L6.HSM3  R3L7.PTF3  R3L8.RMM1  R4L1.HSM3
ENSG000000000003     242        199        180        217        160
                           R4L2.HSF1  R4L3.RMM3  R4L4.PTF1  R4L6.PTM2  R4L7.RMF3
ENSG000000000003     157        367        289        369        238
                           R4L8.HSM2  R5L1.RMF1  R5L2.HSM1  R5L3.PTF3  R5L4.RMM2
ENSG000000000003     202        252        61        206        672
                           R5L8.RMF2  R6L2.PTM3  R6L4.PTM3  R6L6.PTF2  R8L1.HSF3
ENSG000000000003     165        216        212        216        78
                           R8L2.HSF3
ENSG000000000003         90
> data$R1L1.HSM1[1:5]
[1] 60 0 17 50 9
> data$R5L2.HSM1[1:5]
[1] 61 0 22 64 6
```

cbind関数

- ヒト(HS)
 - オス3匹(M1, M2, **M3**)
 - メス3匹(F1, F2, F3)
- チンパンジー(PT)
 - オス3匹(M1, M2, M3)
 - メス3匹(F1, F2, F3)
- アカゲザル(RM)
 - オス3匹(M1, M2, M3)
 - メス3匹(F1, F2, F3)

任意のベクトル同士を列(column)方向で結合(bind)するのがcbind関数。①列を単純に結合することができる。



R Console

```
> data$R1L1.HSM1[1:5]
[1] 60 0 17 50 9
> data$R5L2.HSM1[1:5]
[1] 61 0 22 64 6
> cbind(data$R1L1.HSM1[1:5], data$R5L2.HSM1[1:5])
     [,1] [,2]
[1,]    60   61
[2,]      0   0
[3,]    17   22
[4,]    50   64
[5,]      9    6
> |
```

cbind関数

ヒト(HS)

- オス3匹(M1, M2, **M3**)
- メス3匹(F1, F2, F3)

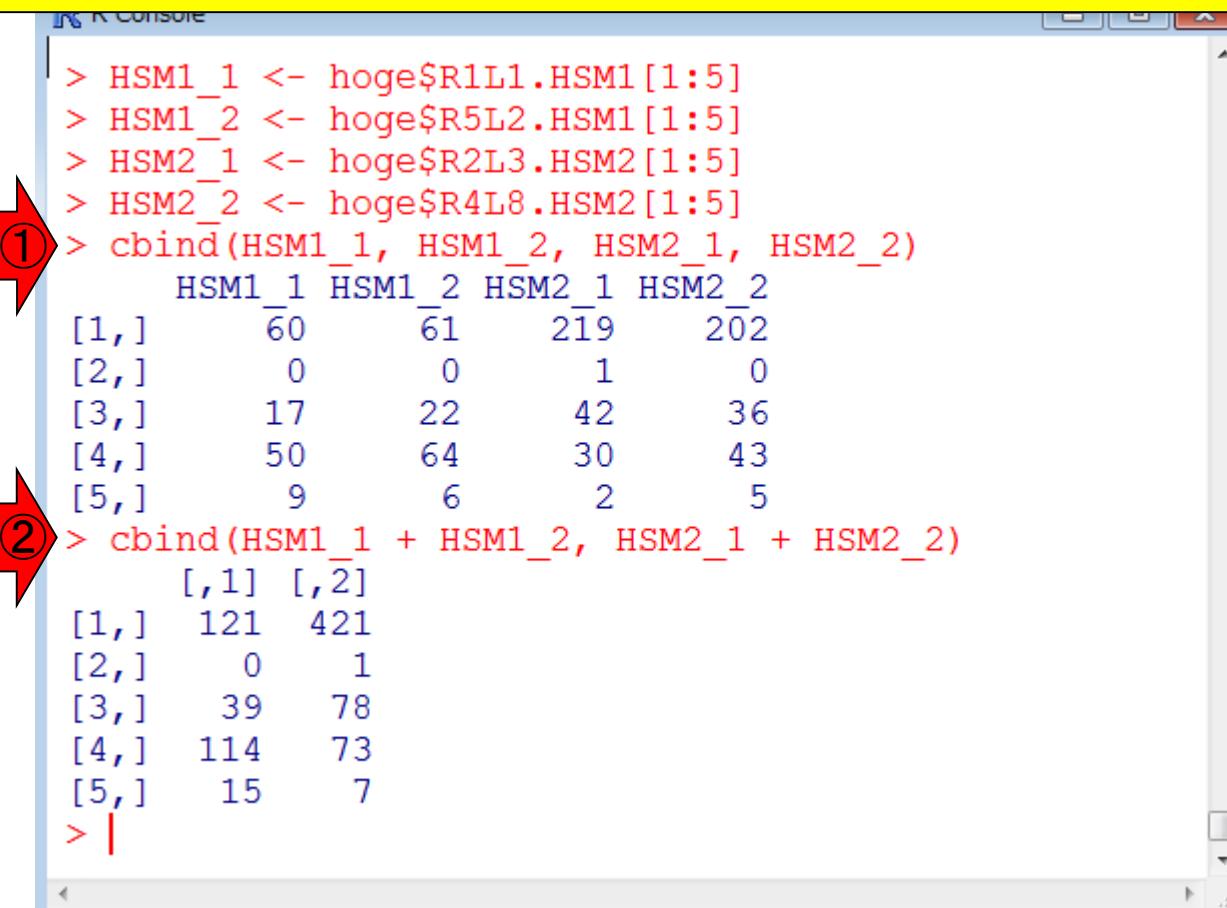
チンパンジー(PT)

- オス3匹(M1, M2, M3)
- メス3匹(F1, F2, F3)

アカゲザル(RM)

- オス3匹(M1, M2, M3)
- メス3匹(F1, F2, F3)

こんな書き方もできる。①の拡張版として「HSM1, HSM2, HSM3, HSF1, …, RMF2, RMF3」のような並びにしておけば、後の発現変動解析のときにいろいろと便利。②同一個体の反復データを足す場合。これはtechnical replicatesデータをマージ(合併)させることに相当する。一般的な発現変動解析は、technical replicatesデータをマージして、biological replicatesのみからなるデータにしたものを入力として行う。



```
R> K:Console
> HSM1_1 <- hoge$R1L1.HSM1[1:5]
> HSM1_2 <- hoge$R5L2.HSM1[1:5]
> HSM2_1 <- hoge$R2L3.HSM2[1:5]
> HSM2_2 <- hoge$R4L8.HSM2[1:5]
> cbind(HSM1_1, HSM1_2, HSM2_1, HSM2_2)
      HSM1_1 HSM1_2 HSM2_1 HSM2_2
[1,]     60     61    219    202
[2,]      0      0      1      0
[3,]     17     22     42     36
[4,]     50     64     30     43
[5,]      9      6      2      5
> cbind(HSM1_1 + HSM1_2, HSM2_1 + HSM2_2)
      [,1] [,2]
[1,] 121  421
[2,]  0   1
[3,] 39   78
[4,] 114  73
[5,] 15   7
> |
```

元データを整形

ここまで説明で、例題41の下記コードの中身がかなり理解できるはずです

- ・(削除予定)個別パッケージのインストール (last modified 2015/02/20)

- ・基本的な利用法 (last modified 2015/04/03)

- ・サンプルデータ (last modified 2015/06/15) NEW

- ・バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ)|NGSハンドブック

- ・バイオハンドブック

- ・書籍

- ・書籍

- ・書籍

サンプルデータ NEW

1. 41. Blekhman et al., Genome Res., 2010のリアルカウントデータです。Supplementary Table1で提供されているエクセルファイル (<http://genome.cshlp.org/content/suppl/2009/12/16/gr.099226.109.DC1/suppTable1.xls>; 約4.3MB) からカウントデータのみ抽出し、きれいに整形しなおしたものがここでの出力ファイルになります。20,689 genes×36 samplesのカウントデータ([sample_blekhman_36.txt](#))です。実験デザインの詳細はFigure S1中に描かれていますが、ヒト(Homo Sapiens; HS), チンパンジー(Pan troglodytes; PT), アカゲザル(Rhesus macaque; RM)の3種類の生物種の肝臓サンプル(liver sample)の比較を行っています。生物種ごとにオス3個体メス3個体の計6個体使われており(six individuals; six biological replicates)、技術的なばらつき(technical variation)を見積もるべく各個体は2つに分割されてデータが取得されています(duplicates; two technical replicates)。それゆえ、ヒト12サンプル、チンパンジー12サンプル、アカゲザル12サンプルの計36サンプル分のデータということになります。以下で行っていることはカウントデータの列のみ「ヒトのメス(HSF1, HSF2, HSF3)」「ヒトのオス(HSM1, HSM2, HSM3)」「チンパンジーのメス(PTF1, PTF2, PTF3)」「チンパンジーのオス(PTM1, PTM2, PTM3)」「アカゲザルのメス(RMF1, RMF2, RMF3)」「アカゲザルのオス(RMM1, RMM2, RMM3)」の順番で並び替えたものをファイルに保存しています。もう少し美しくやることも原理的には可能ですが、そこは本質的な部分ではありませんので、ここではアドホック(その場しのぎ、の意味)な手順で行っています。当然ながら、エクセルなどでファイルの中身を眺めて完全に列名を把握しているという前提です。尚、「R1L4.HSF1」と「R4L2.HSF1」が「HSF1」というヒトのメス一個体のtechnical replicatesであることは列名や文脈から読み解けます。

```
#in_f <- "http://genome.cshlp.org/content/suppl/2009/12/16/gr.099226.109.DC1/suppTable1.xls">#入力ファイル名を指定してin_fに格納
in_f <- "suppTable1.xls"
out_f <- "sample_blekhman_36.txt" #出力ファイル名を指定してout_fに格納
```

#入力ファイルの読み込み

```
hoge <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="")#in_fで指定したファイルの読み込み
dim(hoge) #行数と列数を表示
```

#サブセットの取得

```
data <- cbind(
  hoge$R1L4.HSF1, hoge$R4L2.HSF1, hoge$R2L7.HSF2, hoge$R3L2.HSF2, hoge$R8L1.HSF3, hoge$R8L2.HSF3,
  hoge$R1L1.HSM1, hoge$R5L2.HSM1, hoge$R2L3.HSM2, hoge$R4L8.HSM2, hoge$R3L6.HSM3, hoge$R4L1.HSM3,
  hoge$R1L2.PTF1, hoge$R4L4.PTF1, hoge$R2L4.PTF2, hoge$R6L6.PTF2, hoge$R3L7.PTF3, hoge$R5L3.PTF3,
  hoge$R1L6.PTM1, hoge$R3L3.PTM1, hoge$R2L8.PTM2, hoge$R4L6.PTM2, hoge$R6L2.PTM3, hoge$R6L4.PTM3,
  hoge$R1L7.RMF1, hoge$R5L1.RMF1, hoge$R2L2.RMF2, hoge$R5L8.RMF2, hoge$R3L4.RMF3, hoge$R4L7.RMF3,
  hoge$R1L3.RMM1, hoge$R3L8.RMM1, hoge$R2L6.RMM2, hoge$R5L4.RMM2, hoge$R3L1.RMM3, hoge$R4L3.RMM3)
```

元データを整形

例題41をkoppe。Internet Explorer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。

```
#in_f <- "http://genome.cshlp.org/content/suppl/2009/12/16/gr.099226.109.DC1/suppTable1.xls" #入力ファイル名を指定してin_fに格納
in_f <- "suppTable1.xls"
out_f <- "sample_blekhman_36.txt" #出力ファイル名を指定してout_fに格納

#入力ファイルの読み込み
hoge <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="") #in_fで指定したファイルの読み込み
dim(hoge) #行数と列数を表示

#サブセットの取得
data <- cbind(
  hoge$R1L4.HSF1, hoge$R4L2.HSF1, hoge$R2L7.HSF2, hoge$R3L2.HSF2, hoge$R8L1.HSF3, hoge$R8L2.HSF3,
  hoge$R1L1.HSM1, hoge$R5L2.HSM1, hoge$R2L3.HSM2, hoge$R4L8.HSM2, hoge$R3L6.HSM3, hoge$R4L1.HSM3,
  hoge$R1L2.PTF1, hoge$R4L4.PTF1, hoge$R2L4.PTF2, hoge$R6L6.PTF2, hoge$R3L7.PTF3, hoge$R5L3.PTF3,
  hoge$R1L6.PTM1, hoge$R3L3.PTM1, hoge$R2L8.PTM2, hoge$R4L6.PTM2, hoge$R6L2.PTM3, hoge$R6L4.PTM3,
  hoge$R1L7.RMF1, hoge$R5L1.RMF1, hoge$R2L2.RMF2, hoge$R5L8.RMF2, hoge$R3L4.RMF3, hoge$R4L7.RMF3,
  hoge$R1L3.RMM1, hoge$R3L8.RMM1, hoge$R2L6.RMM2, hoge$R5L4.RMM2, hoge$R3L1.RMM3, hoge$R4L3.RMM3)
colnames(data) <- c( #列名を付加
  "R1L4.HSF1", "R4L2.HSF1", "R2L7.HSF2", "R3L2.HSF2", "R8L1.HSF3", "R8L2.HSF3",
  "R1L1.HSM1", "R5L2.HSM1", "R2L3.HSM2", "R4L8.HSM2", "R3L6.HSM3", "R4L1.HSM3",
  "R1L2.PTF1", "R4L4.PTF1", "R2L4.PTF2", "R6L6.PTF2", "R3L7.PTF3", "R5L3.PTF3",
  "R1L6.PTM1", "R3L3.PTM1", "R2L8.PTM2", "R4L6.PTM2", "R6L2.PTM3", "R6L4.PTM3",
  "R1L7.RMF1", "R5L1.RMF1", "R2L2.RMF2", "R5L8.RMF2", "R3L4.RMF3", "R4L7.RMF3",
  "R1L3.RMM1", "R3L8.RMM1", "R2L6.RMM2", "R5L4.RMM2", "R3L1.RMM3", "R4L3.RMM3")
rownames(data) <- rownames(hoge) #行名を付加
dim(data) #行数と列数を表示

#ファイルに保存(テキストファイル)
tmp <- cbind(rownames(data), data) #保存したい情報をtmpに格納
write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F) #tmpの中身を指定したファイル名で保存
```

元データを整形

```
#in_f <- "http://genome.cshlp.org/content/suppl/2009/12/16/gr.099999.suppTable1.xls"
in_f <- "suppTable1.xls"
out_f <- "sample_blekhman_36.txt" #①
```

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

```
#入力ファイルの読み込み
hoge <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="")
dim(hoge) #行数と列数を表示
```

#サブセットの取得

```
data <- cbind( #必要な列を抽出
  hoge$R1L4.HSF1, hoge$R4L2.HSF1, hoge$R2L7.HSF2,
  hoge$R1L1.HSM1, hoge$R5L2.HSM1, hoge$R2L3.HSM2,
  hoge$R1L2.PTF1, hoge$R4L4.PTF1, hoge$R2L4.PTF2,
  hoge$R1L6.PTM1, hoge$R3L3.PTM1, hoge$R2L8.PTM2,
  hoge$R1L7.RMF1, hoge$R5L1.RMF1, hoge$R2L2.RMF2,
  hoge$R1L3.RMM1, hoge$R3L8.RMM1, hoge$R2L6.RMM2)
colnames(data) <- c( #列名を付す
  "R1L4.HSF1", "R4L2.HSF1", "R2L7.HSF2", "R3L3.HSF3",
  "R1L1.HSM1", "R5L2.HSM1", "R2L3.HSM2", "R4L4.HSM3",
  "R1L2.PTF1", "R4L4.PTF1", "R2L4.PTF2", "R6L6.PTF3",
  "R1L6.PTM1", "R3L3.PTM1", "R2L8.PTM2", "R4L8.PTM3",
  "R1L7.RMF1", "R5L1.RMF1", "R2L2.RMF2", "R5L8.RMF3",
  "R1L3.RMM1", "R3L8.RMM1", "R2L6.RMM2", "R5L4.RMM3)
rownames(data) <- rownames(hoge) #行名を付す
dim(data) #行数と列数を表示
```

#ファイルに保存(テキストファイル)

```
tmp <- cbind(rownames(data), data) #保存用データフレーム
write.table(tmp, out_f, sep="\t", append=F, quote=F)
```

正常終了時の状態。①出力ファイル
(sample_blekhman_36.txt)の中身は、ヘッ
ダ一行や行名部分を除くと、②20,689行
× 36列からなるカウントデータ行列。

```
R Console
+   hoge$R1L2.PTF1, hoge$R4L4.PTF1, hoge$R2L4.PTF2$
+   hoge$R1L6.PTM1, hoge$R3L3.PTM1, hoge$R2L8.PTM2$
+   hoge$R1L7.RMF1, hoge$R5L1.RMF1, hoge$R2L2.RMF2$
+   hoge$R1L3.RMM1, hoge$R3L8.RMM1, hoge$R2L6.RMM2$
> colnames(data) <- c( #列名を付す
+   "R1L4.HSF1", "R4L2.HSF1", "R2L7.HSF2", "R3L3.HSF3",
+   "R1L1.HSM1", "R5L2.HSM1", "R2L3.HSM2", "R4L4.HSM3",
+   "R1L2.PTF1", "R4L4.PTF1", "R2L4.PTF2", "R6L6.PTF3",
+   "R1L6.PTM1", "R3L3.PTM1", "R2L8.PTM2", "R4L8.PTM3",
+   "R1L7.RMF1", "R5L1.RMF1", "R2L2.RMF2", "R5L8.RMF3",
+   "R1L3.RMM1", "R3L8.RMM1", "R2L6.RMM2", "R5L4.RMM3")
> rownames(data) <- rownames(hoge) #行名を付す
> dim(data)
[1] 20689 36
>
> #ファイルに保存(テキストファイル)
> tmp <- cbind(rownames(data), data) #保存用データフレーム
> write.table(tmp, out_f, sep="\t", append=F, quote=F)
> |
```

データ概観

R Console

```
> data[1:2, ]
```

	R1L4.HSF1	R4L2.HSF1	R2L7.HSF2	R3L2.HSF2	R8L1.HSF3	R8L2.HSF3
ENSG000000000003	172	157	147	153	78	90
ENSG000000000005	0	0	0	0	0	0
	R1L1.HSM1	R5L2.HSM1	R2L3.HSM2	R4L8.HSM2	R3L6.HSM3	R4L1.HSM3
ENSG000000000003	60	61	219	202	199	160
ENSG000000000005	0	0	1	0	0	0
	R1L2.PTF1	R4L4.PTF1	R2L4.PTF2	R6L6.PTF2	R3L7.PTF3	R5L3.PTF3
ENSG000000000003	285	289	213	216	180	206
ENSG000000000005	1	0	1	3	0	1
	R1L6.PTM1	R3L3.PTM1	R2L8.PTM2	R4L6.PTM2	R6L2.PTM3	R6L4.PTM3
ENSG000000000003	176	233	316	369	216	212
ENSG000000000005	0	0	0	1	1	0
	R1L7.RMF1	R5L1.RMF1	R2L2.RMF2	R5L8.RMF2	R3L4.RMF3	R4L7.RMF3
ENSG000000000003	259	252	299	165	242	238
ENSG000000000005	0	0	1	0	0	2
	R1L3.RMM1	R3L8.RMM1	R2L6.RMM2	R5L4.RMM2	R3L1.RMM3	R4L3.RMM3
ENSG000000000003	207	217	676	672	338	367
ENSG000000000005	1	1	0	0	0	0

```
> |
```

データ概観

ヒト(HS)

- オス3匹(M1, M2, **M3**)
- メス3匹(F1, F2, F3)

チンパンジー(PT)

- オス3匹(M1, M2, M3)
- メス3匹(F1, **F2**, F3)

アカゲザル(RM)

- オス3匹(M1, M2, M3)
- メス3匹(**F1**, F2, F3)

ENSG000000000000

ENSG000000000003

ENSG000000000005

ENSG000000000003

ENSG000000000005

ENSG000000000003

ENSG000000000005

> |

全体的に、四角で囲ったtechnical replicates(同一個体の反復)間の類似度が、biological replicates(同一生物種の別個体)間の類似度よりも高そうであることがわかる。

	R1L4.HSF1	R4L2.HSF1	R2L7.HSF2	R3L2.HSF2	R8L1.HSF3	R8L2.HSF3
	172	157	147	153	78	90
	0	0	0	0	0	0
	R1L1.HSM1	R5L2.HSM1	R2L3.HSM2	R4L8.HSM2	R3L6.HSM3	R4L1.HSM3
	60	61	219	202	199	160
	0	0	1	0	0	0
	R1L2.PTF1	R4L4.PTF1	R2L4.PTF2	R6L6.PTF2	R3L7.PTF3	R5L3.PTF3
	285	289	213	216	180	206
	1	0	1	3	0	1
	R1L6.PTM1	R3L3.PTM1	R2L8.PTM2	R4L6.PTM2	R6L2.PTM3	R6L4.PTM3
	176	233	316	369	216	212
	0	0	0	1	1	0
	R1L7.RMF1	R5L1.RMF1	R2L2.RMF2	R5L8.RMF2	R3L4.RMF3	R4L7.RMF3
	259	252	299	165	242	238
	0	0	1	0	0	2
	R1L3.RMM1	R3L8.RMM1	R2L6.RMM2	R5L4.RMM2	R3L1.RMM3	R4L3.RMM3
	207	217	676	672	338	367
	1	1	0	0	0	0

EXCELで概観

出力ファイル(sample_blekhman_36.txt)をEXCELで眺めるところなん感じ。①はENSG000000000971という遺伝子領域上に2,262リードマップされたことを表す。②はENSG000000001460の遺伝子領域上に3リードマップされたことを表す。もしこの2つの配列長が同じなら、マップされたリード数が多い前者①の発現レベルが高いという理解でよい。

	A	B	C	D	E	F	G	H	I	J
1		R1L4.HSF1	R4L2.HSF1	R2L7.HSF2	R3L2.HSF2	R8L1.HSF3	R8L2.HSF3	R1L1.HSM1	R5L2.HSM1	R2L3
2	ENSG000000000003	172	157	147	153	78	90	60	61	2
3	ENSG000000000005	0	0	0	0	0	0	0	0	0
4	ENSG000000000419	36	45	26	35	16	40	17	22	
5	ENSG000000000457	41	50	28	34	34	42	50	64	
6	ENSG000000000460	3	3	8	9	7	5	9	6	
7	ENSG000000000938	23	21	30	35	112	98	32	41	
8	ENSG000000000971	2262	2503	3473	3752	1665	1740	1726	1874	32
9	ENSG00000001036	155	142	118	133	79	110	99	101	1
10	ENSG00000001084	323	307	377	360	151	155	155	181	4
11	ENSG00000001167	19	17	15	15	16	20	13	16	
12	ENSG00000001460	3	0	0	1	1	4	0	1	
13	ENSG00000001461	25	24	22	15	14	20	13	15	
14	ENSG00000001497	59	58	46	47	46	43	39	41	
15	ENSG00000001561	22	26	23	27	28	25	29	33	
16	ENSG00000001617	30	34	24	27	77	73	40	30	
17	ENSG00000001626	9	3	12	32	37	33	24	19	

EXCELで概観

サンプル(列)ごとにマップされた総リード数を計算した結果。サンプル間比較の場合には、この総リード数を揃えるのが基本戦略。

sample_blekhman_36.txt - Excel

ファイル ホーム 挿入 ページレイアウト 数式 データ 校閲 表示 アドイン 門田幸二

B20692 : fx =SUM(B2:B20690)

	A	B	C	D	E	F	G	H	I
20677	ENSG00000221765	0	0	0	0	0	0	0	0
20678	ENSG00000221766	0	0	0	0	0	0	0	0
20679	ENSG00000221767	0	0	0	0	0	0	0	0
20680	ENSG00000221768	0	0	0	0	0	0	0	0
20681	ENSG00000221770	4	2	4	0	2	2	0	0
20682	ENSG00000221771	0	0	0	0	0	0	0	0
20683	ENSG00000221775	0	0	0	0	0	0	0	0
20684	ENSG00000221778	0	0	0	0	0	0	0	0
20685	ENSG00000221781	0	0	0	0	0	0	0	0
20686	ENSG00000221782	0	0	0	0	0	0	0	0
20687	ENSG00000221783	0	0	0	0	0	1	0	0
20688	ENSG00000221784	0	0	0	0	0	0	0	0
20689	ENSG00000221786	0	0	0	0	0	0	0	0
20690	ENSG00000221788	0	0	0	0	0	0	0	0
20691									
20692		1665987	1719125	1620189	1801009	1393867	1450604	1346515	1497738
20693									

sample_blekhman_36

準備完了

100%

EXCELで概観

もし揃えずに、例えば①と②のサンプル間比較(発現変動遺伝子(DEG)検出)を行うと、①のほうが②に比べて全体的に $(1,801,009 / 1,346,515 = 1.34)$ 倍高発現な状態であることを意味するので、①で高発現となるDEGが多く検出されるだろう。もちろんそれは間違い。

The screenshot shows an Excel spreadsheet with a table of data. The columns are labeled A through I. The rows are numbered from 20677 to 20693. The cell B20692 contains the formula =SUM(B2:B20690). The cell C20692 is highlighted with a red arrow labeled ①. The cell B20692 is also highlighted with a red arrow labeled ②.

	A	B	C	D	E	F	G	H	I
20677	ENSG00000221765	0	0	0	0	0	0	0	0
20678	ENSG00000221766	0	0	0	0	0	0	0	0
20679	ENSG00000221767	0	0	0	0	0	0	0	0
20680	ENSG00000221768	0	0	0	0	0	0	0	0
20681	ENSG00000221770	4	2	4	0	2	2	0	0
20682	ENSG00000221771	0	0	0	0	0	0	0	0
20683	ENSG00000221775	0	0	0	0	0	0	0	0
20684	ENSG00000221778	0	0	0	0	0	0	0	0
20685	ENSG00000221781	0	0	0	0	0	0	0	0
20686	ENSG00000221782	0	0	0	0	0	0	0	0
20687	ENSG00000221783	0	0	0	0	0	1	0	0
20688	ENSG00000221784	0	0	0	0	0	0	0	0
20689	ENSG00000221786	0	0	0	0	0	0	0	0
20690	ENSG00000221788	0	0	0	0	0	0	0	0
20691									
20692		1665987	1719125	1620189	1801009	1393867	1450604	1346515	1497738
20693									

colSumsとrange

colSums関数は、列ごとの総リード数を調べるときに便利。③総リード数の最小と最大を調べる場合は、range関数を利用する。

R Console window showing R code and its output.

```

> colSums(data)
R1L4.HSF1 R4L2.HSF1 R2L7.HSF2 R3L2.HSF2 R8L1.HSF3 R8L2.HSF3
 1665987  1719125  1620189  1801009  1393867  1450604
R1L1.HSM1 R5L2.HSM1 R2L3.HSM2 R4L8.M2 R3L6.HSM3 R4L1.HSM3
 1346515  1497738  2217235  2167994  1974228  1825373
R1L2.PTF1 R4L4.PTF1 R2L4.PTF2 R6L6.PTF2 R3L7.PTF3 R5L3.PTF3
 2667264  2677771  1910402  1881431  1838275  1813918
R1L6.PTM1 R3L3.PTM1 R2L8.PTM2 R4L6.PTM2 R6L2.PTM3 R6L4.PTM3
 1481536  1694688  1608138  1946512  1745188  1803555
R1L7.RMF1 R5L1.RMF1 R2L2.RMF2 R5L8.RMF2 R3L4.RMF3 R4L7.RMF3
 2400660  2110806  2338433  1533906  2685655  2534595
R1L3.RMM1 R3L8.RMM1 R2L6.RMM2 R5L4.RMM2 R3L1.RMM3 R4L3.RMM3
 2657274  2505941  1942296  1974502  2119496  2411707
> range(colSums(data))
[1] 1346515 2685655
>

```

The R console shows the execution of `colSums(data)` and `range(colSums(data))`. The output displays the total read counts for each column, with the minimum value being 1346515 and the maximum being 2685655.

Below the console, a data viewer window is open, showing a subset of the data frame. Red arrows point to specific values in the data frame:

- ① Points to the value 1801009 in the fourth column of the data frame.
- ② Points to the value 1346515 in the first column of the data frame.
- ③ Points to the value 2685655 in the second column of the data frame.

The data viewer also shows the row numbers 20689, 20690, 20691, 20692, and 20693, and the sample identifier `sample_blekhman_36`.

apply, min, max

① ②

R Console

```
> apply(data, 2, sum)
```

```
R1L4.HSF1 R4L2.HSF1 R2L7.HSF2 R3L2.HSF2 R8  
1665987 1719125 1620189 1801009
```

```
R1L1.HSM1 R5L2.HSM1 R2L3.HSM2 R4L8.HSM2 R3  
1346515 1497738 2217235 2167994
```

```
R1L2.PTF1 R4L4.PTF1 R2L4.PTF2 R6L6.PTF2 R3L7.PTF3 R5L3.PTF3  
2667264 2677771 1910402 1881431 1838275 1813918
```

```
R1L6.PTM1 R3L3.PTM1 R2L8.PTM2 R4L6.PTM2 R6L2.PTM3 R6L4.PTM3  
1481536 1694688 1608138 1946512 1745188 1803555
```

```
R1L7.RMF1 R5L1.RMF1 R2L2.RMF2 R5L8.RMF2 R3L4.RMF3 R4L7.RMF3  
2400660 2110806 2338433 1533906 2685655 2534595
```

```
R1L3.RMM1 R3L8.RMM1 R2L6.RMM2 R5L4.RMM2 R3L1.RMM3 R4L3.RMM3  
2657274 2505941 1942296 1974502 2119496 2411707
```

```
> age <- apply(data, 2, sum)
```

```
> min(age)
```

```
[1] 1346515
```

```
> max(age)
```

```
[1] 2685655
```

```
> |
```

行列演算といえばapply関数。行列dataを入力として、
 ①列ごと(MARGIN=2)に、②sum関数を実行せよ、とい
 う意味。総リード数の最小と最大は、range関数でなく
 てもminとmax関数を用いて別々に計算してもよい。
 様々な関数を紹介しているが、自分が使う際はどれか
 一つでよい。一度でも見ておけば、少しでも記憶に残
 るだろうという思想のもと、羅列的に紹介している。

The screenshot shows the RStudio interface. The top bar has the R logo and the word "Console". Below it is the R Console window containing R code and its output. The code uses the `apply` function to calculate the sum of each column in a data frame named `data`. It also shows the `min` and `max` functions being applied to the resulting vector of sums. The bottom part of the interface is a data viewer showing a 10x10 grid of data from a file named `sample_blekhman_36`. The first two rows are labeled with their respective row numbers (20692, 20693). The first column is labeled with the file name. The data consists of various numerical values.

colMeans, rowMeans

R Console

```
> head(apply(data, 2, mean)) ①
R1L4.HSF1 R4L2.HSF1 R2L7.HSF2 R3L2.HSF2 R8L1.HSF3
 80.52525 83.09367 78.31161 87.05152 67.37237
> head(colMeans(data)) ②
R1L4.HSF1 R4L2.HSF1 R2L7.HSF2 R3L2.HSF2 R8L1.HSF3 R8L2.HSF3
 80.52525 83.09367 78.31161 87.05152 67.37237 70.11475
> head(apply(data, 1, mean)) ③
ENSG00000000003 ENSG00000000005 ENSG00000000419
 237.2500000 0.3888889 34.3055556
ENSG00000000457 ENSG00000000460 ENSG00000000938
 63.6111111 4.3333333 38.6666667
> head(rowMeans(data)) ④
ENSG00000000003 ENSG00000000005 ENSG00000000419
 237.2500000 0.3888889 34.3055556
ENSG00000000457 ENSG00000000460 ENSG00000000938
 63.6111111 4.3333333 38.6666667
> head(rowSums(data)) ⑤
ENSG00000000003 ENSG00000000005 ENSG00000000419
 8541 14 1235
ENSG00000000457 ENSG00000000460 ENSG00000000938
 2290 156 1392
> |
```

①列ごとにマップされたリード数の平均を算出、
 ②colMeans関数も同じ機能。③行ごとにマップされたリード数の平均を算出、④rowMeans
 関数も同じ機能。⑤行ごとにマップされたリード数を算出。rowSums関数は、低発現遺伝子
 のフィルタリング時にも利用される。

EXCELと比較

sample_blekhman_36.txt - Excel

	A	B	C	D	E	F	G	H	I	J
1		R1L4.HSF1	R4L2.HSF1	R2L7.HSF2	R3L2.HSF2	R8L1.HSF3	R8L2.HSF3	R1L1.HSM1	R5L2.HSM1	R2L3
2	ENSG000000000003	172	157	147	153	78	90	60	61	1
3	ENSG000000000005	0	0	0	0	0	0	0	0	1
4	ENSG000000000419	36	45	26	35	16	40	17	22	1
5	ENSG000000000457	41	50	28	24	34	42	50	64	1
						7	5	9	6	
						112	98	32	41	
						1665	1740	1726	1874	32
						79	110	99	101	1
						151	155	155	181	2
						16	20	13	16	
						1	4	0	1	
						14	20	13	15	
						46	43	39	41	
						28	25	29	33	
						77	73	40	30	
						37	33	24	19	

R Console

```
> head(rowMeans(data))
ENSG000000000003 ENSG000000000005 ENSG000000000419
 237.2500000 0.3888889 34.3055556
ENSG000000000457 ENSG000000000460 ENSG000000000938
 63.6111111 4.3333333 38.6666667
> head(rowSums(data))
ENSG000000000003 ENSG000000000005 ENSG000000000419
 8541 14 1235
ENSG000000000457 ENSG000000000460 ENSG000000000938
 2290 156 1392
> |
```

summary関数

R Console

```
> summary(data[, 1:6])
```

	R1L4.HSF1	R4L2.HSF1	R2L7.HSF2
Min. :	0.00	0.00	0.00
1st Qu.:	0.00	0.00	0.00
Median :	4.00	4.00	3.00
Mean :	80.53	83.09	78.31
3rd Qu.:	35.00	36.00	32.00
Max. :	276431.00	287958.00	241969.00

	R3L2.HSF2	R8L1.HSF3	R8L2.HSF3
Min. :	0.00	0.00	0.00
1st Qu.:	0.00	0.00	0.00
Median :	4.00	5.00	5.00
Mean :	87.05	67.37	70.11
3rd Qu.:	36.00	35.00	36.00
Max. :	265900.00	111548.00	117717.00

> |

サンプルごとの要約統計量を概観する場合によく用いる。ここでは、最初の6サンプル分(HS群のメス)に絞って表示。私の最初の着眼点は黒枠のあたり。特に、①1st Qu. (第一四分位数)が全6サンプルで0であることから、20,689遺伝子中の少なくとも25%はゼロカウントであることがわかる。

①

①

summary関数

R Console

```
> summary(data[, 1:6])
   R1L4.HSF1          R4L2.HSF1
Min. : 0.00    Min. : 0.00
1st Qu.: 0.00    1st Qu.: 0.00
Median : 4.00    Median : 4.00
Mean   : 80.53    Mean   : 83.09
3rd Qu.: 35.00    3rd Qu.: 36.00
Max.   :276431.00 Max.   :287958.00
   R3L2.HSF2          R8L1.HSF3          R8L2.HSF3
Min. : 0.00    Min. : 0.00    Min. : 0.00
1st Qu.: 0.00    1st Qu.: 0.00    1st Qu.: 0.00
Median : 4.00    Median : 5.00    Median : 5.00
Mean   : 87.05    Mean   : 67.37    Mean   : 70.11
3rd Qu.: 36.00    3rd Qu.: 35.00    3rd Qu.: 36.00
Max.   :265900.00 Max.   :111548.00 Max.   :117717.00
> |
```

次に見るのは②Medianの値。これは2nd Qu. (第二四分位数)と同じである。サンプル全体にわたって、ここを概観する。そして、低発現遺伝子のフィルタリングの際に、(ここでは最初の6サンプル分しか示していないが)マップされたリード数が5以下のものを除く処理を行うと、半分以上が落とされるだろう、などの見込みをつける。



summary関数

R Console

```
> summary(data[, 1:6])
   R1L4.HSF1          R4L2.HSF1
Min. : 0.00    Min. : 0.00
1st Qu.: 0.00   1st Qu.: 0.00
Median : 4.00   Median : 4.00
Mean   : 80.53  Mean   : 83.09
3rd Qu.: 35.00  3rd Qu.: 36.00
Max.  :276431.00 Max.  :287958.00
   R3L2.HSF2          R8L1.HSF3          R8L2.HSF3
Min. : 0.00    Min. : 0.00    Min. : 0.00
1st Qu.: 0.00   1st Qu.: 0.00   1st Qu.: 0.00
Median : 4.00   Median : 5.00   Median : 5.00
Mean   : 87.05  Mean   : 67.37  Mean   : 70.11
3rd Qu.: 36.00  3rd Qu.: 35.00  3rd Qu.: 36.00
Max.  :265900.00 Max.  :111548.00 Max.  :117717.00
> |
```

ちなみに私は、③Mean(平均値)をほとんど見ません。一応見ますが重要視していません。黒枠内の数値の関係(Mean > 3rd Qu.)から、ごく一部の異常に高発現(リード数の多い)の遺伝子の影響がかなり大きそうだから、この種の外れ値の効果を排除できないMeanのような要約統計量は使わないほうがよいと判断します。

③

③

実用上は…

サンプル間比較の場合は、Rの発現変動解析用Rパッケージをそのまま利用すればよい(うまくデータの正規化を行ってくれる)。8/5の統計解析のところで発現変動解析を行う予定です。

■ 総リード数補正(RPM補正)

- Mortazavi et al., *Nat. Methods*, 2008
- 総リード数を100万など一定の値に揃えるベーシックな補正。外れ値に影響されやすい

■ TMM補正(edgeRパッケージ)

- Robinson and Oshlack, *Genome Biol.*, 2010
- 高発現側と低発現側で一定数をトリムして外れ値の影響を排除

■ TbT補正(TCCパッケージ)

- Kadota et al., *Algorithms Mol. Biol.*, 2012
- TMMを含むedgeR(やDESeq)を内部的に利用して、高発現側と低発現側の外れ値に相当する発現変動遺伝子(DEG)をより正確に排除することで頑健な正規化を達成。
DEG-elimination strategy (DEGES)の基本形を提唱した論文

■ DEGES補正(TCCパッケージ)

- Sun et al., *BMC Bioinformatics*, 2013。TCC原著論文
- DEGESを一般化して、より高速かつ頑健な正規化を達成。edgeRやDESeq(後にDESeq2)の通常の手順を内部的に繰り返し実行して頑健な結果を得る枠組みを提供。
- Multi-group comparisonでもTCCの枠組みが有効であることを示した論文が近々…。

クラスタリング

- [解析 | 発現量推定\(トランスクリプトーム配列を利用\) \(last modified 2014/02/05\)](#)
- [解析 | クラスタリング | について \(last modified 2014/02/05\)](#)
- [解析 | クラスタリング | サンプル間 | \[hclust\]\(#\) \(last modified 2015/02/26\) NEW](#)
- [解析 | クラスタリング | サンプル間 | \[TCC\\(Sun 2013\\)\]\(#\) \(last modified 2015/03/02\) NEW](#)
- [解析 | クラスタリング | 遺伝子間 | \[MBCluster.Seq \\(Si 2014\\)\]\(#\) \(last modified 2014/02/05\)](#)
- [解析 | クラスタリング | データ構造 \(last modified 2014/02/05\)](#)
- [解析 | クラスタリング | パッケージ \(last modified 2014/02/05\)](#)

入力ファイルは20,689遺伝子×36サンプルのカウントデータファイル。ヒト(HS)、チンパンジー(PT)、アカゲザル(RM)の3生物種のデータ。各12サンプル。TCCパッケージを用いて、これのサンプル間クラスタリングを行います。

解析 | クラスタリング | サンプル間 | TCC(Sun_2013) NEW

TCCパッケージを用いてサンプル間クラスタリングを行うやり方を示します。clusterSample関数を利用した頑健なクラスタリング結果を返します。
「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。

7. サンプルデータ41のリアルデータ([sample_blekhman_36.txt](#))の場合:

Blekhman et al., Genome Res., 2010 の 20,689 genes×36 samples のカウントデータです。

```
in_f <- "sample_blekhman_36.txt"          #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.png"                       #出力ファイル名を指定してout_fに格納
param_fig <- c(700, 400)                    #ファイル出力時の横幅と縦幅を指定(単位はピクセル)

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

#入力ファイルの読み込み
data <- read.table(in_f, header=TRUE, row.names=1, 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
out <- clusterSample(data, dist.method="spearman",#クラスタリング実行結果をoutに格納
                      hclust.method="average", unique.pattern=TRUE)#クラスタリング実行結果をoutに格納

#ファイルに保存
png(out_f, pointsize=13, width=param_fig[1], height=param_fig[2])#出力ファイルの各種パラメータを指定
par(mar=c(0, 4, 1, 0))                         #下、左、上、右の順で余白(行)を指定
plot(out, sub="", xlab="", cex.lab=1.2,         #樹形図(デンドログラム)の表示
     cex=1.3, main="", ylab="Height")            #樹形図(デンドログラム)の表示
dev.off()                                       #おまじない
```

①出力は、[hoge7.png](#)という名前のPNGファイル。②サイズは、700 × 400ピクセル。これは論文の図としても使えるレベル(実際我々の論文中でも使っている)

クラスタリング

7. サンプルデータ41のリアルデータ([sample blekhman 36.txt](#))の場合:

[Blekhman et al., Genome Res., 2010](#)の 20,689 genes×36 samplesのカウントデータです。

```

in_f <- "sample_blekhman_36.txt"          #入力ファイル名を指定してin_fに格納
out_f <- "hoge7.png"                      #出力ファイル名を指定してout_fに格納
param_fig <- c(700, 400)                  #ファイル出力時の横幅と縦幅を指定(単位はピクセル)

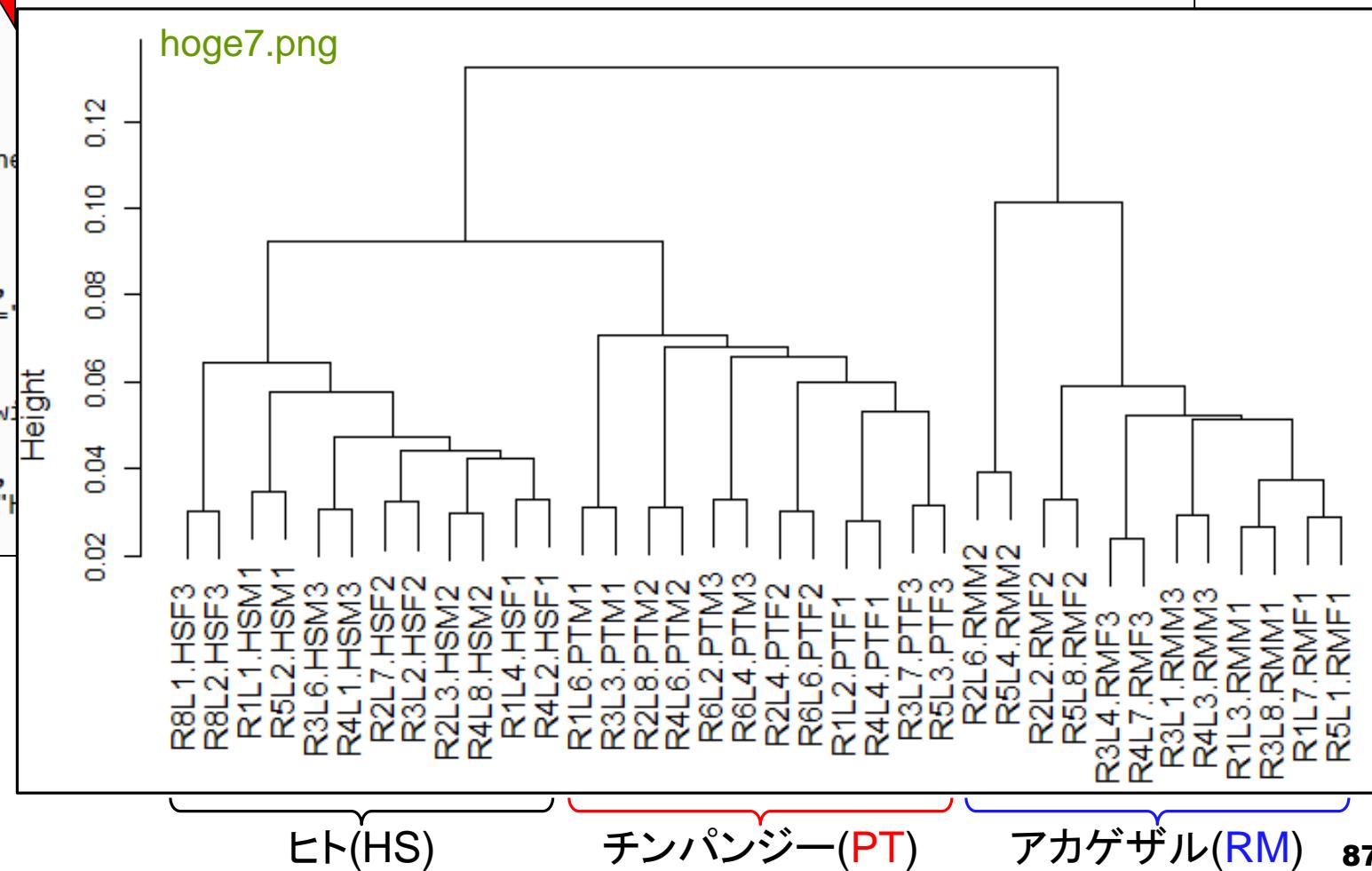
#必要なパッケージをロード
library(TCC)

#入力ファイルの読み込み
data <- read.table(in_f, header=TRUE)
dim(data)

#本番
out <- clusterSample(data,
                      hclust.method="ward.D")

#ファイルに保存
png(out_f, pointsize=13, width=700, height=400)
par(mar=c(0, 4, 1, 0))
plot(out, sub="", xlab="", ylab="Height",
     cex=1.3, main="hoge7.png",
     dev.off())

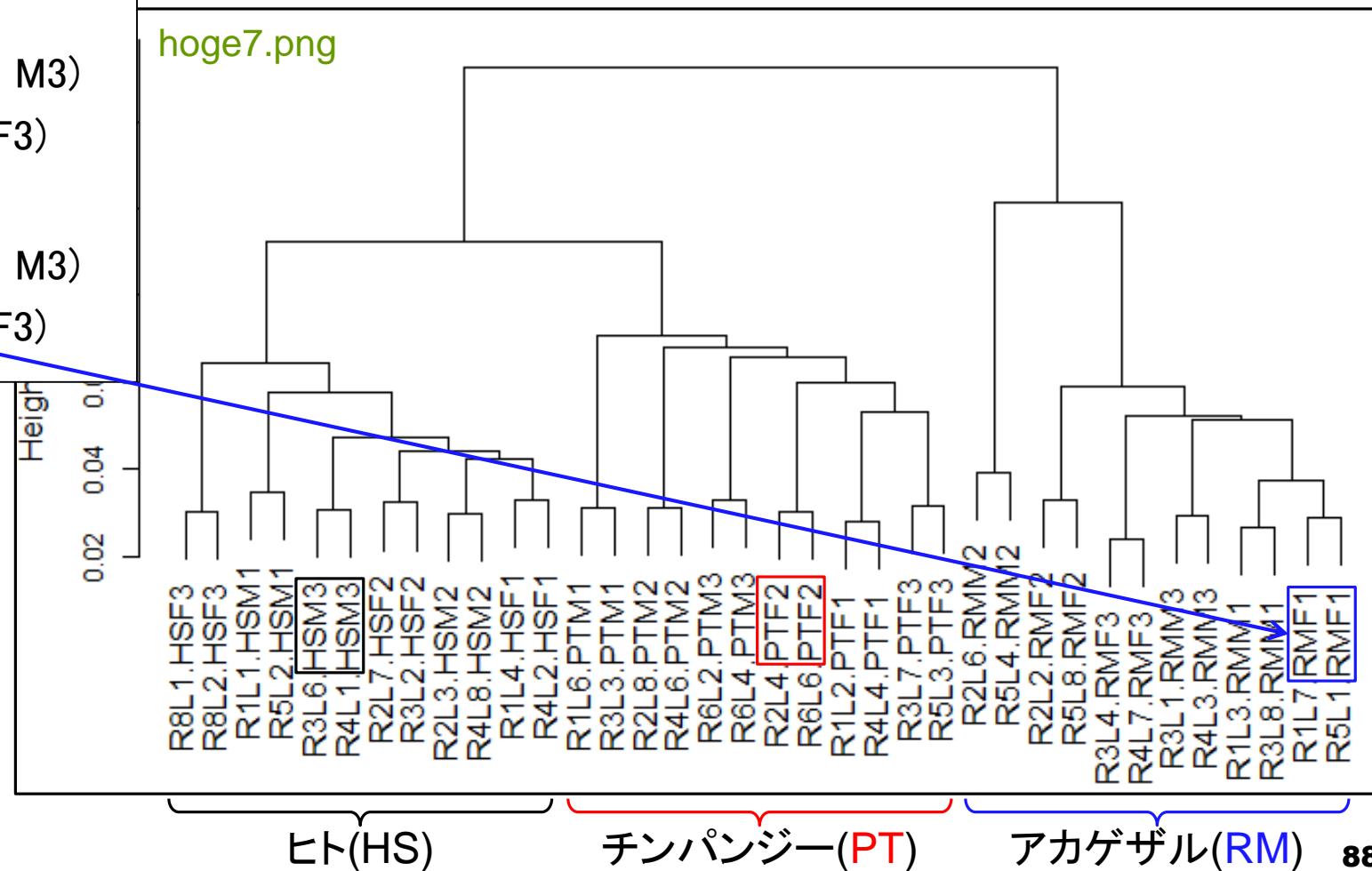
```



クラスタリング

- ヒト(HS)
 - オス3匹(M1, M2, **M3**)
 - メス3匹(F1, F2, F3)
- チンパンジー(PT)
 - オス3匹(M1, M2, M3)
 - メス3匹(F1, **F2**, F3)
- アカゲザル(RM)
 - オス3匹(M1, M2, M3)
 - メス3匹(**F1**, F2, F3)

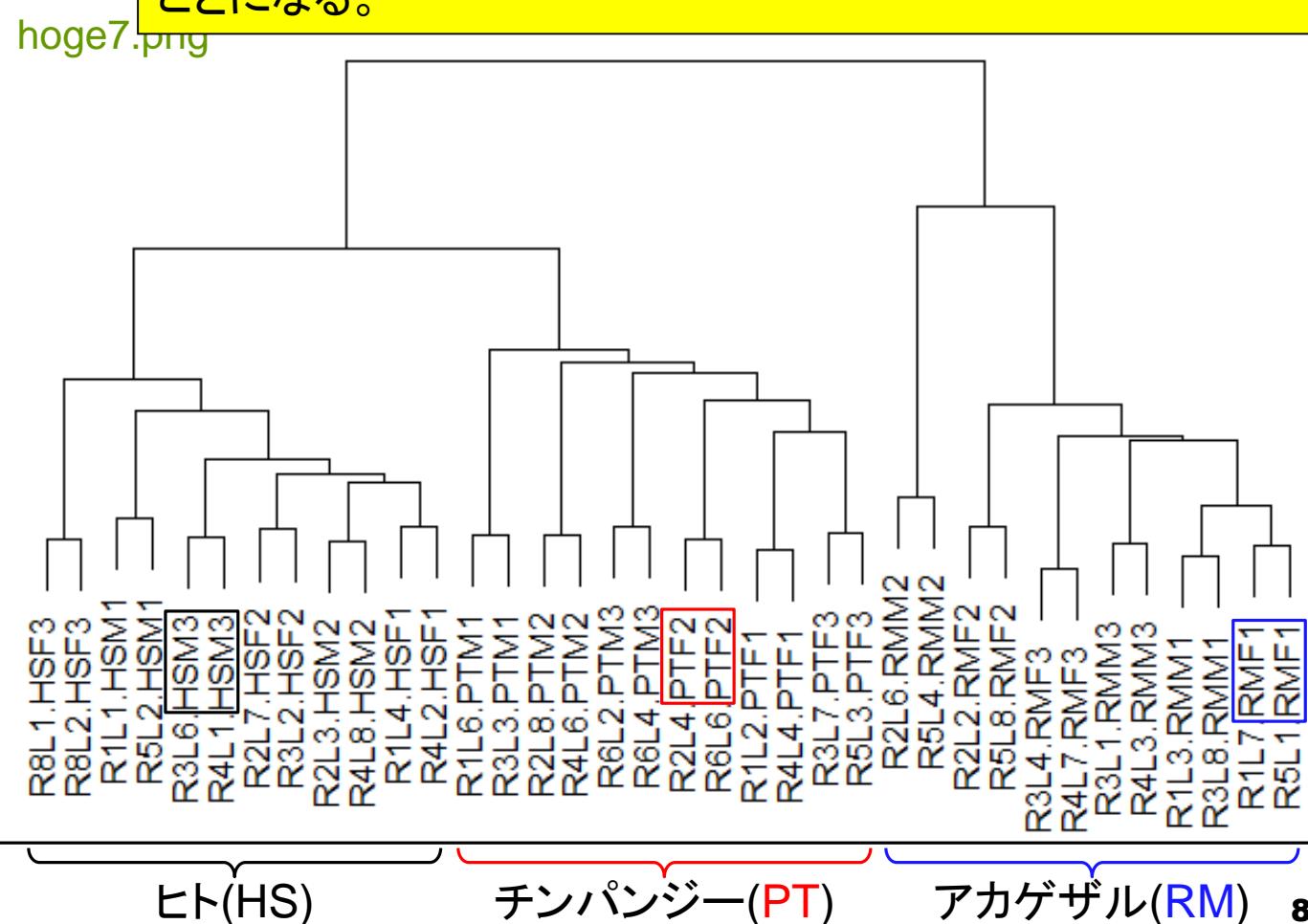
全個体について、同一個体を分割したtechnical replicatesのデータで末端のクラスターを形成していることが分かる。これはtechnical replicatesのデータ同士の類似度が非常に高いことを示している。妥当ですよね。



クラスタリング

- ヒト(HS)
 - オス3匹(M1, M2, **M3**)
 - メス3匹(F1, F2, F3)
- チンパンジー(PT)
 - オス3匹(M1, M2, M3)
 - メス3匹(F1, **F2**, F3)
- アカゲザル(RM)
 - オス3匹(M1, M2, M3)
 - メス3匹(**F1**, F2, F3)

統計的手法で2群間比較(例えばMales vs. Females)をする目的は、同一群内の別個体(biological replicates)のばらつきの程度を見積もっておき(モデル構築)、比較する2群間で発現に変動がないという前提(帰無仮説)からどれだけ離れているのかをp値で評価することである。p値が低ければ低いほど「発現変動していない(帰無仮説に従う)」とは考えにくく、帰無仮説を棄却して「発現変動している(DEGである)」と判定することになる。



サブセット抽出と整形

・(削除予定)個別パッケージのインストール (last modified 2015/02/20)

・基本的な利用法 (last modified 2015/04/03)

・サンプルデータ (last modified 2015/06/15) NEW

・バイオインフォマティクス人材育成カリキュラム(次世代シーケンサ)|NGSハンドブック

・バイオ

・書籍

・書籍

・書籍

サンプルデータ NEW

1. ① [Blekhman et al., Genome Res., 2010](#) のリアルカウントデータです。

1つ前の sample41.txt とは違って、technical replicates の 2 列分のデータは足して 1 列分のデータとしています。20,689 genes × 18 samples のカウントデータ ([sample_blekhman_18.txt](#)) です。

```
#in_f <- "http://genome.cshlp.org/content/suppl/2009/12/16/gr.099226.109.DC1/suppTable1.xls">#入力ファイル名を指定してin_fに格納
in_f <- "suppTable1.xls"
out_f <- "sample_blekhman_18.txt" #出力ファイル名を指定してout_fに格納

#入力ファイルの読み込み
hoge <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="")
dim(hoge) #行数と列数を表示

#サブセットの取得
data <- cbind(
  hoge$R1L4.HSF1 + hoge$R4L2.HSF1, hoge$R2L7.HSF2 + hoge$R3L2.HSF2, hoge$R8L1.HSF3 + hoge$R8L2.HSF3,
  hoge$R1L1.HSM1 + hoge$R5L2.HSM1, hoge$R2L3.HSM2 + hoge$R4L8.HSM2, hoge$R3L6.HSM3 + hoge$R4L1.HSM3,
  hoge$R1L2.PTF1 + hoge$R4L4.PTF1, hoge$R2L4.PTF2 + hoge$R6L6.PTF2, hoge$R3L7.PTF3 + hoge$R5L3.PTF3,
  hoge$R1L6.PTM1 + hoge$R3L3.PTM1, hoge$R2L8.PTM2 + hoge$R4L6.PTM2, hoge$R6L2.PTM3 + hoge$R6L4.PTM3,
  hoge$R1L7.RMF1 + hoge$R5L1.RMF1, hoge$R2L2.RMF2 + hoge$R5L8.RMF2, hoge$R3L4.RMF3 + hoge$R4L7.RMF3,
  hoge$R1L3.RMM1 + hoge$R3L8.RMM1, hoge$R2L6.RMM2 + hoge$R5L4.RMM2, hoge$R3L1.RMM3 + hoge$R4L3.RMM3)
colnames(data) <- c( #列名を付加
  "HSF1", "HSF2", "HSF3", "HSM1", "HSM2", "HSM3",
  "PTF1", "PTF2", "PTF3", "PTM1", "PTM2", "PTM3",
  "RMF1", "RMF2", "RMF3", "RMM1", "RMM2", "RMM3")
rownames(data) <- rownames(hoge) #行名を付加
dim(data) #行数と列数を表示
```

統計的手法の多くは、biological replicates のデータを前提としている。technical replicates のデータをマージ(merge; collapseともいうらしい)したものを作成。③出力ファイルは [sample_blekhman_18.txt](#)。サンプル名部分は必要最小限の情報のみにしている。

クラスタリング

20,689遺伝子 × 18サンプルの
biological replicatesのみからなる
カウントデータでクラスタリング。

- 解析 | 発現量推定(トランскriptom配列を利用) (last modified 2014/07/09)
- **解析 | クラスタリング | について** (last modified 2014/02/05)
- 解析 | クラスタリング | サンプル間 | [hclust](#) (last modified 2015/02/26) NEW
- 解析 | クラスタリング | サンプル間 | [TCC\(Sun_2013\)](#) (last modified 2015/03/02) NEW
- 解析 | クラスタリング | 遺伝子間 | [MBCluster.Seq \(Si et al 2014\)](#) (last modified 2014/02/05)
- **解析 | クラスタリング | サンプル間 | TCC(Sun_2013) NEW**
- 解析

①

解析 | クラスタリング | サンプル間 | TCC(Sun_2013) NEW

TCCパッケージを用いてサンプル間クラスタリングを行うやり方を示します。clusterSample関数を利用した頑健なクラスタリング結果を返します。
「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、下をコピペ

1. 59.

Neyret-
ン|ゲノ

in_f
out_f
param

#必要
library

#入力
data
dim(d

#本番
out <

in_f <- "sample_blekhman_18.txt"
out_f <- "hoge8.png"
param_fig <- c(700, 400)

#必要なパッケージをロード
library(TCC)

#入力ファイルの読み込み
data <- read.table(in_f, header=TRUE, row.names=1, sep="\t", quote="")#in_fで指定したファイル
dim(data) #オブジェクトdataの行数と列数を表示

#本番
out <- clusterSample(data, dist.method="spearman", #クラスタリング実行結果をoutに格納
hclust.method="average", unique.pattern=TRUE)#クラスタリング実行結果をoutに格納

#ファイルに保存
png(out_f, pointsize=13, width=param_fig[1], height=param_fig[2])#出力ファイルの各種パラメータ
par(mar=c(0, 4, 1, 0)) #下、左、上、右の順で余白（行）を指定
plot(out, sub="", xlab="", cov.lab=1, 2 #樹形図(デンドログラム)の表示

②

8. サンプルデータ42のリアルデータ(sample_blekhman_18.txt)の場合:

Blekhman et al., Genome Res., 2010 の 20,689 genes×18 samples のカウントデータです。

#入力ファイル名を指定してin_fに格納

#出力ファイル名を指定してout_fに格納

#ファイル出力時の横幅と縦幅を指定(単位はピクセル)

#パッケージの読み込み

#in_fで指定したファイル
#オブジェクトdataの行数と列数を表示

#クラスタリング実行結果をoutに格納

#クラスタリング実行結果をoutに格納

#出力ファイルの各種パラメータ

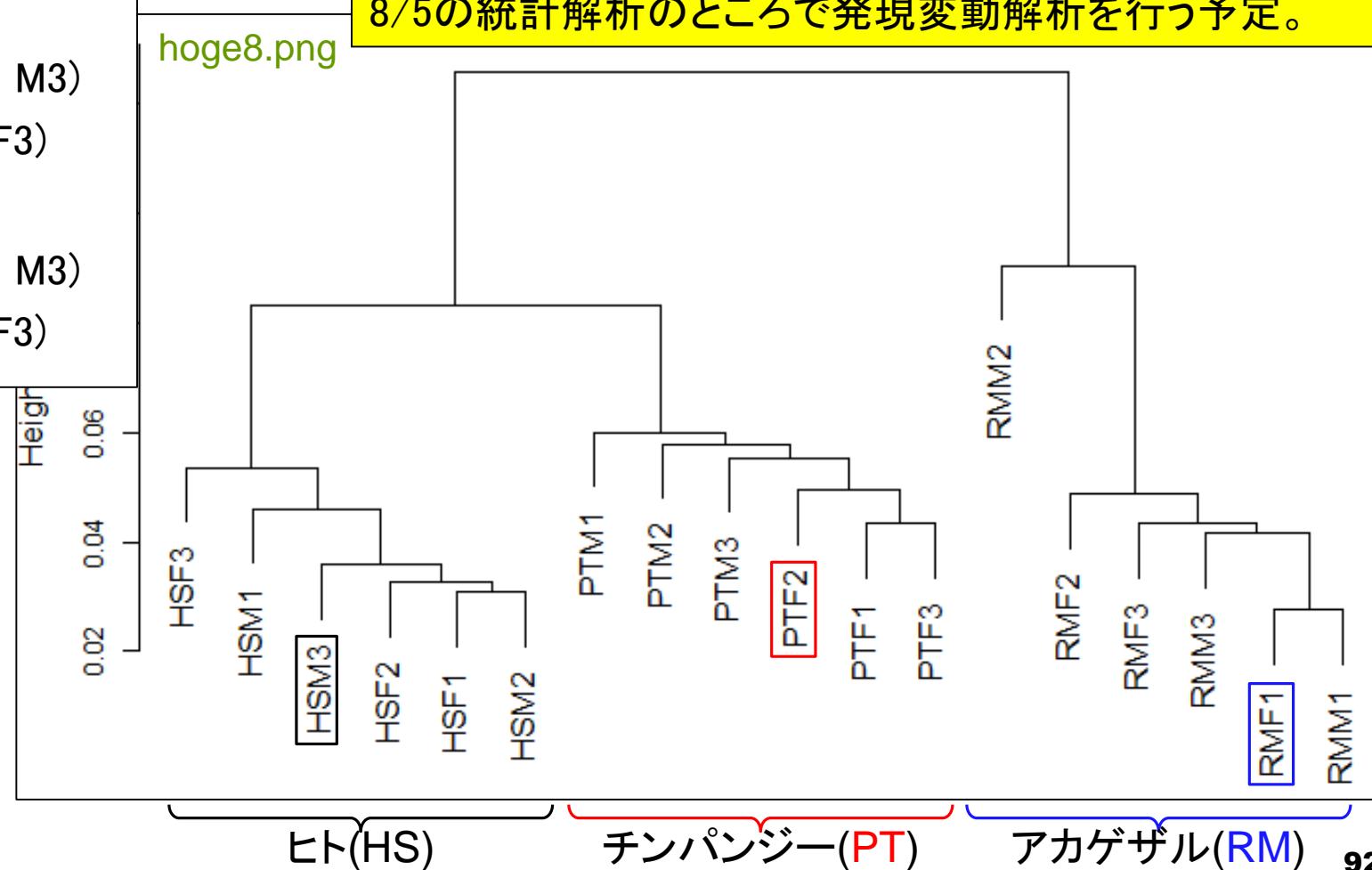
#下、左、上、右の順で余白（行）を指定

#樹形図(デンドログラム)の表示

クラスタリング

- ヒト(HS)
 - オス3匹(M1, M2, **M3**)
 - メス3匹(F1, F2, F3)
- チンパンジー(PT)
 - オス3匹(M1, M2, M3)
 - メス3匹(F1, **F2**, F3)
- アカゲザル(RM)
 - オス3匹(M1, M2, M3)
 - メス3匹(**F1**, F2, F3)

36サンプルのときの結果と同様、全体的なトポロジーは同じ。このクラスタリング結果を眺めるだけで、**DEG**検出結果のイメージは大体つかめる。例1:「HS vs. RMで得られる**DEG**数」のほうが「HS vs. PTで得られる**DEG**数」よりも多そう。例2:ヒトは「オス vs. メス」での**DEG**数は0に近いだろう。例3:RMM2が外れサンプルっぽいので、これを除去すれば生物種間比較時に**DEG**数が増えるだろう。8/5の統計解析のところで発現変動解析を行う予定。



Contents

- R基礎(初級)
 - おさらい
 - コード内部の説明(ファイルの読み込み、行列演算の基礎)
 - リアルRNA-seqカウントデータ(数値行列データ)
- R各種パッケージ(中級): 代表的なパッケージの利用法
 - (パッケージのインストール法)
 - 基本情報取得(コンティグ数、配列長、N50、GC含量)
 - 任意の領域の切り出し
 - GC含量計算部分の説明

FASTA形式

FASTAフォーマット [編集]

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

FASTAでは、シーケンスデータの記述形式として FASTAフォーマットという形式を使う。FASTAフォーマットはブレ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| cytochrome c oxidase subunit I precursor
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATP
EWIWGGFSVDKATLNRRFAFHFILEPFTMVALAGVHIL
LLILLLLLLALLSPDMLGDPDNHMPADPLNTPLH
GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTIENY
```

The screenshot shows a Windows Notepad window with the title "hoge4.fa - メモ帳". The window contains the following FASTA sequence data:

```
>contig_1
CGGACAGCTCCTCGGCATCCGGAT
>contig_2
GTCTGCCTCAAGCGCCCCAAGTGGGTTGGAGGCCTAACATCGCAAGTCG
ACACTCAGTCCGGCCGTCTGGTGGCAGGGGCAGAGACCCAGCACACCCT
GTC
>contig_3
TGTAGGAGAAGGGCGGTATCAGCGTCCACTTACACGATCCGTTACTAATT
GTATGAGGTCTGGGCA
>contig_4
CGTGCTGATTCCACACAGCAGTAAACGCGGACCTCTACCTATGAACATG
```

ゲノム配列

実際のゲノム配列はここからも取得可能。Rで染色体ごとの配列長やGC含量の計算ができる。

(Rで)塩基配列解析

～NGS、RNA-seq、ゲノム、トランскриプトーム、正規化、発現変動、統計、モデル、バイオインフォマティクス～

(last modified 2015/04/28)

- ・ 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 2015/02/19)
- ・ 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 2015/02/19)

What's new

- ・ このウェブサイトと必要なツールと必要なツール (Windows) (2015/04/28)
- ・ 私の所蔵例年東方 (2015/03/28)
- ・ R本体お (2015/04/28)
- ・ MBClust (2015/04/28)

イントロ | 一般 | 配列取得 | ゲノム配列 | 公共DBから

- ・ [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)
 - [Ensembl Genomes](#) (Flieck et al., Nucleic Acids Res., 2014)
 - バクテリア (Bacteria)
 - [乳酸菌 \(Lactobacillus casei 12A\)](#)
 - [乳酸菌 \(Lactobacillus casei A2-362\)](#)
 - [乳酸菌 \(Lactobacillus casei BL23\)](#)
 - [菌類 \(Fungi\)](#)
 - [後生動物 \(Metazoa\)](#)

入力と出力の関係

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

入力: hoge4.fa

The screenshot shows a Windows Notepad window titled "hoge4.fa - メモ帳". The menu bar includes "ファイル(F)", "編集(E)", "書式(O)", "表示(V)", and "ヘルプ(H)". The main content area displays four contigs in FASTA format:

```
>contig_1
CGGACAGCTCCTCGGCATCCGGAT
>contig_2
GTCTGCCTCAAGCGCCCCAAGTGGGTTGGAGGCCTAACATCGCAAGTCG
ACACTCAGTCCGGCCGTCTGGTGGCAGGGGCAGAGACCCAGCACACCCCT
GTC
>contig_3
TGTAGGAGAAGGGCGGTATCAGCGTCCACTTACACGATCCGTTACTAATT
GTATGAGGTCGGGCA
>contig_4
CGTGCTGATTCCACACAGCAGTAAACGCGGACCTCTACCTATGAACATG
```

出力: hoge1.txt

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

基本情報取得

- ・ 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 | [TranscriptDb](#) | について (last modified 2014/03/28)
- ・ イントロ | NGS | アノテーション情報取得 | [TranscriptDb](#) | [TxDb.*から](#) (last modified 2015/02/1)
- ・ イントロ | NGS | アノテーション情報取得 | [TranscriptDb](#) | [GenomicFeatures\(Lawrence 2013\)](#) (
- ・ イントロ | NGS | アノテーション情報取得 | [TranscriptDb](#) | [GFF/GTF形式ファイルから](#) (last mo
- ・ イントロ | NGS | 読み込み | FASTA形式 | [基本情報を取得](#) (last modified 2014/08/18)
- ・ イントロ | NGS | 読み込み | FASTA形式 | [description行の記述を整形](#) (last modified 2014/04/05)
- ・ イントロ | NGS | 読み込み | [FASTQ形式](#) (last modified 2014/07/17)
- ・ イントロ | NGS | 読み込み | FASTQ形式 | [description行の記述を整形](#) (last modified 2014/08/21)
- ・ イントロ | NGS | 読み込み | [II](#)
- ・ イントロ | NGS | 読み込み | [II](#)
- ・ イントロ | ファイル形式の変換
- ・ イントロ | クオリティチェック | [II](#)
- ・ イントロ | フィルタリング | [PHR](#)

①ここです。コードの最初のほうに入力ファイルと出力ファイルを記述するので、コピペで実行した結果としてどういう名前のファイルがoutputされるべきかわかる。[hoge4.fa](#)はhogeフォルダ中にもありますが、②ここからも右クリックでダウンロードできます。



イントロ | NGS | 読み込み | FASTA形式 | 基本情報を取得

multi-FASTAファイルを読み込んで、Total lengthやaverage lengthなどの各種情報取得を行うためのやり方を示します。
「ファイル」→「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

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

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

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

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

#本番(基本情報取得)
Total_len <- sum(width(fasta)) #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
```

getwdとlist.files

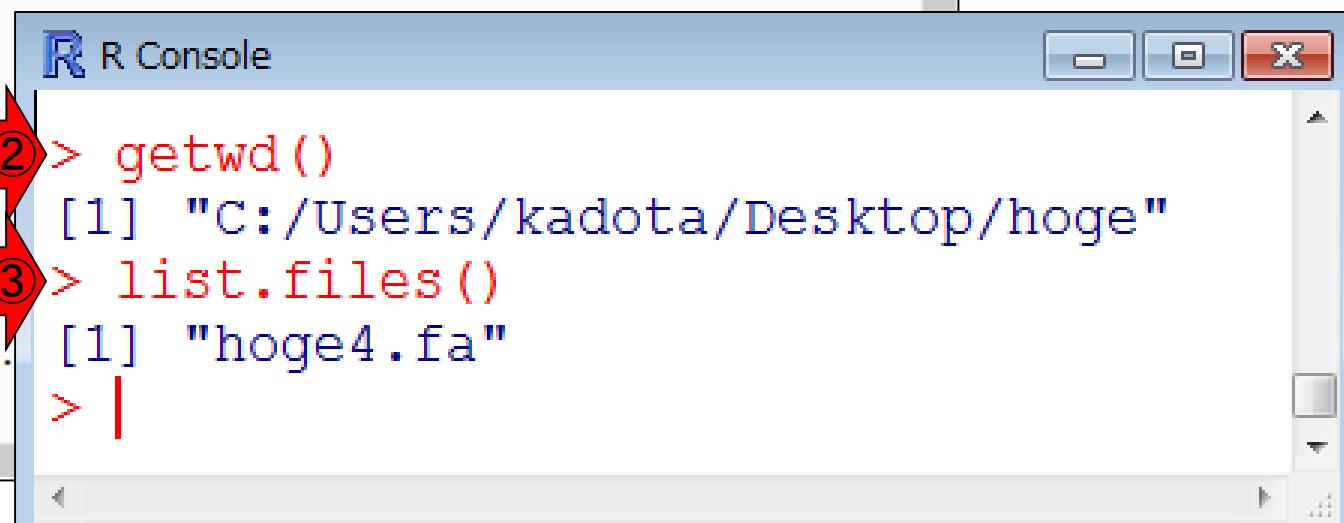
①例題の入力ファイル(hoge4.fa)をダウンロード。
②R上で作業ディレクトリの確認。③作業ディレクトリに解析したい入力ファイルがあることを確認。

イントロ | NGS | 読み込み | FASTA形式 | 基本情報を取得

multi-FASTAファイルを読み込んで、Total lengthやaverage lengthなどの各種情報取得を行うためのやり方を示します。
「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

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

```
in_f <- "hoge4.fa"          #入力ファイル名を指定してin_fに格納  
out_f <- "hoge1.txt"        #出力ファイル名を指定してout_fに格納  
  
#必要なパッケージをロード  
library(Biostrings)          #パッケージの読み込み  
  
#入力ファイルの読み込み  
fasta <- readDNAStringSet(in_f, format="fasta") #in_fで指定したファイルの読み込み  
  
#本番(基本情報取得)  
Total_len <- sum(width(fasta))  
Number_of_contigs <- length(fasta)  
Average_len <- mean(width(fasta))  
Median_len <- median(width(fasta))  
Max_len <- max(width(fasta))  
Min_len <- min(width(fasta))  
  
#本番(N50情報取得)  
sorted <- rev(sort(width(fasta)))  
obj <- (cumsum(sorted) >= Total_len*0.5)  
N50 <- sorted[obj][1]
```



コピーペースト

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

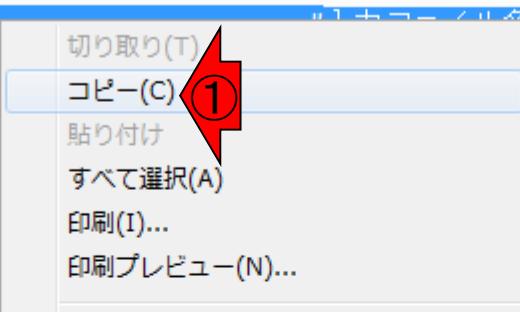
```
in_f <- "hoge4.fa"
out_f <- "hoge1.txt"

#必要なパッケージを読み込む
library(Biostrings)

#入力ファイルの読み込み
fasta <- readDNAString()

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

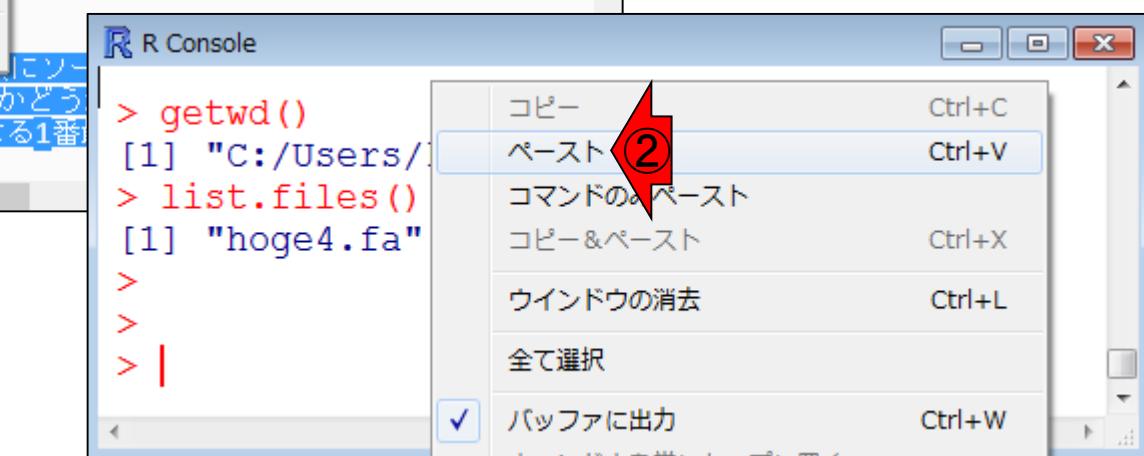
#本番(N50情報取得)
sorted <- rev(sort(width(fasta), decreasing = TRUE)) #ソートする
obj <- (cumsum(sorted) >= Total_len*0.5) #条件を満たすかどうか
N50 <- sorted[obj][1] #objがTRUEとなる1番目
```



「`in_f`」を指定して `in_f` に格納
「`out_f`」を指定して `out_f` に格納

で指定したファイルの読み込み

「`Total_len`」を取得
「`Number_of_contigs`」を取得
「`Average_len`」を取得
「`Median_len`」を取得
「`Max_len`」を取得
「`Min_len`」を取得



- 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。
ブラウザがInternet Explorerの場合は、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。

実行結果

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

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

```

in_f <- "hoge4.fa"
out_f <- "hoge1.txt" #②

#必要なパッケージをロード
library(Biostrings)

#入力ファイルの読み込み
fasta <- readDNAStringSet(in_f, format="fasta")#③

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

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

R Console

```

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

実行結果

出力ファイルをテキストエディタやExcelで眺めてもよいが、オブジェクトtmpの中身を出力しているだけなので①R上で眺めている。

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

```

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

#本番(GC含量情報取得)
hoge <- alphabetFrequency(fasta)
CG <- rowSums(hoge[,2:3])
ACGT <- rowSums(hoge[,1:4])
GC_content <- sum(CG)/sum(ACGT)

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

```

R Console

```

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

```

実行結果

contig_1の配列が最短、contig_2の配列が最長であることがわかる。入力と出力の関係を確認。

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
GTCTGCCTCAAGCGCCCCAAGTGGGTTGGAGGCCTAACATCGCAAGTCG
ACACTCAGTCCGGCCGTCTGGTGGCAGGGGCAGAGACCCAGCACACCCCT
GTC
>contig_3
TGTAGGAGAAGGGCGGTATCAGCGTCCACTTACACGATCCGTTACTAATT
GTATGAGGTCGGGCA
>contig_4
CGTGCTGATTCCACACAGCAGTAAACGCAGGACCTCTACCTATGAACATG
```

ID	Length
contig_1	24
contig_2	103
contig_3	65
contig_4	49

出力: hoge1.txt

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

N50

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。

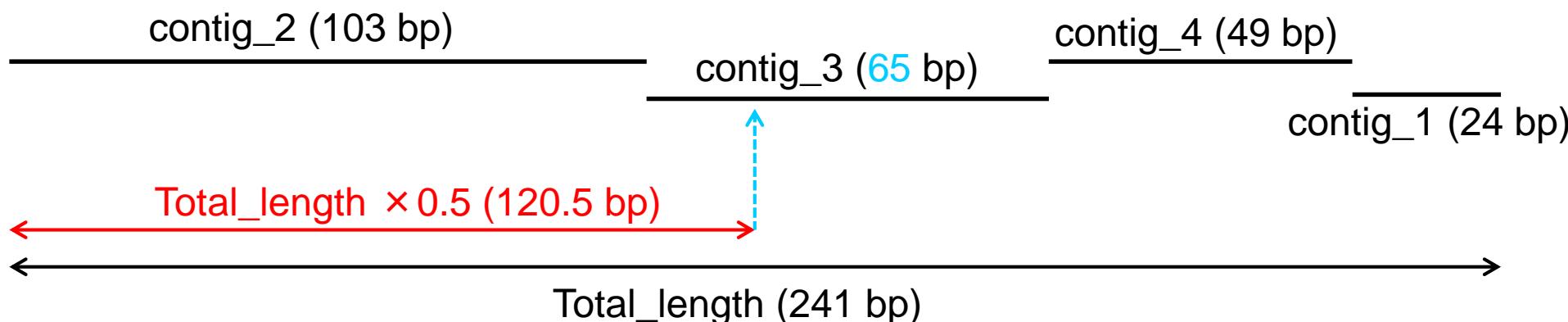
■ アセンブル結果の評価基準の一つ

- 長いコンティグから足していくと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

出力: hoge1.txt

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



コード内部の説明

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

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

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

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

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

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

```
R Console
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files()
[1] "hoge4.fa"
> in_f <- "hoge4.fa"          #入力ファイル名$ 
> out_f <- "hogel.txt"        #出力ファイル名$ 
>
> #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
> |
```

コード内部の説明

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

```

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

#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

#本番(基本情報取得)
Total_len <- sum(width(fasta))    #コンティグの「トータル長」
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」
Median_len <- median(width(fasta)) #コンティグの「中央値」
Max_len <- max(width(fasta))      #コンティグの「最大値」
Min_len <- min(width(fasta))      #コンティグの「最小値」

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

```

入力ファイル情報を格納したものが
fastaオブジェクト。widthの位置にあるのがコンティグごとの配列長情報。

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
GTCTGCCTCAAGCGCCCCAAAGTGGGTTTGAGGCCTAACATCGCAAGTCG
ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCCT
GTC
>contig_3
TGTAGGAGAAGGGCGGTATCAGCGTCCACTTACACGATCCGTTACTAATT
GTATGAGGTGGGCA
>contig_4
CGTGCTGATTCCACACAGCAGTAAACGCGGACCTCTACCTATGAACATG

```

R Console

```

> fasta <- readDNAStringSet(in_f, format="fasta")#in_fで指定
> fasta
A DNAStringSet instance of length 4
  width seq
[1] 24 CGGACAGCTCCTCGGCATCCGGAT contig_1
[2] 103 GTCTGCCTCAAG...GCACACCCCTGTC contig_2
[3] 65 TGTAGGAGAAGG...TGAGGTGGGCA contig_3
[4] 49 CGTGCTGATTCC...CCTATGAACATG contig_4
> width(fasta)
[1] 24 103 65 49
> sum(width(fasta))
[1] 241
>

```

コード内部の説明

width(fasta)にsum関数を適用すれば、
トータルの配列長(配列長の総和)になる。

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

```

in_f <- "hoge4.fa"          #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))
Min_len <- min(width(fasta))

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

```

```

R Console
> fasta <- readDNAStringSet(in_f, format="fasta")#in_fで$ 
> fasta
A DNAStringSet instance of length 4
      width seq                               names
[1]   24 CGGACAGCTCCTCGGCATCCGGAT contig_1
[2]  103 GTCTGCCTCAAG...GCACACCCCTGTC contig_2
[3]   65 TGTAGGAGAAAGG...TGAGGGTCGGGCA contig_3
[4]   49 CGTGCTGATTCC...CCTATGAACATG contig_4
> width(fasta)
[1] 24 103 65 49
> sum(width(fasta))
[1] 241
>

```

コード内部の説明

length関数は要素数を返す。この場合、fastaオブジェクトの要素数(つまりコンティグ数)を返す。

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

```

in_f <- "hoge4.fa"          #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="")

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

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

```

R Console
> fasta
A DNAStringSet instance of length 4
width seq names
[1] 24 CGGACAGCTCCTCGGCATCCGGAT contig_1
[2] 103 GTCTGCCTCAAG...GCACACCCCTGTC contig_2
[3] 65 TGTAGGAGAAAGG...TGAGGTCGGGCA contig_3
[4] 49 CGTGCTGATTCC...CCTATGAACATG contig_4
> length(fasta)
[1] 4
> 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 GTCTGCCTCAAG...GCACACCCCTGTC contig_2
>
  
```

Tips: 条件判定

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

```
in_f <- "hoge4.fa"          #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)

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

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

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

R Console

```
> fasta
A DNAStringSet instance of length 4
  width seq                               names
[1]    24 CGGACAGCTCCTCGGCATCCGGAT contig_1
[2]   103 GTCTGCCTCAAG...GCACACCCCTGTC contig_2
[3]    65 TGTAGGAGAAAGG...TGAGGTCGGGCA contig_3
[4]    49 CGTGCTGATTCC...CCTATGAACATG contig_4
> width(fasta) > 100
[1] FALSE TRUE FALSE FALSE
> width(fasta) == 65
[1] FALSE FALSE TRUE FALSE
> width(fasta) >= 50
[1] FALSE TRUE TRUE FALSE
> obj <- width(fasta) >= 50
> fasta[obj]
A DNAStringSet instance of length 2
  width seq                               names
[1]   103 GTCTGCCTCAAG...GCACACCCCTGTC contig_2
[2]    65 TGTAGGAGAAAGG...TGAGGTCGGGCA contig_3
> |
```

Tips: 条件判定

コードの中身が分かると応用範囲が飛躍的に増大。一定以上のスキルをもつバイオインフォマティシャンは「例題を探す」よりも「自分で作る」ヒトのほうが多いかも...。

- 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 2014/08/21)
- 前処理 | フィルタリング | 指定した長さの範囲の 配列を 抽出 (last modified 2015/02/06)
- 前処理 | フィルタリング | 任意の ID で フィルタリング (last modified 2015/02/06)
- 前処理 | フィルタリング | Illumina の 適用範囲 (last modified 2015/02/06)
- 前処理 | フィルタリング | GFF/GTF の 適用範囲 (last modified 2015/02/06)
- 前処理 | フィルタリング | 組合せ | ACGT の 適用範囲 (last modified 2015/02/06)
- 前処理 | トリミング | ポリA配列除去 (last modified 2015/02/06)
- 前処理 | トリミング | アダプター配列 (last modified 2015/02/06)
- 前処理 | トリミング | アダプター配列 (last modified 2015/02/06)
- 前処理 | トリミング | アダプター配列 (last modified 2015/02/06)
- 前処理 | トリミング | アダプター配列 (last modified 2015/02/06)
- 前処理 | トリミング | 指定した末端 (last modified 2015/02/06)
- アセンブル | について (last modified 2015/02/06)
- アセンブル | ゲノム用 (last modified 2015/02/06)

前処理 | フィルタリング | 指定した長さ以上の配列を抽出

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. multi-FASTAファイル(hoge4.fa)の場合:

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

```

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

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

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

#本番
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fastalに格納
#確認してるだけです

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

```

Contents

■ R基礎(初級)

- おさらい
- コード内部の説明(ファイルの読み込み、行列演算の基礎)
- リアルRNA-seqカウントデータ(数値行列データ)

■ R各種パッケージ(中級): 代表的なパッケージの利用法

- (パッケージのインストール法)
- 基本情報取得(コンティグ数、配列長、N50、GC含量)
- 任意の領域の切り出し
- GC含量計算部分の説明

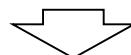
任意の領域の切り出し

subseq関数を用いて、任意の領域の配列を切り出すことができます。

- ・ イントロ | 一般 | ランダムな塩基配列を生成 (last modified 2014/06/16)
- ・ イントロ | 一般 | 任意の長さの可能な全ての塩基配列を作成 (last modified 2015/02/19)
- ・ イントロ | 一般 | 任意の位置の塩基を置換 (last modified 2013/09/12)
- ・ イントロ | 一般 | 指定した範囲の配列を取得 (last modified 2015/04/06) NEW
- ・ イントロ | 一般 | 指定したID(染色体やdescriptor)の配列を取得 (last modified 2014/03/10)
- ・ イントロ | 一般 | 翻訳配列(translate)を取得(基礎)
- ・ イントロ | 一般 | 翻訳配列(translate)を取得(応用)
- ・ イントロ | 一般 | 相補鎖(complement)を取得 (last modified 2014/03/10)
- ・ イントロ | 一般 | 逆相補鎖(reverse complement)を取得 (last modified 2014/03/10)
- ・ イントロ | 一般 | 逆鎖(reverse)を取得 (last modified 2014/03/10)
- ・ イントロ | 一般 | 2連続塩基の出現頻度情報を取得 (last modified 2014/03/10)
- ・ イントロ | 一般 | 3連続塩基の出現頻度情報を取得 (last modified 2014/03/10)

入力: sample1.fasta

```
>kadota  
AGTGACGGTCTT
```



出力: hoge1.fasta

```
>kadota  
TGACGGT
```

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。この項目は、「この染色体の、ここから、ここまで」という指定の仕方になります。例えば入力ファイルがヒトゲノムだった場合に、chr3の20000から500000の座標の配列取得を行いたい場合などに利用します。したがって、chr4とchr8の配列のみ抽出といったやり方には対応していませんのでご注意ください。また、ファイルダウンロード時に、*.fastaという拡張子が*.txtに勝手に変更されることがありますのでご注意ください。
「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)の配列を抽出するやり方です。

```
in_f <- "sample1.fasta"  
out_f <- "hoge1.fasta"  
param <- c(3, 9) #入力ファイル名を指定してin_fに格納  
#出力ファイル名を指定してout_fに格納  
#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  
#確認してます  
  
#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  
#確認してます  
  
#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
```

Tips: 関数のオプション

subseq関数実行時に、①数値を直接指定してもいいし、②オプション名を明記してもよい。

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.fasta"          #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)

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

#本番
fasta <- subseq(fasta, param[1], param[2]) #確認します
fasta

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

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

```
> #入力ファイルの読み込み
> fasta <- readDNAStringSet(in_f, format="fasta") #確認します
> 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
> subseq(fasta, 3, 9)
A DNAStringSet instance of length 1
width seq names
[1] 7 TGACGGT kadota
> subseq(fasta, start=3, end=9)
A DNAStringSet instance of length 1
width seq names
[1] 7 TGACGGT kadota
>
```

入力: sample1.fasta

```
>kadota
AGTGACGGTCTT
```

出力: hoge1.fasta

```
>kadota
TGACGGT
```

①

②

Tips: 関数のオプション

①原因既知状態でエラーを出す。②「3番目の位置から5塩基分抽出」という他のオプション(endではなくwidth)を利用する。

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)の配列を抽出するやり方です。

```
in_f <- "sample1.fasta"
out_f <- "hoge1.fasta"
param <- c(3, 9)

#必要なパッケージをロード
library(Biostrings)

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

#本番
fasta <- subseq(fasta, param[1], param[2])#param[1]とparam[2]を確認
fasta #確認用

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

#入力ファイル名を指定してin_fに格納
#出力ファイル名を指定してout_fに格納
#抽出したい範囲の始点と終点を指定



R Console

```
> subseq(fasta, start=3, end=12)
A DNAStringSet instance of length 1
  width seq                               names
[1] 10 TGACGGTCTT                         kadota
> subseq(fasta, start=3, end=13)
以下にエラー .Call2("solve_user_SEW", refwidths, $ 
  solving row 1: 'allow.nonnarrowing' is FALSE and$ 
> subseq(fasta, start=3, width=5)
A DNAStringSet instance of length 1
  width seq                               names
[1] 5  TGACG                            kadota
> |
```

入力: sample1.fasta

```
>kadota
AGTGACGGTCTT
```

出力: hoge1.fasta

```
>kadota
TGACGGT
```

Tips: 関数の使用法

R R Core ①

```
> ?subseq
starting httpd help server ... done
> |
```

①「?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、慣れておこう。

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](#)]

任意の領域の切り出し

- ・ イントロ | 一般 | ランダムな塩基配列を生成 (last modified 2014/06/16)
- ・ イントロ | 一般 | 任意の長さの可能な全ての塩基配列を作成 (last modified 2015/02/19)
- ・ イントロ | 一般 | 任意の位置の塩基を置換 (last modified 2013/09/12)
- ・ イントロ | 一般 | 指定した範囲の配列を取得 (last modified 2015/04/06) NEW
- ・ イントロ | 一般 | 指定したID(染色体やdescriptor)の配列を取得 (last modified 2014/03/10)

イントロ | 一般 | 指定した範囲の配列を取得 NEW

single-FASTA形式やmulti-FASTA形式では、「この染色体の、ここから、場合に、chr3の20000から5000000chr8の配列のみ抽出といったやり方に、*.fastaという拡張子が*.txtに」「ファイル」-「ディレクトリの変更」

1. (single-)FASTA形式ファイル

任意の範囲(始点が3, 終点が9)

```
in_f <- "sample1.fasta"
out_f <- "hoge1.fasta"
param <- c(3, 9)

#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_f1, format="fasta")#in_f1で指定したファイルの読み込み
posi <- read.table(in_f2)
fasta
```

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、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"
in_f2 <- "list_sub2.txt"
out_f <- "hoge4.fasta"

#必要なパッケージをロード
library(Biostrings)

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

#本番
hoge <- NULL
for(i in 1:nrow(posi)){
  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
}
fasta <- hoge
fasta

#ファイルに保存
```

#入力ファイル名を指定してin_f1に格納(multi-FASTA)
#入力ファイル名を指定してin_f2に格納(リストファイル)
#出力ファイル名を指定してout_fに格納

#パッケージの読み込み
#最終的に得る結果を格納するためのプレースホルダー
#length(posi)回だけループを回す
#条件を満たすかどうかを判定した結果をobjに格納
#hogeの中身をfastaに格納
#確認しているだけです

入力がmulti-FASTAファイル(hoge4.fa)で、リストファイル(list_sub2.txt)で指定した複数領域を切り出したい場合

contig_4	1	10
contig_2	3	8

任意の領域の切り出し

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"
in_f2 <- "list_sub2.txt" ←
out_f <- "hoge4.fasta"

#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
                                                    #確認してるだけです

#本番
hoge <- NULL
for(i in 1:nrow(posi)){
  obj <- names(fasta) == posi[i,1]
  hoge <- append(hoge, subseq(fasta[obj])
}
fasta <- hoge
fasta                                         #hogeの中身をfas$ #確認してるだけ$

#ファイルに保存
<
```

入力2: [list_sub2.txt](#)

contig_4	1	10
contig_2	3	8

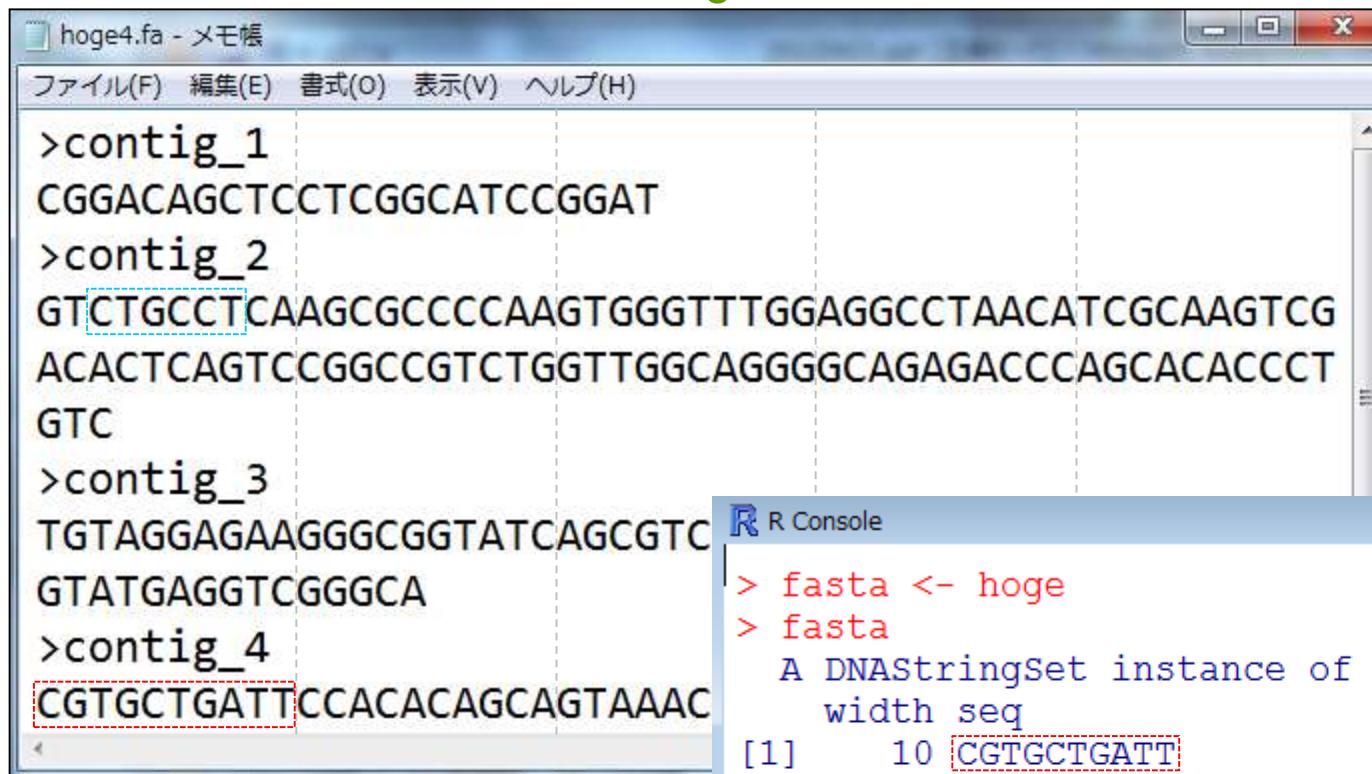
R Console window showing the execution of R code to extract DNA sequences from hoge4.fasta based on positions specified in list_sub2.txt.

```
R Console
> fasta <- hoge
> fasta
A DNAStringSet instance of length 2
  width seq
[1]    10 CGTGCTGATT
[2]     6 CTGCCT
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", widths=c(10, 6))
> |
```

The R console output shows the extraction of two DNA sequences from the multi-FASTA file 'hoge4.fasta'. The first sequence has a width of 10 and the second has a width of 6. The extracted sequences are then saved to a file 'out_f' in FASTA format.

任意の領域の切り出し

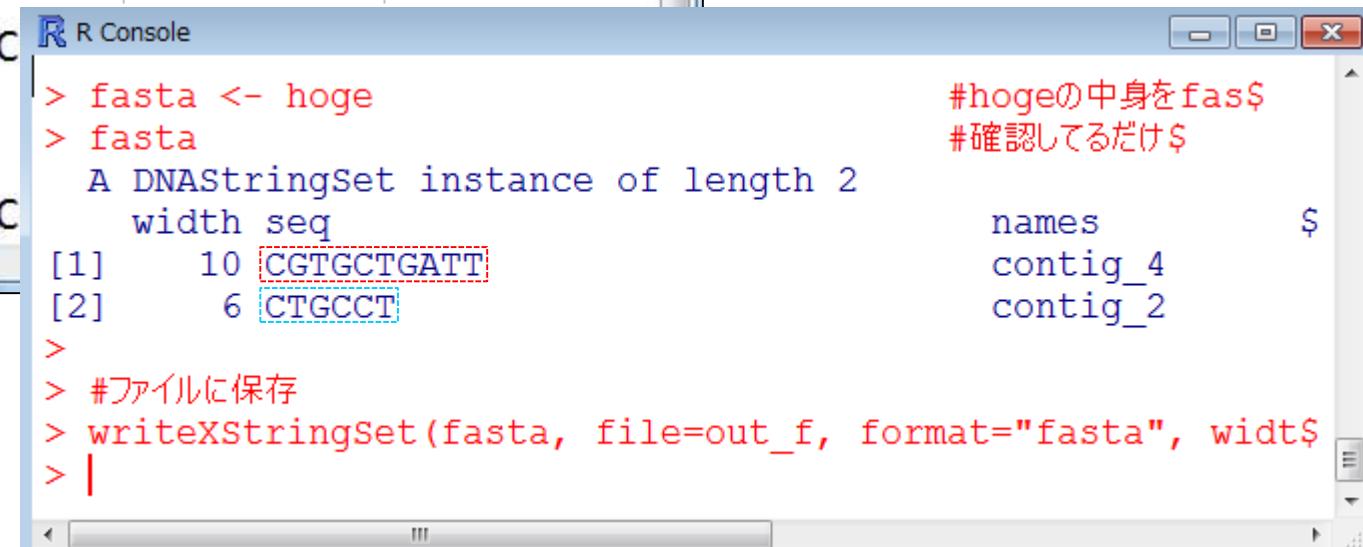
入力1: hoge4.fa



```
>contig_1
CGGACAGCTCCTCGGCATCCGGAT
>contig_2
GTCCTGCCTCAAGCGCCCCAAGTGGGTTGGAGGCCTAACATCGCAAGTCG
ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCCT
GTC
>contig_3
TGTAGGAGAAAGGGCGGTATCAGCGTC
GTATGAGGTCGGGCA
>contig_4
CGTGCTGATTCCACACACAGCAGTAAAC
```

入力2: list_sub2.txt

contig_4	1	10
contig_2	3	8



```
R Console
> fasta <- hoge
> fasta
A DNAStringSet instance of length 2
  width seq
[1]    10 CGTGCTGATT
[2]      6 CTGCCT
>
> #ファイルに保存
> writeXStringSet(fasta, file=out_f, format="fasta", width=10)
>
```

#hogeの中身をfas\$
#確認してるだけ\$

names	\$
contig_4	
contig_2	

FastQCと同じ結果を得る

FastQC Report

Summary

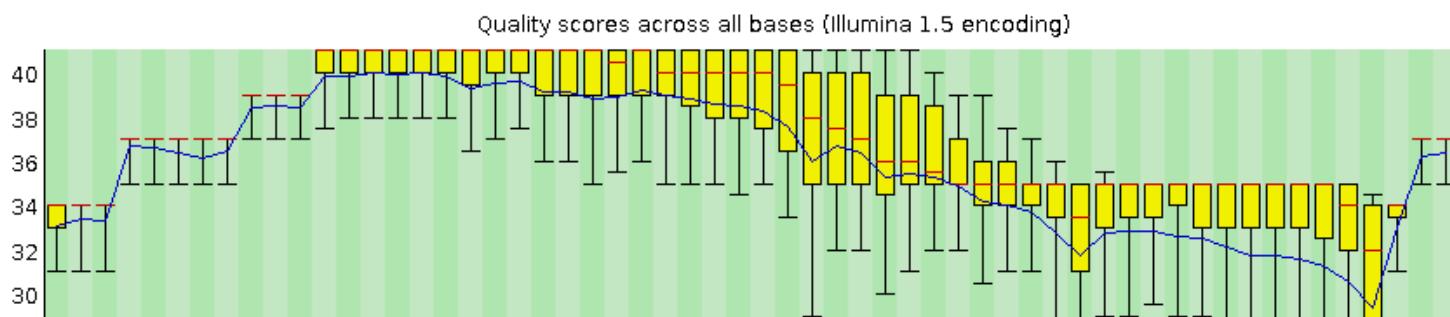
- Basic Statistics
- Per base sequence quality
- Per tile sequence quality
- Per sequence quality scores
- Per base sequence content
- Per sequence GC content
- Per base N content
- Sequence Length Distribution
- Sequence Duplication Levels
- Overrepresented sequences
- Adapter Content
- Kmer Content

Basic Statistics

Measure	Value
Filename	SRR616268sub_1.fastq.gz
File type	Conventional base calls
Encoding	Illumina 1.5
Total Sequences	1000000
Sequences flagged as poor quality	0
Sequence length	107
%GC	50

①100万リード、②107bpからなる③乳酸菌RNA-seqデータのFastQC解析結果のうち、例えば④のOverrepresented sequencesと同じ結果をsubseqとtable関数を使って得ることができます。

Per base sequence quality



FastQCと同じ結果を得る

FastQC Report

Summary

- Basic Statistics
- Per base sequence quality
- Per tile sequence quality
- Per sequence quality scores
- Per base sequence content
- Per sequence GC content
- Per base N content
- Sequence Length Distribution
- Sequence Duplication Levels
- Overrepresented sequences
- Adapter Content
- Kmer Content

Overrepresented sequences

Sequence	Count	Percentage	Possible Source
CCCCGGTATATTTCGGCGCAGTGCCACTCGACTAGTGAGCTATTACGCA	14383	1.4383	No Hit
GGCCTATTCACTGCGGCTGACCTTGCAGTCAGCACCCCTTCTCCGAAGT	11044	1.1044	No Hit
GTGCTTTCACCTTCCCTCACGGTACTGGTTCACTATCGGTCACTAGGG	8892	0.8892000000000001	No Hit
CCCGGTATATTTCGGCGCAGTGCCACTCGACTAGTGAGCTATTACGCAC	8474	0.8474	No Hit
GTCACTAGGGAGTATTTAGCCTGGGAGATGGTCCTCCCGGATTCCGACG	8189	0.8189	No Hit
GCGGGCATTCTCACTTCTAAGCGCTCCAGCCGTCTCACGATCGACCTTC	8132	0.8132	No Hit
GTCCAGTCTACAACCCCGAGAAGCAAGCTTCGGTTGGCTTCCC	6663	0.6663	No Hit
GTCGGTTGCGGTACGGTAGTTATTCTCACTAGAAAGCTTCTGGC	6411	0.6411	No Hit
GGTCACTTGGTTTGGGTCTACATCTGCTTACTCATCGCCCTGTTAGA	5502	0.5502	No Hit
GCGGGCATTCTCACTTCTAAGCGCTCCAGCCGTCTCACGATCAACCTTC	4845	0.4845000000000004	No Hit
CCCTCCATCGCTAAACAAAATAAAACTAGTGCAGGAATCTAACCTGCTT	4395	0.4394999999999995	No Hit
CCGGTATATTTCGGCGCAGTGCCACTCGACTAGTGAGCTATTACGCACT	4385	0.4385	No Hit
CCCGCGTCTGCCGCCGCCAGCTATGTATTCACTGACAAGCAATACACTG	4366	0.4366	No Hit
CCACAGTTGGTATTATGCTTAGCCCCGGTATATTTCGGCGCAGTGC	4314	0.4314	No Hit
CTGGGCTGTTCCCTTCAACAATGGACCTTATCGCTCACTGTCTGACTC	4113	0.4113000000000005	No Hit
CCGCCGTACTCAGGATCCTGGACGGAGGGTTCGACGTTCGCTTACAGGG	4081	0.4081	No Hit
CCGGCATTCTCACTTCTAAGCGCTCCAGCCGTCTCACGATCGACCTCA	3846	0.3846	No Hit
GTAGGTCACTTGGTTGGGTCTACATCTGCTTACTCATTGCCCTGTT	3823	0.3823	No Hit

result_without_nogroup.html

①頻出する配列をリストアップ。②トップは「CCCCGGTATA…」という50塩基の配列で14,383回出現。Percentageは1.4383%。全部で100万リードなので妥当。オリジナル107 bpのうち最初の50 bpで解析している。

subseq関数を使っています。やってみましょう。

Overrepresented seq.

- 前処理 | クオリティコントロール |について (last modified 2015/06/25) NEW
- 前処理 | クオリティチェック | QuasR(Gaidatzis 2015) (last modified 2015/06/15)
- 前処理 | クオリティチェック | qrqc (last modified 2014/07/17)
- 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 2015/06/22) NEW
- 前処理 | クオリティチェック | Overrepresented sequences | ShortRead(Morgan 2009)
- 前処理 | トリミング | ポリA配列除去 | ShortRead(Morgan 2009) (last modified 2014/06/20)
- 前処理 | トリミング | アダプター配列除去(基礎) | QuasR(Gaidatzis 2015) (last modified 2015/06/15)
- 前処理 | トリミング | フィルタリング | ShortRead(Morgan 2009) (last modified 2015/06/22)
- 前処理 | トリミング | フィルタリング | Overrepresented sequences | ShortRead(Morgan 2009) NEW
- 前処理 | トリミング | フィルタリング | ShortRead(Morgan 2009) (last modified 2015/06/22)
- 前処理 | トリミング | フィルタリング | Overrepresented sequences | ShortRead(Morgan 2009) NEW

①

前処理 | クオリティチェック | Overrepresented sequences | ShortRead(Morgan_2009) NEW

ShortReadパッケージを用いてリードの種類ごとの出現回数を得るやり方を示します。[FastQCの Overrepresented sequences の項目](#)と同じ結果が得られます。[前処理 | フィルタリング | 重複のない配列セットを作成](#)も基本的にやっていることは同じです。

4. gzip圧縮FASTQ形式ファイル(SRR616268sub_1.fastq.gz)の場合

乳酸菌RNA-seqデータSRR616268の最初の100万リード分(約73MB)です。長さは全て107 bpです。3と基本的に同じですが、FastQCのデフォルトと同じく、最初の50bp分のみで解析するやり方です。「[イントロ | 一般 | 指定した範囲の配列を取得](#)」のテクニックとの併用になります。

1. gzip圧縮

small RNA-seq

```
in_f <- "SRR616268sub_1.fastq.gz"
out_f <- "hoge4.txt"
library(ShortRead)

#入力ファイルをロード
fastq <- readFastq(in_f)
fasta <- sread(fastq)
fasta

#前処理(指定した範囲のみ抽出)
fasta <- subseq(fasta, start=param[1], end=param[2]) #paramで指定した始点と終点の範囲のみ抽出
```

②



```
R Console
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files(pattern="SRR")
[1] "SRR616268sub_1.fastq.gz"
#確認
```

#paramで指定した始点と終点の範囲のみ抽出

完璧に同じ結果を得られていることが分かります

Overrepresented seq.

4. gzip圧縮FASTQ形式ファイル(SRR616268sub_1.fastq.gz)の場合:

乳酸菌RNA-seqデータSRR616268の最初の100万リード分(約73MB)です。長さは全て107 bpです。3.と基本的に同じですが、FastQCのデフォルトと同じく、最初の50bp分のみで解析するやり方です。「イントロ | 一般 | 指定した範囲の配列を取得」のテクニックとの併用になります。

```
in_f <- "SRR616268sub_1.fastq.gz"      #入力ファイル名を指定してin_fに格納
out_f <- "hoge4.txt"                     #出力ファイル名を指定してout_fに格納
param <- c(1, 50)                         ##指定した範囲の配列を取得

#必要なパッケージをロード
library(ShortRead)

#入力ファイルの読み込み
fastq <- readFastq(in_f)                 #i
fasta <- sread(fastq)                     #f
fasta

#前処理(指定した範囲のみ抽出)
fasta <- subseq(fasta, start=param[1], end=param[2]) ##抽出
fasta

#本番
out <- table(fasta)                      #i
out <- sort(out, decreasing=T)            #上位順
head(out)                                 ##確認

#ファイルに保存
```

R Console

```
> head(out)                                #確認してだけ
  fasta
  CCCCGGTATATTTCGGCGCAGTGCCACTCGACTAGTGAGCTATTACGCA
  14383
  GGCCTATTCACTGCGGCTGACCTTGCAGTCAGCACCCCTTCTTCCGAAGT
  11044
  GTGCTTTCACCTTCCCTCACGGTACTGGTTCACTATCGGTCACTAGGG
  8892
  CCCGGTATATTTCGGCGCAGTGCCACTCGACTAGTGAGCTATTACGCAC
  8474
  GTCACTAGGGAGTATTAGCCTTGGGAGATGGTCCTCCGGATTCCGACG
  8189
  GCCGGCATTCTCACTTCTAAGCGCTCCAGCCGTCCACGATCGACCTTC
  8132
>
> #ファイルに保存
> tmp <- cbind(names(out), out)           #保存したい情報
> write.table(tmp, out_f, sep="\t", append=F, quote=F, row.names=F)
>
```



Contents

■ R基礎(初級)

- おさらい
- コード内部の説明(ファイルの読み込み、行列演算の基礎)
- リアルRNA-seqカウントデータ(数値行列データ)

■ R各種パッケージ(中級): 代表的なパッケージの利用法

- (パッケージのインストール法)
- 基本情報取得(コンティグ数、配列長、N50、GC含量)
- 任意の領域の切り出し
- GC含量計算部分の説明

GC含量計算部分の説明

右のサイドバーを下に移動させるとGC含量計算部分を見られる。

- ・ イントロ | NGS | アノテーション情報取得 | TranscriptDb | について (last modified 2014/03/28)
- ・ イントロ | NGS | アノテーション情報取得 | TranscriptDb | TxDb.*から (last modified 2015/02/19)
- ・ イントロ | NGS | アノテーション情報取得 | TranscriptDb | GenomicFeatures(Lawrence 2013) (last modified 2014/08/18)
- ・ イントロ | NGS | アノテーション情報取得 | TranscriptDb | GFF/GTF形式ファイルから (last modified 2014/08/18)
- ・ イントロ | NGS | 読み込み | FASTA形式 | 基本情報を取得 (last modified 2014/08/18)
- ・ イントロ | NGS | 読み込み | FASTA形式 | description行の記述を整形 (last modified 2014/04/05)
- ・ イントロ | NGS | 読み込み | FASTQ形式
- ・ イントロ | NGS | 読み込み | FASTQ形式
- ・ イントロ | NGS | 読み込み | Illuminaの*
- ・ イントロ | NGS | 読み込み | Illuminaの*
- ・ イントロ | ファイル形式の変換 | について
- ・ イントロ | ファイル形式の変換 | BAM -->
- ・ イントロ | ファイル形式の変換 | FASTQ ->
- ・ イントロ | ファイル形式の変換 | Genbank
- ・ イントロ | ファイル形式の変換 | gseq -->
- ・ イントロ | ファイル形式の変換 | gseq -->
- ・ イントロ | ファイル形式の変換 | gseq -->
- ・ 前処理 | クオリティチェック | について (la)
- ・ 前処理 | クオリティチェック | qrc (last mod)
- ・ 前処理 | クオリティチェック | PHREDスコア
- ・ 前処理 | クオリティチェック | 配列長分布
- ・ 前処理 | フィルタリング | PHREDスコアカ

イントロ | NGS | 読み込み | FASTA形式 | 基本情報を取得

multi-FASTAファイルを読み込んで、Total lengthやaverage lengthなどの各種情報取得を行うためのやり方を示します。
「ファイル」→「ディレクトリの変更」で解析したいファイルを置いてあるディレクトリに移動し以下をコピペ。

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

```

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

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

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

#本番(基本情報取得)
Total_len <- sum(width(fasta))    #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
obj <- (cumsum(sorted) >= Total_len*0.5) #条件を満たすかどうかを判定した結果をobjに格納(長い)
N50 <- sorted[obj][1]               #objがTRUEとなる1番最初の要素のみ抽出した結果をN50に

```

GC含量計算部分の説明

fastaオブジェクトを出発点として、配列全体のGC含量(57.68%)を得るところの説明です。

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

```

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

#本番(GC含量情報取得)
hoge <- alphabetFrequency(fasta)      #A,C,G,T,...の数を配列ごとにカウントした結果をhogel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 <- sum(CG)/sum(ACGT)        #トータルのGC含量を計算してGC_contentに格納

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

```

R Console

```

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

```

GC含量計算部分の説明

黒枠部分を再度コピペしたのち、①fastaオブジェクトの中身を表示させたところ。

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に格納

#本番(基本情報取得)
Total_len <- sum(width(fasta))    #コンティグの長さ
Number_of_contigs <- length(fasta) #「コンティグ数」
Average_len <- mean(width(fasta)) #コンティグの平均長
Median_len <- median(width(fasta))#コンティグの中央値
Max_len <- max(width(fasta))      #コンティグの最大長
Min_len <- min(width(fasta))      #コンティグの最小長

#本番(N50情報取得)
sorted <- rev(sort(width(fasta)))  #長さ情報を降順
obj <- (cumsum(sorted) >= Total_len*0.5) #条件を満たすか
N50 <- sorted[obj][1]              #objがTRUEとなる位置
```

①

R Console

```
> getwd()
[1] "C:/Users/kadota/Desktop/hoge"
> list.files()
[1] "hoge4.fa"
> 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")$fasta
>
A DNAStringSet instance of length 4
  width seq           names
[1]   24 CGGACAGC...TCCGGAT contig_1
[2]  103 GTCTGCCT...CCCTGTC contig_2
[3]   65 TGTAGGAG...TCGGGCA contig_3
[4]   49 CGTGCTGA...GAACATG contig_4
> |
```

GC含量計算部分の説明

alphabetFrequency関数は、
塩基ごとの出現回数を返す。

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

```

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

#本番(GC含量情報取得)
hoge <- alphabetFrequency(fasta)      #A,C,G,T,...の数を配列ごとにカウントした結果をh
CG <- rowSums(hoge[,2:3])            #C,Gの総数を計算して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 <- sum(CG)/sum(ACGT)       #トータルのGC含量の情報を取得

#ファイルに保存
tmp <- NULL
tmp <- rbind(tmp, c("Total length (bp)", Total_len))
tmp <- rbind(tmp, c("Number of contigs", Number_of_co
tmp <- rbind(tmp, c("Average length", Average_len))
tmp <- rbind(tmp, c("Median length", Median_len))
tmp <- rbind(tmp, c("Max length", Max_len))
tmp <- rbind(tmp, c("Min length", Min_len))
tmp <- rbind(tmp, c("N50", N50))
tmp <- rbind(tmp, c("GC content", GC_content))
write.table(tmp, out_f, sep="\t", append=F, quote=F,

```

R Console window showing the following session:

```

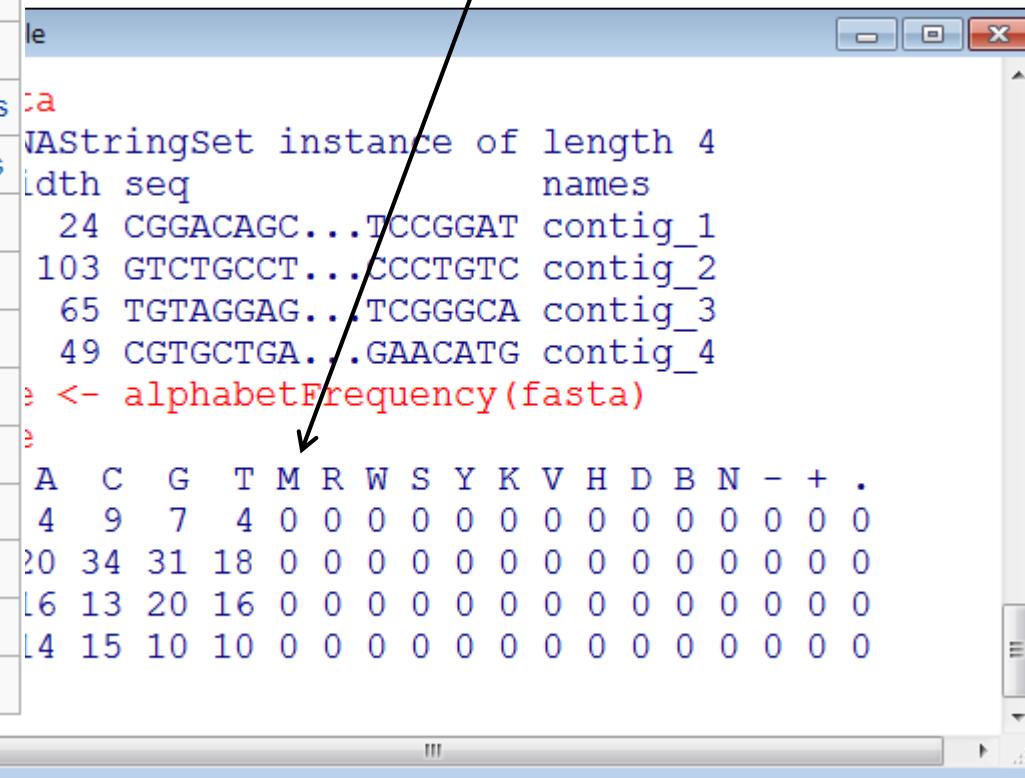
> fasta
A DNAStringSet instance of length 4
width seq names
[1] 24 CGGACAGC...TCCGGAT contig_1
[2] 103 GTCTGCCT...CCCTGTC contig_2
[3] 65 TGTAGGAG...TCGGGCA contig_3
[4] 49 CGTGCTGA...GAACATG contig_4
> hoge <- alphabetFrequency(fasta)
> hoge
      A   C   G   T   M   R   W   S   Y   K   V   H   D   B   N   -   +
[1,]  4   9   7   4   0   0   0   0   0   0   0   0   0   0   0   0   0   0
[2,] 20  34  31  18  0   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   0
[4,] 14  15  10  10  0   0   0   0   0   0   0   0   0   0   0   0   0   0
>

```

GC含量計算部分の説明

DNA配列上のMは「A or C」、Rは「A or G」などというルールがあるようです。

Nucleic Acid Code	Meaning	Mnemonic
A	A	Adenine
C	C	Cytosine
G	G	Guanine
T	T	Thymine
U	U	Uracil
R	A or G	puRine
Y	C, T or U	pYrimidines
K	G, T or U	bases which are Ketones
M	A or C	bases with aMino groups
S	C or G	Strong interaction
W	A, T or U	Weak interaction
B	not A (i.e. C, G, T or U)	B comes after A
D	not C (i.e. A, G, T or U)	D comes after C
H	not G (i.e., A, C, T or U)	H comes after G
V	neither T nor U (i.e. A, C or G)	V comes after U
N	A C G T U	Nucleic acid
X	masked	
-	gap of indeterminate length	



```

le
ca
NAStringSet instance of length 4
length seq
names
24 CGGACAGC...TCCGGAT contig_1
103 GTCTGCCT...CCCTGTC contig_2
65 TGTAGGAG...TCGGGCA contig_3
49 CGTGCTGA...GAACATG contig_4
<- alphabetFrequency(fasta)
<-
A   C   G   T   M   R   W   S   Y   K   V   H   D   B   N   -   +
4    9    7    4    0    0    0    0    0    0    0    0    0    0    0    0    0
20   34   31   18   0    0    0    0    0    0    0    0    0    0    0    0    0
16   13   20   16   0    0    0    0    0    0    0    0    0    0    0    0    0
14   15   10   10   0    0    0    0    0    0    0    0    0    0    0    0    0

```

GC含量計算部分の説明

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

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

#本番(GC含量情報取得)

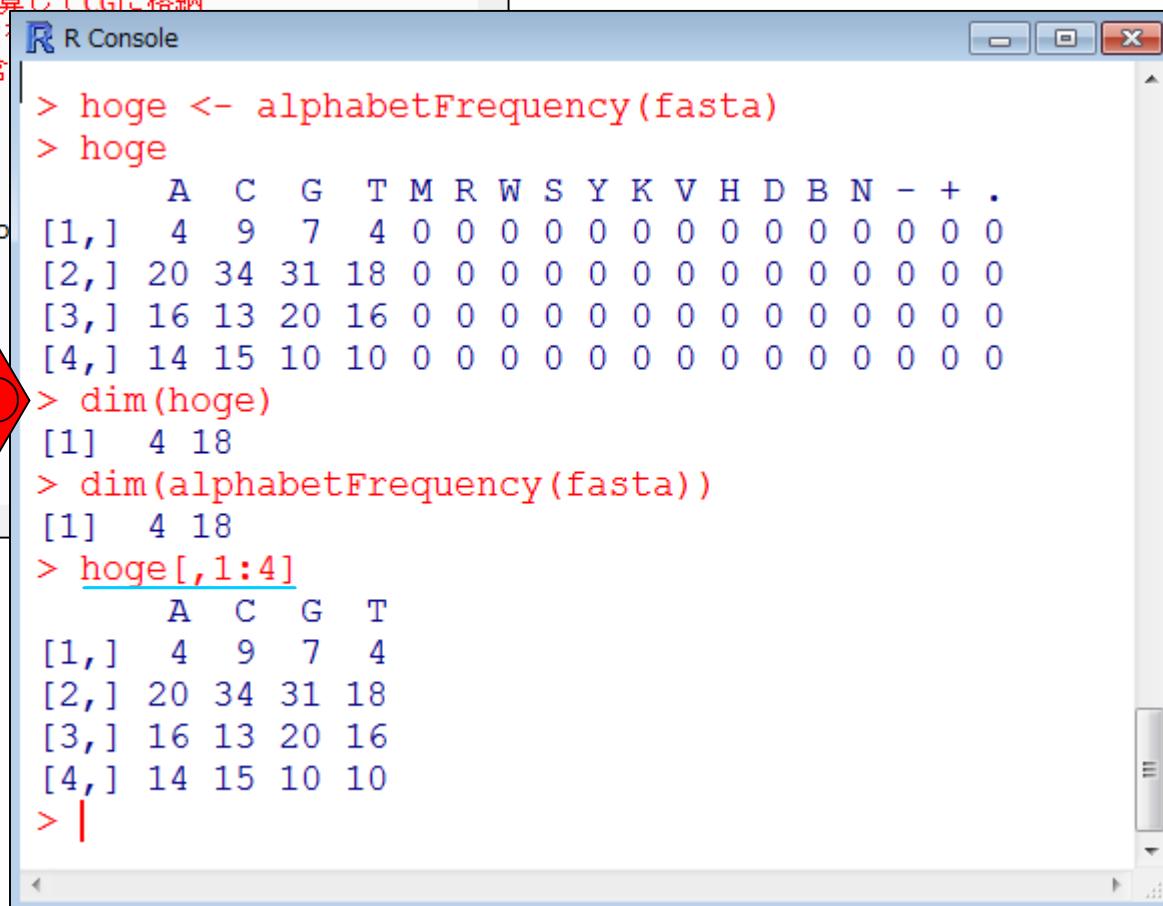
```
hoge <- alphabetFrequency(fasta)
CG <- rowSums(hoge[,2:3])
ACGT <- rowSums(hoge[,1:4])
GC_content <- sum(CG)/sum(ACGT)
```

#ファイルに保存

```
tmp <- NULL
tmp <- rbind(tmp, c("Total length (bp)", Total_len))
tmp <- rbind(tmp, c("Number of contigs", Number_of_contigs))
tmp <- rbind(tmp, c("Average length", Average_len))
tmp <- rbind(tmp, c("Median length", Median_len))
tmp <- rbind(tmp, c("Max length", Max_len))
tmp <- rbind(tmp, c("Min length", Min_len))
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,
```

#A,C,G,T,...の数を配列ごとにカウントした結果をhogeに格納
#C,Gの総数を計算してCGに格納
#A,C,G,Tの総数を計算してACGTに格納
#トータルのGC含量を計算してGC_contentに格納

①dim関数は行列の行数と列数を返す。alphabetFrequency関数出力結果は、4行×18列からなることが分かる。キーボードの上下キーを上手に利用して最小限の労力をキータイプ(あるいはコピペ)すべし!



The screenshot shows an R console window titled "R Console". The code entered is:

```
> hoge <- alphabetFrequency(fasta)
> 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 18
> dim(alphabetFrequency(fasta))
[1] 4 18
> hoge[,1:4]
      A   C   G   T
[1,]  4    9    7    4
[2,] 20   34   31   18
[3,] 16   13   20   16
[4,] 14   15   10   10
> |
```

A red circle with the number "1" is drawn around the first line of the R code: "hoge <- alphabetFrequency(fasta)".

GC含量計算部分の説明

任意のサブセットを取得可能。
2:3やc(1,4)などをうまく利用。

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:

```

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

#本番(GC含量情報取得)
hoge <- alphabetFrequency(fasta)
CG <- rowSums(hoge[,2:3])             #A,C,G,T,...の数を配列ごとにカウントした結果をh
                                         #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 <- sum(CG)/sum(ACGT)        #トータルのGC含量の情報を取得

#ファイルに保存
tmp <- NULL
tmp <- rbind(tmp, c("Total length (bp)", Total_len))
tmp <- rbind(tmp, c("Number of contigs", Number_of_contigs))
tmp <- rbind(tmp, c("Average length", Average_len))
tmp <- rbind(tmp, c("Median length", Median_len))
tmp <- rbind(tmp, c("Max length", Max_len))
tmp <- rbind(tmp, c("Min length", Min_len))
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の中身を指定し

```

```

R R Console
> hoge[, 2:3]
   C  G
[1,] 9  7
[2,] 34 31
[3,] 13 20
[4,] 15 10
> hoge[, c(1, 4)]
   A  T
[1,] 4  4
[2,] 20 18
[3,] 16 16
[4,] 14 10
> 2:3
[1] 2 3
> c(1, 4)
[1] 1 4
>

```

GC含量計算部分の説明

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

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

#本番(GC含量情報取得)

```
hoge <- alphabetFrequency(fasta)
CG <- rowSums(hoge[,2:3])           #A,C,G,T,...の数を配列ごとにカウントした結果をh
ACGT <- rowSums(hoge[,1:4])         #C,Gの総数を計算してCGに格納
GC_content <- sum(CG)/sum(ACGT)     #A,C,G,Tの総数を計算してACGTに格納
                                      #トータルのGC含量の情報を取得
```

#ファイルに保存

```
tmp <- NULL
tmp <- rbind(tmp, c("Total length (bp)", Total_len))
```

```
tmp <- rbind(tmp, hoge[1,])
tmp <- rbind(tmp, hoge[2,])
tmp <- rbind(tmp, hoge[3,])
tmp <- rbind(tmp, hoge[4,])
```

```
write(tmp, "hoge4.fa", "text")
```

```
>contig_1
CGGACAGCTCCTCGGCATCCGGAT
>contig_2
GTCTGCCTCAAGCGCCCCAAGTGGGTTTGAGGCCAACATCGCAAGTCG
ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCCT
GTC
>contig_3
TGTAGGAGAAGGGCGGTATCAGCGTCCACTTACACGATCCGTTACTAATT
GTATGAGGTCGGGCA
>contig_4
CGTGCTGATTCCACACAGCAGTAAACGCGGACCTCTACCTATGAACATG
```

黒丸中の数値はcontig_1中のAの数が4個、赤丸中の数値は、contig_4中のTの数が10個であるということ。
rowSums関数は行ごとの和を返す。

```
R Console
> hoge[,1:4]
   A   C   G   T
[1,] 4  9  7  4
[2,] 20 34 31 18
[3,] 16 13 20 16
[4,] 14 15 10 10
> rowSums(hoge[,1:4])
[1] 24 103 65 49
> age <- hoge[,1:4]
> rowSums(age)
[1] 24 103 65 49
> apply(age, 1, sum)
[1] 24 103 65 49
> |
```

GC含量計算部分の説明

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

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

#本番(GC含量情報取得)

```
hoge <- alphabetFrequency(fasta)
CG <- rowSums(hoge[,2:3])           #A,C,G,T,...の数を配列ごとにカウントした結果をhogeに格納
ACGT <- rowSums(hoge[,1:4])          #C,Gの総数を計算してCGに格納
GC_content <- sum(CG)/sum(ACGT)       #A,C,G,Tの総数を計算してACGTに格納
GC_content                         #トータルのGC含量の情報を取得
```

#ファイルに保存

```
tmp <- NULL
tmp <- rbind(tmp, c("Total length (bp)", Total_len))
tmp <- rbind(tmp, c("GC content (%)", GC_content))
```

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

rowSums関数の入力として、ACGTのみのカウント数を与えていいが、その結果(返り値)は、配列中にNなどを含まない場合は実質的にコンティグごとの配列長と同じ。

```
R Console
> hoge[,1:4]
  A  C  G  T
[1,]  4  9  7  4
[2,] 20 34 31 18
[3,] 16 13 20 16
[4,] 14 15 10 10
> rowSums(hoge[,1:4])
[1] 24 103 65 49
> age <- hoge[,1:4]
> rowSums(age)
[1] 24 103 65 49
> apply(age, 1, sum)
[1] 24 103 65 49
> |
```

GC含量計算部分の説明

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

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

#本番(GC含量情報取得)

```
hoge <- alphabetFrequency(fasta)
CG <- rowSums(hoge[,2:3])           #A,C,G,T,...の数を配列ごとにカウントしCGに格納
ACGT <- rowSums(hoge[,1:4])          #C,Gの総数を計算してCGに格納
GC_content <- sum(CG)/sum(ACGT)       #A,C,G,Tの総数を計算してACGTに格納
                                         #トータルのGC含量の情報を取得
```

#ファイルに保存

```
tmp <- NULL
tmp <- rbind(tmp, c("Total length (bp)", Total_len))
tmp <- rbind(tmp, c("GC content (%)", GC_content))
```

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
GTCTGCCTCAAGCGCCCCAAGTGGGTTGGAGGCCTAACATC
ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAG
GTC
>contig_3
TGTAGGAGAAGGGCGGTATCAGCGTCCACTTACACGATCCGT
GTATGAGGTCGGGCA
>contig_4
CGTGCTGATTCCACACAGCAGTAAACGCGGACCTCTACCTAT
```

オブジェクトCG中には、配列(コンティグ)ごとのCとGのカウント数が格納されている。オブジェクトACGT中には、配列ごとのA, C, G, Tのカウント数が格納されている。例えば49塩基からなるcontig_4中に、ACGTの4種類の塩基が49個、CGの数は25個あることを意味する。sum関数は、ベクトルの要素の和を返す。

R Console

```
> hoge <- alphabetFrequency(fasta)
> CG <- rowSums(hoge[,2:3])
> ACGT <- rowSums(hoge[,1:4])
> GC_content <- sum(CG)/sum(ACGT)
> cd
エラー: オブジェクト 'cd' がありません
> cg
エラー: オブジェクト 'cg' がありません
> CG
[1] 16 65 33 25
> ACGT
[1] 24 103 65 49
> sum(CG)
[1] 139
> sum(ACGT)
[1] 241
> |
```

GC含量計算部分の説明

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

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

#本番(GC含量情報取得)

```
hoge <- alphabetFrequency(fasta)
CG <- rowSums(hoge[,2:3])           #A,C,G,T,...の数を配列ごとにカウントした結果をCGに格納
ACGT <- rowSums(hoge[,1:4])          #C,Gの総数を計算してCGに格納
GC_content <- sum(CG)/sum(ACGT)      #A,C,G,Tの総数を計算してACGTに格納
                                         #トータルのGC含量の情報を取得
```

#ファイルに保存

```
tmp <- NULL
tmp <- rbind(tmp, c("Total length (bp)", Total_len))
```

```
tmp <- rbind(tmp, hoge4.fa)
```

tmp <- write.table(tmp, "hoge4.fa", row.names = FALSE, col.names = FALSE)

>contig_1

CGGACAGCTCCTCGGCATCCGGAT

>contig_2

GTCTGCCTCAAGCGCCCCAAGTGGGTTTGGAGGCCAACATCGCAAGTCG

ACACTCAGTCCGGCCGTCTGGTTGGCAGGGGCAGAGACCCAGCACACCCCT

GTC

>contig_3

TGTAGGAGAAGGGCGGTATCAGCGTCCACTTACACGATCCGTTACTAATT

GTATGAGGTGGGCA

>contig_4

CGTGCTGATTCCACACAGCAGTAAACGCGGACCTCTACCTATGAACATG

ここでは①sum関数を用いて配列全体の総和でGC含量計算をしているが、②sum関数を用いずに「CG/ACGT」とやると、コンティグごとのGC含量を得られる。例えば、contig_1はCGの数が16個で、ACGTの数が24個。それゆえGC含量は $16/24 = 0.6666667$ となる。

R Console window showing R code and results:

```

> sum(CG)/sum(ACGT)
[1] 0.5767635
> CG/ACGT
[1] 0.6666667 0.6310680 0.5076923
[4] 0.5102041
>

```

配列ごとのGC含量計算

- 正規化 | サンプル間 | 3群間 | 複製あり | iDEGES/edgeR(Sun 2013) (last modified 2015/03/30) 推奨 NEW
- 正規化 | サンプル間 | 3群間 | 複製あり | TMM(Robinson 2010) (last modified 2015/03/30) NEW
- 解析 | 一般 | アライメント(ペアワイズ; 基礎1) (last modified 2010/6/8)
- 解析 | 一般 | アライメント(ペアワイズ; 基礎2) (last modified 2010/6/8)
- 解析 | 一般 | アライメント(ペアワイズ; 応用) (last modified 2010/6/8)
- 解析 | 一般 | パターンマッチング (last modified 2013)
- 解析 | 一般 | GC含量(GC contents) (last modified 2013)
- 解析 | 一般 | Sequence logos(Schneider 1990) (last modified 2013)
- 解析 | 上流配列解析 | LDSS(Yamamoto 200)
- 解析 | 上流配列解析 | Relative Appearance R
- 解析 | 基礎 | k-mer | ゲノムサイズ推定(基礎) | grqc
- 解析 | 基礎 | 平均-分散プロット | Technical replicate
- 解析 | 基礎 | 平均-分散プロット | Biological replicate
- 解析 | 新規転写物同定(ゲノム配列を利用) (last modified 2013)
- 解析 | 発現量推定(トランск립トーム配列を利用)
- 解析 | クラスタリング | について (last modified 2014/1)
- 解析 | クラスタリング | サンプル間 | hclust (last modified 2014/1)
- 解析 | クラスタリング | サンプル間 | TCC(Sun 2013)
- 解析 | クラスタリング | 遺伝子間(基礎) | MBCluster
- 解析 | クラスタリング | 遺伝子間(応用) | MBCluster
- 解析 | シミュレーションカウントデータ | について (last modified 2013)
- 解析 | シミュレーションカウントデータ | Technical replicate
- 解析 | シミュレーションカウントデータ | Biological replicate
- 解析 | シミュレーションカウントデータ | Biological replicate

sum関数を用いずに「CG/ACGT」とやって、コンティグごとのGC含量を得るための項目。記述内容がほぼ同じであることが分かる。

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)
CG <- rowSums(hoge[,2:3])
ACGT <- rowSums(hoge[,1:4])
GC_content <- CG/ACGT*100                         #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
                                                       #C,Gの総数を計算してCGに格納
                                                       #A,C,G,Tの総数を計算してACGTに格納
                                                       #%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

```

配列ごとのGC含量計算

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"
out_f <- "hoge1.txt"

#必要なパッケージをロード
library(Biostrings)

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

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

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

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

R Console

```
> #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", "Length")
> write.table(tmp, out_f, sep="\t", append=F, quote=F)
> 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"

配列ごとのGC含量計算

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す。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) × 100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"
out_f <- "hoge1.txt"

#必要なパッケージをロード
library(Biostrings)

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

#本番
hoge <- alphabetFrequency(fasta)          #A,C,G,T,...の$%
CG <- rowSums(hoge[,2:3])                 #C,Gの$%
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", "Length")
write.table(tmp, out_f, sep="\t", append=F, quote=F)
```

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

R Console

```
> #本番
> hoge <- alphabetFrequency(fasta)
> CG <- rowSums(hoge[,2:3])
> ACGT <- rowSums(hoge[,1:4])
> GC_content <- CG/ACGT*100
>
> #ファイルに保存
> tmp <- cbind(names(fasta), CG, ACGT, width(fasta))
> colnames(tmp) <- c("description", "CG", "ACGT", "Length")
> write.table(tmp, out_f, sep="\t", append=F, quote=F)
> tmp
```

	description	CG	ACGT	Length	%GC_contents
[1,]	"contig_1"	"16"	"24"	"24"	"66.666666666667"
[2,]	"contig_2"	"65"	"103"	"103"	"63.1067961165049"
[3,]	"contig_3"	"33"	"65"	"65"	"50.7692307692308"
[4,]	"contig_4"	"25"	"49"	"49"	"51.0204081632653"

ACGT列は4種類の塩基のみの出現数、Length列は配列長情報を表す。配列長は、ACGT以外の全てを含むので、2つの数値の差分(Length - ACGT)がNなどのACGT以外の塩基のトータルの出現回数ということになる。Length \geq ACGTという関係。